UNIVERSITY
OF TRENTO

DEPARTMENT OF INFORMATION AND COMMUNICATION TECHNOLOGY

38050 Povo – Trento (Italy), Via Sommarive 14
http://www.dit.unitn.it

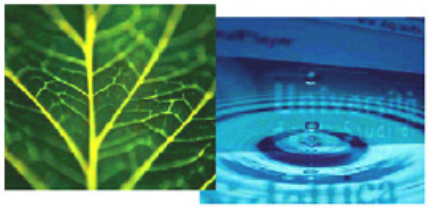
ITERATIVE SCHEMA-BASED SEMANTIC MATCHING

Pavel Shvaiko

December 2006

Technical Report # DIT-06-102

PhD Dissertation



International Doctorate School in Information and Communication Technology

DIT - University of Trento

ITERATIVE SCHEMA-BASED SEMANTIC MATCHING

Pavel Shvaiko

Advisor:

Prof. Fausto Giunchiglia

Università degli Studi di Trento

November 2006

Abstract

An ontology typically provides a vocabulary that describes a domain of interest and a specification of the meaning of terms used in the vocabulary. Depending on the precision of this specification, the notion of ontology encompasses several data and conceptual models, for example, classifications, database schemas, fully axiomatized theories. Ontologies tend to be put everywhere. They are viewed as the silver bullet for many applications, such as information integration, peer-to-peer systems, electronic commerce, semantic web services, social networks, and so on. They, indeed, are a practical means to conceptualize what is expressed in a computer format. However, in open or evolving systems, such as the semantic web, different parties would, in general, adopt different ontologies. Thus, just using ontologies does not reduce heterogeneity: it raises heterogeneity problems to a higher level.

Ontology matching is a promising solution to the semantic heterogeneity problem. It finds correspondences between semantically related entities of the ontologies. These correspondences can be used for various tasks, such as ontology merging, query answering, data translation, or for navigation on the semantic web. Thus, matching ontologies enables the knowledge and data expressed in the matched ontologies to interoperate. This dissertation focuses only on the task of discovering correspondences between various forms of ontologies with a particular consideration of classifications.

Many various solutions of matching have been proposed so far. This work concentrates on a schema-based solution, namely a solution exploiting only the schema information, and not considering instance information. To ground the choice of the solution, this thesis provides a comprehensive coverage of the schema-based approaches used in ontology matching as well as their applications by reviewing state of the art in the field in a uniform way. It also points out how the approach developed in the thesis fits in with existing work.

The thesis proposes the so-called *semantic matching* approach. This approach is based on two key ideas. The first is that correspondences are calculated between entities of ontologies by computing logical relations (e.g., equivalence, subsumption, disjointness), instead of computing coefficients rating match quality in the [0 1] range, as it is the case in many other approaches. The second idea is that the relations are determined by analyzing the meaning which is codified in the elements and the structures of ontologies. In particular, labels at nodes, written in natural language, are automatically translated into propositional formulas which explicitly codify the labels' intended meaning. This allows the translation of the matching problem into a propositional validity problem, which can then be efficiently resolved using sound and complete state of the art propositional satisfiability deciders.

The basic and iterative semantic matching algorithms as well as explanations of the correspondences produced have been designed and developed. The approach has been evaluated on various real world test cases with encouraging results, thus, proving empirically its benefits.

Keywords

Ontology matching, schema matching, ontology alignment, semantic heterogeneity, semantic matching, iterative semantic matching

Acknowledgments

I am extremely grateful to Fausto Giunchiglia, my scientific advisor, for many lessons on how to do research and write articles, for being very supportive in my work, for guiding my entrance into AI domain and life in general. Specifically, I am thankful for the countless hours he spent with me in teaching how to shape the early ideas with the help of examples, turn hard research problems into fun and how to follow the high standards of scholarship, precision and technical depth in a research work. Also, I am thankful for his insightful suggestions that helped me made the right strategic choices at many crucial decision points along these years.

I am grateful to external thesis committee members, Alessandro Artale, Jérôme Euzenat, Nicola Guarino and Stefano Spaccapietra for the time and energy they have spent in reviewing my thesis and their detailed technical feedback.

I thank all my friends and everyone who have contributed to this thesis through many fruitful discussions, technical advice, encouraging words and in many other ways.

Finally, I everlastingly thank my parents, Larysa and Leonid, who have given love, support and understanding over all of these years. I owe very special thanks to my beloved Marlene for accepting my style of living during these years. Her inspiration and love have been an endless source of energy that invaluably helped me in completing this thesis.

Contributions and publications

This work has been developed in collaboration with various people (as the publications indicate) and in particular with: Fausto Giunchiglia, Jérôme Euzenat, Deborah L. McGuinness, Paulo Pinheiro da Silva, and Mikalai Yatskevich.

This thesis makes the following contributions:

- An overview of the ontology matching applications and requirements these applications pose towards a plausible solution;
- A detailed survey of state of the art schema-based ontology matching approaches and systems under a uniform framework;
- Design and development of a new approach to ontology matching, called semantic matching;
- Design and development of the algorithms for semantic matching;
- Creation of a real world data set from the cultural heritage domain for the evaluation of quality results of matching systems;
- Empirical evaluation of the semantic matching approach on various data sets;
- An overview of future trends in the ontology matching field.

Part of the material of the thesis has been published in various conferences, journals and books (in order of appearance):

- [96]: Fausto Giunchiglia and Pavel Shvaiko. Semantic matching. In *Proceedings of the Workshop on Ontologies and Distributed Systems at the International Joint Conference on Artificial Intelligence (IJCAI)*, 2003.

- [97]: Fausto Giunchiglia and Pavel Shvaiko. Semantic matching. *The Knowledge Engineering Review*, 18(3), 2003.

- [98]: Fausto Giunchiglia, Pavel Shvaiko, and Mikalai Yatskevich. S-Match: an algorithm and an implementation of semantic matching. In *Proceedings of the European Semantic Web Symposium (ESWS)*, 2004.

- [213]: Pavel Shvaiko. A classification of schema-based matching approaches. In *Proceedings of the Meaning Coordination and Negotiation Workshop at the International Semantic Web Conference (ISWC)*, 2004.

- [154]: Deborah L. McGuinness, Pavel Shvaiko, Fausto Giunchiglia, and Paulo Pinheiro da Silva. Towards explaining semantic matching. In *Proceedings of the International Workshop on Description Logics (DL) at the International Conference on the Principles of Knowledge Representation and Reasoning (KR)*, 2004.

- [215]: Pavel Shvaiko, Fausto Giunchiglia, Paulo Pinheiro da Silva, and Deborah McGuinness. Web explanations for semantic heterogeneity discovery. In *Proceedings of the European Semantic Web Conference (ESWC)*, 2005.

- [134]: Alan Léger, Lyndon Nixon, Pavel Shvaiko, and Jean Charlet. Semantic web applications: Fields and business cases. The industry challenges the research. In *Proceedings of the International Conference on Industrial Applications of Semantic Web (IASW)*, 2005.

- [99]: Fausto Giunchiglia, Pavel Shvaiko, and Mikalai Yatskevich. Semantic schema matching. In *Proceedings of the International Conference on Cooperative Information Systems (CoopIS)*, 2005.

- [133]: Alain Léger, Lyndon Nixon, and Pavel Shvaiko. On identifying knowledge processing requirements. In *Proceedings of the International Semantic Web Conference (ISWC)*, 2005.

- [214]: Pavel Shvaiko and Jérôme Euzenat. A survey of schema-based matching approaches. *Journal on Data Semantics (JoDS)*, IV, 2005.

- [238]: Anna Zhdanova and Pavel Shvaiko. Community-driven ontology matching. In *Proceedings of the European Semantic Web Conference (ESWC)*, 2006.

- [132]: Alain Léger, Johannes Heinecke, Lyndon Nixon, Pavel Shvaiko, Jean Charlet, Paola Hobson and François Goasdoué. The semantic web from an industry perspective. *Tutorial at Reasoning Web, Second International Summer School*, Springer, 2006.

- [74]: Jérôme Euzenat, Malgorzata Mochol, Pavel Shvaiko, Heiner Stuckenschmidt, Ondřej Šváb, Vojtěch Svátek, Willem Robert van Hage, and Mikalai Yatskevich. Results of the ontology alignment evaluation initiative 2006. In *Proceedings of the International Workshop on Ontology Matching (OM) at the International Semantic Web Conference (ISWC)*, 2006.

- [100]: Fausto Giunchiglia, Pavel Shvaiko, and Mikalai Yatskevich. Discovering missing background knowledge in ontology matching. In *Proceedings of the European Conference on Artificial Intelligence (ECAI)*, 2006.

- [103]: Fausto Giunchiglia, Mikalai Yatskevich, and Pavel Shvaiko. Semantic matching: algorithms and implementation. *Journal on Data Semantics (JoDS)*, IX, 2006. to appear.

- [75]: Jérôme Euzenat and Pavel Shvaiko. *Ontology matching*. Springer-Verlag, Heidelberg (DE), 2007. to appear.

Whenever results of any of these works are reported, proper citations are made in the body of the thesis.

Contents

List of Figures

List of Tables

Introduction

Ontology matching

An ontology typically provides a vocabulary describing a domain of interest and a specification of the meaning of terms used in the vocabulary. Depending on the precision of this specification, the notion of ontology encompasses several data and conceptual models, including classifications, database schemas, fully axiomatized theories. Ontologies tend to be put everywhere. They are viewed as the silver bullet for many applications, such as database integration, peer-to-peer systems, e-commerce, semantic web services, social networks [81]. They, indeed, are a practical means to conceptualize what is expressed in a computer format [37]. However, in open or evolving systems, such as the semantic web, different parties would, in general, adopt different ontologies. Thus, merely using ontologies, like using XML, does not reduce heterogeneity: it raises heterogeneity problems to a higher level.

This thesis is devoted to ontology matching as a solution to the semantic heterogeneity problem faced by computer systems. Ontology matching aims at finding correspondences between semantically related entities of different ontologies. These correspondences may stand for equivalence as well as other relations, such as subsumption, or disjointness, between ontology entities. Ontology entities, in turn, are usually the named entities of ontologies, such as classes, properties or individuals. However, these

entities can also be more complex expressions, such as formulas, concept definitions or term building expressions. Ontology matching results, called alignments, can thus express with various degrees of precision the relations between the ontologies under consideration.

Alignments can be used for various tasks, such as ontology merging, data translation, or for query answering the web. Matching ontologies enables the knowledge and data expressed in the matched ontologies to interoperate. It is thus of utmost importance for the above mentioned applications whose interoperability is jeopardized by heterogeneous ontologies.

Many different matching solutions have been proposed so far from various viewpoints, including databases, information systems, and artificial intelligence. They take advantage of various properties of ontologies, e.g., labels, structures or data instances, and use techniques from different fields, e.g., linguistics, automated reasoning, statistics and data analysis, machine learning. These solutions share some techniques and tackle similar problems, but differ in the way they combine and exploit their results.

Motivating example

To motivate the matching problem, let us use two simple XML schemas that are shown in Figure 1 and exemplify one of the possible situations which arise, for example, when resolving a schema integration task.

Suppose an e-commerce company needs to finalize a corporate acquisition of another company. To complete the acquisition we have to integrate databases of the two companies. The documents of both companies are stored according to XML schemas O1 and O2, respectively. Numbers in boxes are the unique identifiers of the XML elements. A first step in integrating the schemas is to identify candidates to be merged or to have taxonomic relationships under an integrated schema. This step refers to

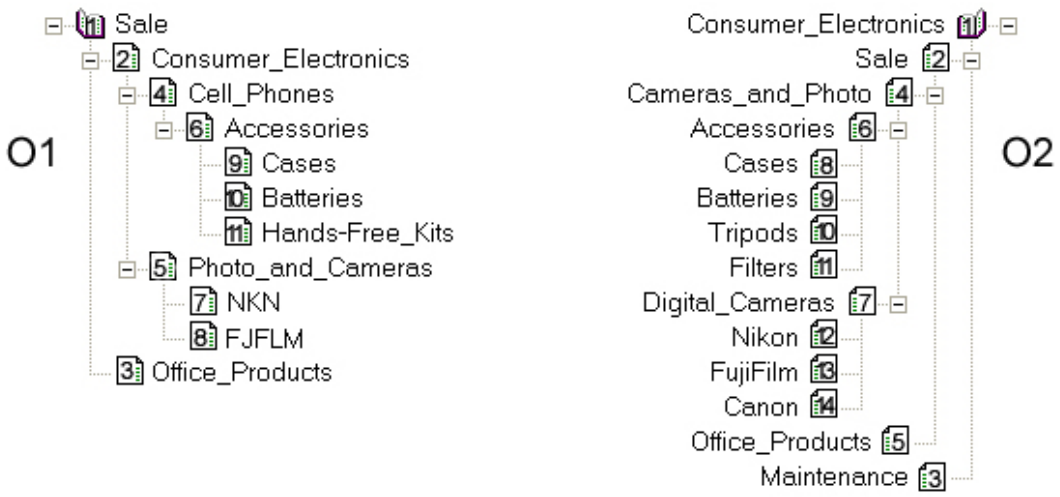


Figure 1: Two XML schemas

a process of schema matching. For example, the elements with labels *Office_Products* in O1 and in O2 are the candidates to be merged, while the element with label *Digital_Cameras* in O2 should be subsumed by the element with label *Photo_and_Cameras* in O1. Once the correspondences between two schemas have been determined, the next step has to generate query expressions that automatically translate data instances of these schemas under an integrated schema.

Solution

Many various solutions of matching have been proposed so far. This work concentrates on a schema-based solution, namely a matching approach exploiting only the schema information, thus not considering instances. It proposes the so-called *semantic matching* approach. This approach is based on two key ideas. The first is that correspondences are calculated between entities of ontologies by computing logical relations (e.g., equivalence, subsumption, disjointness), instead of computing coefficients rating match quality in the [0 1] range, as it is the case of many other

approaches. The second idea is that the relations are determined by analyzing the meaning which is codified in the entities and the structures of ontologies. In particular, labels at nodes, written in natural language, are automatically translated into propositional formulas which explicitly codify the labels' intended meaning. This allows the translation of the matching problem into a propositional validity problem, which can then be efficiently resolved using sound and complete propositional satisfiability deciders.

Structure of the thesis

The thesis is organized in five parts.

Part one is dedicated to the motivation and the definition of the ontology matching problem. The motivation is given in Chapter 1 through a number of applications that can take advantage of matching ontologies and the presentation of how matching contributes to these applications. In Chapter 2, the ontology matching problem is technically defined in various instances of ontology matching occurring in different contexts, such as classifications, databases, XML schemas and finally formal ontologies. It technically defines the ontology matching process and its result: the alignment.

Part two provides a comprehensive coverage of the schema-based approaches used for ontology matching. Chapter 3 defines a classification of the matching approaches, presents some basic methods and matching strategies used for designing an ontology matching system. Chapter 4 presents a large number of state of the art schema-based matching systems, discussed in light of the classifications, methods and strategies of the previous chapter. It also points out how the approach further developed in this thesis fits in with existing work.

Part three is devoted to the semantic matching approach proposed in this thesis. Chapter 5 introduces basic notions and motivations behind the approach. It also discusses the main macro steps realizing the semantic matching algorithm. Chapter 6 discusses how attributes are handled within the semantic matching settings. Chapter 7 presents an extension of the semantic matching approach to deal in a fully automated way with the lack of background knowledge in matching tasks by using semantic matching iteratively. Chapter 8 describes an extension of the semantic matching approach that enables explanation of the answers it produces, thus making the matching result intelligible.

Part four is devoted to the evaluation of ontology matching and semantic matching in particular. Chapter 9 discusses the evaluation criteria for ontology matching approaches as well as the settings in which we ran our experiments. Chapter 10 reports the results of the conducted experiments.

Finally, part five concludes. Chapter 11 summarizes the work done in the thesis. Chapter 12 outlines future trends in the ontology matching field.

Part I

The matching problem

Chapter 1

Applications

Matching metadata models is an important operation in traditional applications, such as ontology integration, schema integration, data warehouses. Typically, these applications are characterized by heterogeneous structural models that are analyzed and matched either manually or semi-automatically at design time. In such applications matching is a prerequisite of running the actual system.

A line of applications that can be characterized by their dynamics, e.g., agents, peer-to-peer systems, web services, is emerging. Such applications, contrary to traditional ones, require (ultimately) a run time matching operation and take advantage of more explicit conceptual models.

Material presented in this chapter has been developed in collaboration with Jérôme Euzenat and published in [214, 75]. Also some lines of work on the topic of this chapter have been supported by the FP6 Knowledge Web[1] Network of Excellence and the FP6 Open Knowledge[2] specific targeted research project, with some results reported in [133, 216].

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

[2] http://openk.org/

In this chapter we first present some well-known applications where matching has been recognized as a plausible solution for a long time. These are ontology evolution (§1.1) and information integration, including schema integration, catalog integration, data warehouses and data integration (§1.2). Then, we discuss some recently emerged applications, such as peer-to-peer information sharing (§1.3), web service composition (§1.4), agent communication (§1.5), and query answering on the web (§1.6).

1.1 Ontology evolution

It is natural that domains of interest, application requirements and the way in which knowledge engineers conceptualize those by means of ontologies undergo changes and evolve over time. Also, ontology development, similar to software code development, is often performed in a distributed and collaborative manner. Therefore, multiple versions of the same ontology often exist. Some applications keep their ontologies up to date, while others may continue to use old ontology versions and update them on their own. These situations arise because knowledge engineers and developers usually do not have a global view of how and where the ontologies have changed. In fact, change logs may not always be available (which is often the case in distributed ontology development). Therefore, developers need to manage and maintain the different versions of their ontologies.

The matching operation is of help here, see Figure 1.1. Its main focus is on discovering the differences, e.g., what ontology entities have been added, deleted or renamed, between two ontology versions [202, 178, 182, 183]. In this scenario it is useful to: (*i*) find the correspondences between the old version (x) and the new version ($x + 1$) of the ontology, (*ii*) generate a transformation by using these correspondences and (*iii*) transform the underlying data instances.

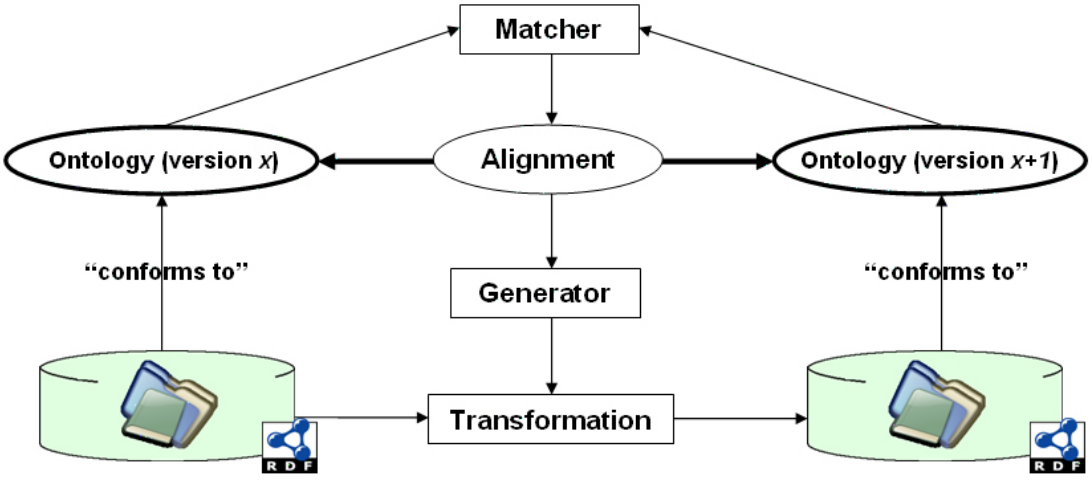


Figure 1.1: Ontology evolution scenario

1.2 Information integration

Information integration is one of the oldest classes of applications where matching is viewed as a plausible solution. Under the information integration heading, we gather here such problems as schema integration [11, 212, 219, 192], data warehousing [26], data integration (also known as enterprise information integration, EII) [44, 233, 65, 114], and catalog integration [1, 121, 31, 99].

A general information integration scenario is presented in Figure 1.2: given a set of local information sources (*Local Ontology 1*, *Local Ontology 2*) potentially storing their data in different formats, e.g., SQL DDL, XML, or RDF, provide users with a uniform query interface via the mediated (or global) ontology *Common Ontology* to all the local information sources. This allows users to avoid querying the local information sources one by one, and obtain a result from them just by querying a common ontology.

For example, if a user poses the query *find a book about ontology matching* to a common ontology, then, an information integration system communicates with local information sources, e.g., *www.amazon.com*, *www.bn.com*,

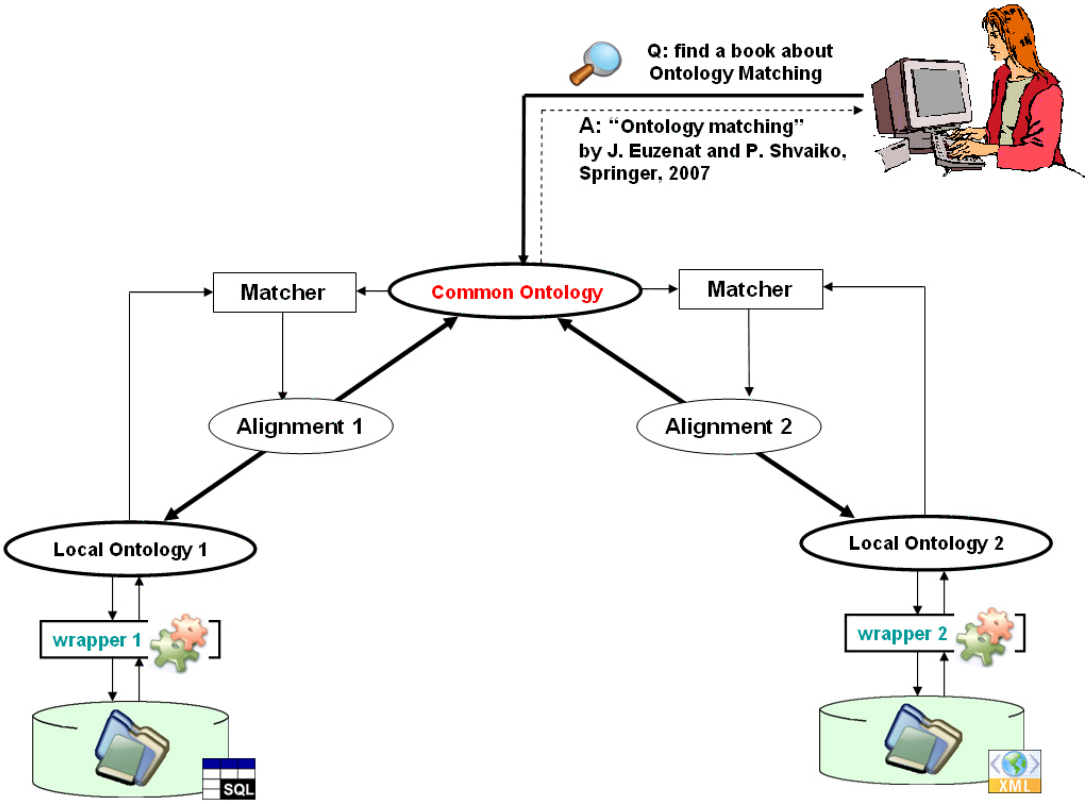


Figure 1.2: A general (centralized) information integration scenario

and returns a reconciled result to the user based on the input provided by those sources. In general, there are a number of macro steps that the information integration system has to perform. These include:

- interpret (rewrite) the query in terms of the common ontology;
- identify the correspondences between semantically related entities of the local information sources and the common ontology;
- translate the relevant data instances of the local information sources (involved in handling the user's request) into a knowledge representation formalism of the information integration system;
- reconcile the results obtained from multiple local information sources, namely detecting and eliminating, e.g., redundancies, duplications,

before returning the final answer.

Most often a step of identifying the correspondences between semantically related entities of the local information sources and the common ontology is referred to as matching. Let us limit our vision of matching to the description above for the moment. We will expand it to some extent in the next sections.

In some concrete information integration scenarios, the common ontology can be either physically existing or virtual. Below, we discuss these scenarios in some detail.

1.2.1 Schema integration

Schema integration is the oldest scenario [11, 212, 221, 220, 192]. Suppose, two (or more) enterprises want to perform either a merger or an acquisition among them. Ultimately, these enterprises have to integrate their databases into a single one. Usually, a first technical step is to identify correspondences between semantically related entities of the schemas. This step is known as matching. Then, by using the identified correspondences, merging the databases is performed. The matching step is still required even if the databases to be integrated are coming from the same domain of interest, e.g., book selling, car rentals. This is because the schemas have been designed and developed independently. In fact, humans follow diverse modeling principles and patterns, even if they have to encode the same real world object. Finally, the schemas to be integrated might have been developed according to different business goals. This makes the matching problem even harder.

Under the schema integration heading we can classify some other scenarios. For example, (tightly-coupled) federated databases [212]. These typically have one global schema providing a unified access to the federation

of component databases. Component databases, in turn, are autonomous. Thus, in this application when, for example, one component schema of the federated database is changed, the federated (global) schema has consequently to be also reconsidered. Matching can help in identifying those changes.

Finally, it is worth noting the applications which we are not discussing here, e.g., distributed database systems [185]. These are usually designed in a centralized way, e.g., by a database administrator, and therefore, semantic heterogeneity does not exist there by construction [70].

1.2.2 Catalog integration

In Business-to-Business (B2B) applications, trade partners store information about their products in electronic catalogs. Typical examples of catalogs are product directories of electronic sales portals, such as *Amazon* or *eBay*. In order for a merchant to participate in the marketplace, e.g., *eBay*, it has to determine correspondences between entries of its catalogs and those of a single catalog of a marketplace. This process of finding correspondences among entries of the catalogs is referred to as the catalog matching problem [31]. Notice that if we look at this problem from a merchant viewpoint, matching has to be performed for each marketplace it would like to participate. Having identified the correspondences between the entries of the catalogs, they are further analyzed in order to generate query expressions that automatically translate data instances between the catalogs. Finally, having matched the catalogs, users of a marketplace have a unified access to the products which are on sale. The above described scenario involving interactions between marketplaces and merchants can be viewed as a typical example of integrating local data sources into a data warehouse, see also [26].

Another catalog integration scenario deals with (typically large-scale)

product classifications, such as *UNSPSC*[3] (The United Nations Standard Products and Services Code) and *eCl@ss*[4] (Standardized Material and Service Classification). In a sense, we can view this scenario as one which enables interoperability among multiple B2B marketplaces, thus, facilitating product exchange schemas between the enterprises subscribing to different product classifications [207]. This is to be achieved by establishing the correspondences between semantically related entities of the standardized product classifications, which is a matching operation as well.

1.2.3 Data integration

Data integration is an approach where integration of information coming from multiple local sources is performed *without* first loading their data into a central warehouse [114]. This allows interoperation across multiple local sources having access to the up-to-date data. Notice that in the above considered catalog integration scenario, merchants are those who have to perform updates of the central warehouse of the marketplace. In this scenario the data integration system provides this functionality.

The scenario is as follows. First, local information sources participating in the application, e.g., bookstore, cultural heritage, are identified. Then, a virtual common ontology is built. Queries are posed over the virtual common ontology, and are then reformulated into queries over the local information sources, e.g., in the cultural heritages application, these might be catalogs of museums. In order to enable semantics-preserving query answering, correspondences between semantically related entities of the local information sources and the virtual ontology are to be established. Establishing these correspondences is known as a matching step.

Query answering is then performed by using these correspondences (map-

[3] http://www.unspsc.org

[4] http://www.eclass.de

pings) within the Local-as-View (LAV), Global-as-View (GAV), or Global-Local-as-View (GLAV) settings [135]. In the LAV approach, local schemas are defined in terms of the global schema, i.e., the mapping is specified by defining each local schema construct as a view over global schema constructs. Queries are processed by means of an inference mechanism that re-expresses the atoms of the global schema in terms of atoms of the local schemas. In GAV, a global schema is defined in terms of the local schemas, i.e., the mapping is specified by writing a definition of each global schema construct as a view over local schema constructs. Queries are processed by means of unfolding, i.e., by expanding the atoms according to their definitions (so as to come up with local schema relations). GLAV, in turn, is a mixed approach. We can think of it as a variation of the LAV approach that allows the head of the view definition to contain any query on the local schema.

1.3 Peer-to-peer information sharing

Peer-to-peer (P2P) is a distributed communication model in which parties (also called peers) have equivalent functional capabilities in providing each other with data and services [236]. P2P networks became popular through a file, e.g., pictures, music, videos, books, sharing paradigm. There exist a number of industry-strength P2P file sharing systems, e.g., *Kazaa*, *Edonkey*, and *BitTorrent*. These applications describe file contents by a simple schema (set of attributes, such as *title* of a song, its *author*, etc.) to which all the peers in the network have to subscribe. These schemas cannot be modified locally by a single peer. Therefore, in the above mentioned systems the semantic heterogeneity problem (at the schema level) does not exist by construction.

The use of a single system schema violates the *total autonomy* of peers. Although industry-strength P2P systems allow peers to connect to and disconnect from the network at any time, thereby respecting some forms of peers autonomy, such as *participation autonomy*, they still restrict the *design autonomy* of peers, in matters such as how to describe the data, what constraints to use on the data [236].

If peers are meant to be totally autonomous, they may use different terminologies and metadata models in order to represent their data, even if they refer to the same domain of interest. Thus, in order to establish (meaningful) information exchange between peers, one of the steps is to identify and characterize relationships between their ontologies. This is a matching operation. Having identified the relationships between ontologies, these can be used for the purpose of query answering, e.g., using techniques applied in data integration systems, see §1.2.

Following the argument of total autonomy of peers, more advanced P2P systems relax the homogeneity requirement of classical P2P systems: they allow peers to use independent schemas and ontologies [236, 122, 173, 203], see Figure 1.3. In this scenario, it is useful to: (i) match relevant parts of the ontologies, (2) generate a mediator for translating queries and sometimes for translating answers.

Such applications pose additional requirements on matching solutions. In P2P settings which respect total autonomy of peers, an assumption that all the peers rely on one global schema, as in data integration, cannot be made because the global schema may need to be updated any time the system evolves [104]. While in the case of data integration schema matching can be performed at design time, in P2P applications peers need to coordinate their databases on-the-fly, therefore ultimately requiring run time schema matching. Finally, incomplete and approximate answers, as long as they are good enough for the application, are also acceptable in

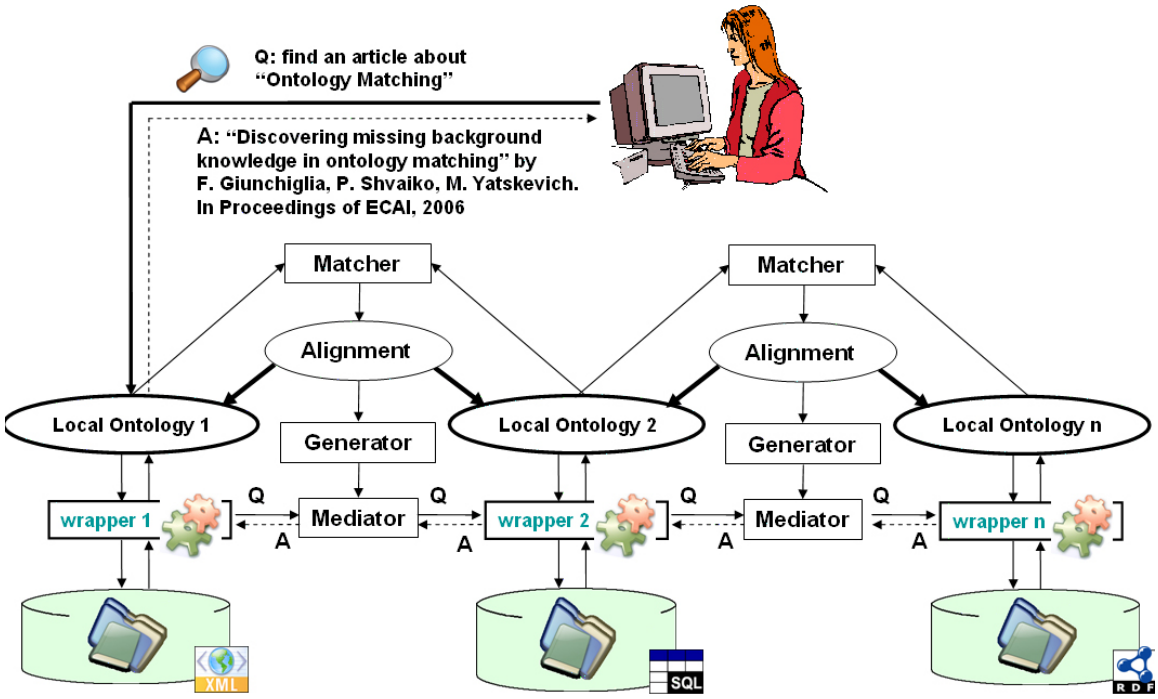


Figure 1.3: P2P query answering

such settings. This is the case because some mappings involved in query answering may become temporarily unavailable or invalid [216].

1.4 Web service composition

Web services are processes that expose their interface to the web so that users can invoke them. Semantic web services provide a richer and more precise way to describe the services through the use of knowledge representation languages and ontologies. Web service discovery and integration is the process of finding a web service able to deliver a particular service and composing several services in order to achieve a particular goal, see [191, 157, 184, 82]. However, semantic web services descriptions have no reasons to be expressed by reference to exactly the same ontologies. Henceforth, both for finding the adequate service and for interfacing services it

is necessary to establish the correspondences between the terms of the descriptions. This can be provided through matching the corresponding ontologies, see Figure 1.4. For instance, if some service provides its output description in some ontology and another service uses a second ontology for describing its input, matching both ontologies will be used for (*i*) checking that what is delivered by the first service matches what is expected by the second one, (*ii*) verifying preconditions of the second service, and (*iii*) generating a mediator able to transform the output of the first service in order to be input to the second one.

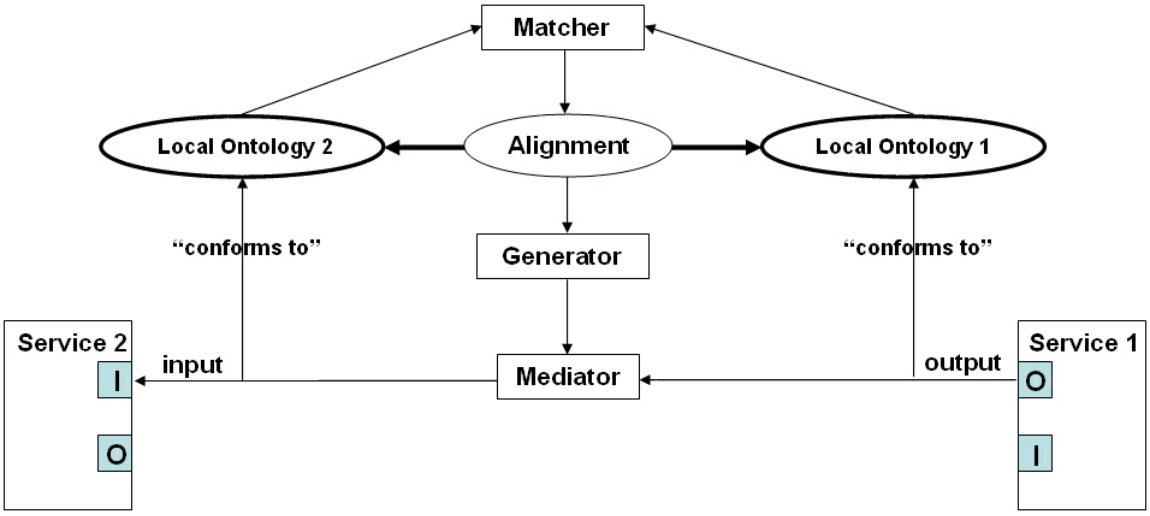


Figure 1.4: Web service composition

1.5 Agent communication

Agents are computer entities characterized by autonomy and capacity of interaction. They communicate through speech-act inspired languages, such as the FIPA Agent Communication Language [84, 83], which determine the "envelope" of the messages and enable agents to position them within a particular interaction context. The actual content of messages is expressed

in knowledge representation languages and often refer to some ontology. As a consequence, when two autonomous and independently designed agents meet, they have the possibility of exchanging messages, but little chance to understand each others if they do not share the same content language and ontology. Thus, it is necessary to provide the possibility for these agents to match their ontologies in order to either translate their messages or integrate bridge axioms in their own models, see [228, 234, 129]. One solution to this problem is to have an ontology alignment protocol that can be interleaved with any other agent interaction protocol and which could be triggered upon receiving a message expressed in a foreign ontology. As a consequence, agents meeting each other for the first time and using different ontologies would be able to negotiate the matching of terms in their respective ontologies and to translate the content of the message they exchange with the help of the alignment, see Figure 1.5. In this scenario it is useful, for example, to: (*i*) match relevant parts of the ontologies used by each of the agents, (*ii*) generate a message translator from *Local Ontology* 1 to *Local Ontology* 2 and (*iii*) apply this translator to the message.

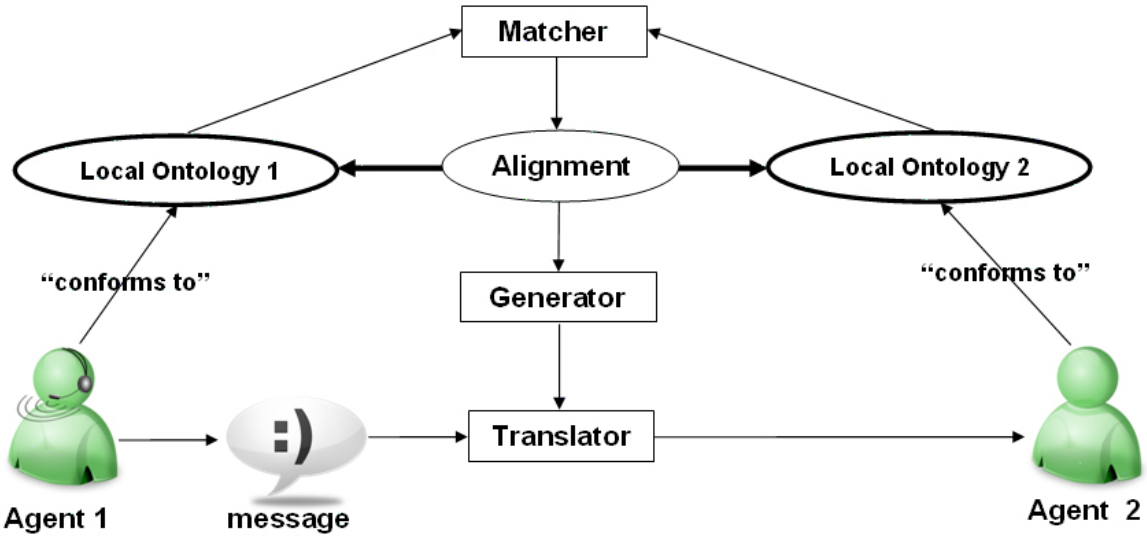


Figure 1.5: Agent communication

1.6 Query answering on the web

In some of the above considered scenarios, e.g., schema integration, data integration, it was assumed that queries were specified by using the terminology of a global schema. In the scenario under consideration, we discard this assumption, and therefore users are free to pose queries by using their own terminology. Then, an information integration system has to interpret (rewrite) the terms used in the query, into the predefined ontology entities of the system, for instance. This rewriting can be viewed as matching. The rest of the query answering process usually proceeds in a similar way as discussed in the previous scenarios. Let us now consider a slight variation of this scenario in distributed settings with the help of examples of the AquaLog and PowerAqua systems [140, 139].

As an example, suppose that a query answering system such as AquaLog [140] is aware of an ontology about academic life which has been populated to describe knowledge related to some university [204]. Also, let us suppose that the following query is posed to the system: *Which projects are related to researchers working with ontologies?* To answer this query, Aqualog needs to interpret it in terms of entities available in the system ontology. For this, Aqualog first translates this query into the following triples: $\langle projects, related\ to, researchers\rangle$ and $\langle researchers, working, ontologies\rangle$. Then it attempts to match these triples to the concepts of the underlying ontology. For example, the term *projects* should be identified to refer to the ontology concept *Project* and *ontologies* is assumed equivalent to the *ontologies* instance of the *Research-Area* concept.

Currently, the scope of AquaLog is limited by the amount of knowledge encoded in the ontology of the system. A new version of AquaLog, called PowerAqua [139], extends its predecessor, as well as some other systems with similar goals, such as Observer [162], towards "open" query answer-

ing. PowerAqua aims to select and aggregate information derived from multiple heterogeneous ontologies on the web. Matching constitutes the core of this selection task. Notice that, unlike AquaLog, matching is now performed between the triples and many on-line ontologies (not just the single ontology of the system). It is not necessary to match all query triples within one ontology. When no ontology concept is found for an element of a triple, the use of more general concepts is also acceptable. Also, it is not necessary to try to match the whole ontology against the query, but only the relevant fragments.

1.7 Summary

The above considered scenarios suggest that matching metadata models is a major issue. Moreover, a need for matching is not limited to one particular application. In fact, it exists in any application that communicates through ontologies. Thus, it is natural that in future more examples of applications requiring matching will appear, e.g., ontology repair [155].

Since semantic heterogeneity is an intrinsic problem of any application involving more than one party, it is reasonable to consider ontology matching as a unified object of study. However, there are notable differences in the way these applications use matching. The application related differences must be clearly identified in order to provide the best suited solution in each case.

These applications can be ordered according to their *dynamics*, namely autonomy of parties participating in an application and rate of changes in an application (see Figure 1.6).

For example, Figure 1.6 shows that agent communication and query answering have a more dynamic profile compared to the other applications. In fact, agents, besides having the ability to enter or leave the network or

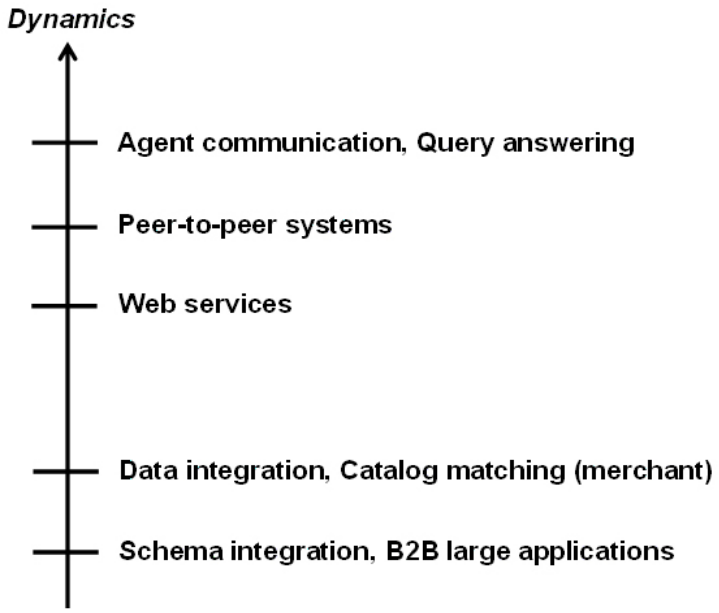


Figure 1.6: Distribution of some applications with regard to their dynamics

to change their ontologies at any moment (as in the peer-to-peer case), are also able to negotiate the alignments and potential mismatches.

Data integration and merchant catalog matching, due to multiple new merchants being willing to participate in marketplaces, have a higher dynamics than schema integration, where typically only a small and limited number of parties participate. Finally, the two bottom classes of applications represent traditional applications, while the three top classes of applications can be considered as dynamic applications. The uneven step in the middle of the dynamics axis in Figure 1.6 is used to stress the above mentioned distinction.

Another dimension along which these applications differ is the purpose for which they perform matching:

- schema integration requires the ability to merge the schemas under consideration into a single schema (the transformations apply at the ontological level and instances translation apply at the data level);

- data integration requires the ability to translate data instances residing in multiple local schemas according to a global schema definition in order to enable query answering over the global schema;

- peer-to-peer systems and more generally query answering systems require bidirectional mediators able to translate queries (ontological level) and translate back answers (data level).

- agent communication requires translators for messages sent from one agent to another, which apply at the data level; similarly, semantic web services require one-way data translations for composing services.

This leads to different requirements for different applications. We summarize what we have found to be the most important requirements to matching solutions according to the applications considered in this chapter, see Table 1.1.

These general requirements concern:

- the type of available input a matching system can rely on, such as schema or instance information. There are cases when data instances are not available, for instance due to security reasons [46] or when there are no instances given beforehand. Therefore, these applications require only a matching solution able to work without instances (here schema-based method).

- some specific behaviour of matching, such as requirements of (*i*) being *automatic*, i.e., not relying on user feed-back; (*ii*) being *correct*, i.e., not delivering incorrect matches; (*iii*) being *complete*, i.e., delivering all the matches; and (*iv*) being performed at *run time*.

- the use of the matching result as described above. In particular, how the identified alignment is going to be processed, e.g., by merging the

Application	instances	run time	automatic	correct	complete	operation
Ontology evolution (§1.1)	√			√	√	transformation
Schema integration (§1.2)	√			√	√	merging
Catalog integration (§1.2)	√			√	√	data translation
Data integration (§1.2)	√			√	√	query answering
P2P information sharing (§1.3)		√				query answering
Web service composition (§1.4)		√	√	√		data mediation
Multi-agent communication (§1.5)		√	√	√	√	data translation
Query answering (§1.6)	√	√				query reformulation

Table 1.1: Summary of applications requirements

data or conceptual models under consideration or by translating data instances among them.

Some of these hard requirements can be derived into comparative (or non-functional) requirements, such as speed, degree of correctness or completeness. These requirements are useful for comparing solutions on a scale instead of with absolute requirements such as mentioned before. Moreover, they allow to trade a requirement, e.g., completeness, for another more important one, e.g., speed.

As an overview of this chapter indicates, there are many different applications which can take advantage of matching ontologies. However, in spite of a common need for matching, the application matching requirements are quite different.

Chapter 2

The matching problem

In a distributed and open system, such as the semantic web and in many other applications presented in the previous chapter, heterogeneity cannot be avoided. Different actors have different interests and habits, use different tools and knowledge, and most often, at different levels of detail. These various reasons for heterogeneity lead to diverse forms of heterogeneity, and, therefore, should be carefully taken into consideration.

Material presented in this chapter has been developed in collaboration with Jérôme Euzenat and published in [214, 75]. Also some work on the topic of this chapter has been supported by the FP6 Knowledge Web[1] Network of Excellence.

In this chapter we first present various existing ways of expressing knowledge that are found in diverse applications (§2.1). We introduce several justifications for heterogeneity (§2.2). These should help the design of a matching strategy as a function of the kind of heterogeneity that has to be addressed. Finally, we define the ontology matching problem (§2.3).

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

2.1 Vocabularies, schemas and ontologies

So far we have considered ontologies without being precise about their meaning. An ontology can be viewed as a set of assertions that are meant to model some particular domain. Usually, the ontology defines a vocabulary used by a particular application. In various areas of computer science there are different data and conceptual models that can be thought of as ontologies. These are, for instance, database schemas, entity-relationship models, directories, thesauri, XML schemas and formal ontologies (see [235, 110, 111] for an in-depth discussion of what is considered to be a proper ontology). These and other examples are given in decreasing order of formality in Figure 2.1.

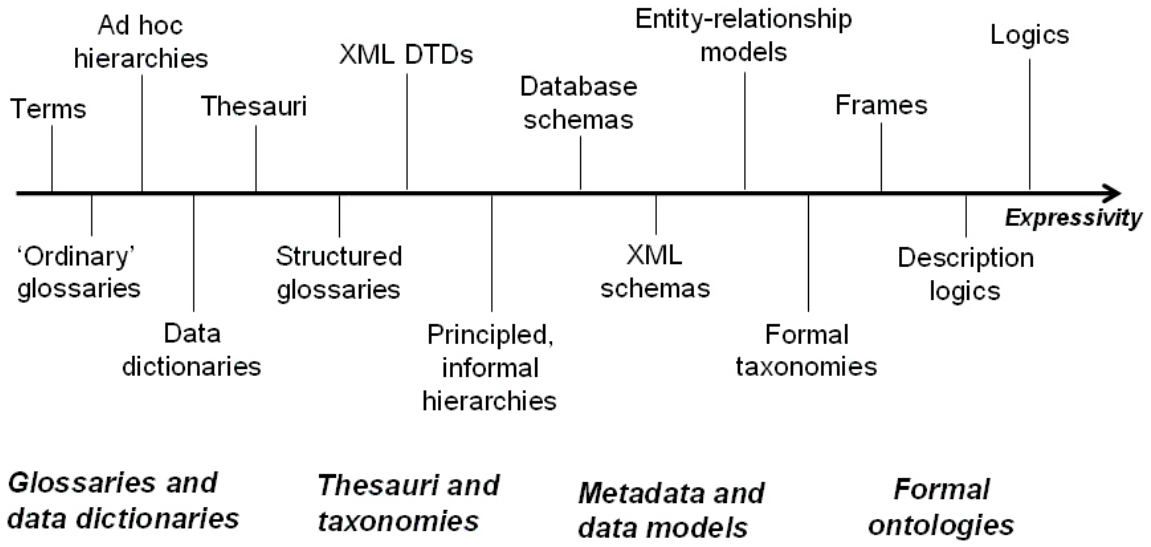


Figure 2.1: Various forms of ontologies ordered by their expressivity (adapted from [109, 225]).

Thus, a top level ontology is supposed to have an explicit well defined semantics, whereas the interpretation of directories in a file system is mostly implicit. In fact, it depends only on what its creator had in mind, i.e., the

meaning of labels, the background knowledge, and the context in which those labels occur are all implicit, and therefore, these are not a part of a directory specification.

We provide below a number of examples of various forms of ontologies of Figure 2.1 and exemplify some heterogeneity problems encountered in these forms.

2.1.1 Classifications

A taxonomy is a partially ordered set of taxons (classes) in which one taxon is greater than another one only if what it denotes includes what is denoted by the other. Classifications or directories are taxonomies that are used by companies for presenting products on sale, by libraries for storing books, or by individuals to classify files on a personal computer. Some well-known examples of classifications include those of *DMOZ*[2], *Google*[3] and *Yahoo*[4]. These classifications are hierarchies of folders identified by labels and containing items, such as bookmarks, or products. The semantics of these folders is given by the items they ultimately contain [95]. Obviously, each independent entity tends to develop its own directory based on its own needs and tastes.

Finally, there exist some consensus classifications. In library science, the Dewey classification has been used for more than a century for classifying books by topics [42]. In natural sciences, the principled classification of species represents another example [206].

[2] http://dmoz.org
[3] http://www.google.com/dirhp
[4] http://www.yahoo.com

2.1.2 Relational database schemas

Relational databases require the data to be organized in a predefined way as tables or relations. A relational schema specifies the names of the tables as well as their types: the names and types of the columns of each table. The relational model also includes the notion of a key for each table: a subset of the columns that uniquely identifies each row. Finally, a column in a table may be specified as a foreign key pointing to a column in another table. This is used to keep referential constraints among various entities.

Finally, it is worth mentioning widely used languages for specifying relational schemas, such as Structured Query Language (SQL) as well as some of its recent versions, e.g., SQL:1999 and SQL:2003. These support many modeling capabilities, such as user-defined types, aggregation, generalization, etc.

2.1.3 XML schemas

Document Type Definition (DTD) and XML schemas have been introduced for specifying the structure of XML documents. The main ingredients of XML schemas include elements, attributes, and types. In turn, elements can be either complex for specifying nested sub-elements, or simple for specifying built-in datatypes, such as *string*, for an element or attribute. XML schemas are rather complementary to classifications: instead of describing how things are classified, they describe how things are made from the inside. Even if element definitions can be extended or restricted as sub-categories of a classification, the emphasis is on their structure: the extension of an element is made by providing the elements which are modified in this structure. The sequential aspect of XML documents is part of the element specification, though it can be overruled.

In fact, these schemas are a shape according to which future documents

are to be created, as opposed to an ontology, which is a description of existing, external objects. The specialization hierarchy in XML schema is a type hierarchy that defines which kind of elements can occupy the place of another kind, e.g., if a shelf contains books, then putting a biography on this shelf is authorized. In principle, this classification structure does not have to correspond to any natural classification of the objects expressed themselves.

2.1.4 Conceptual models

Often database researchers do not consider directly the relational schema but are rather concerned with the underlying entity-relationship model [145]. Conceptual models cover what was properly described as such in [37], as well as entity-relationship models [45] that aim at abstracting databases, and UML [27] models that aim at abstracting object-oriented programs. A spatio-temporal aspect of conceptual models is addressed in [194].

These models offer a rich way of expressing entities which in this case can be meant as entities of some modeled domain, like people in a database or specification of entities to be created like programs. They offer constructors for organizing classes in a hierarchy as well as constructors for describing the internal structure of objects. They thus offer the best of both worlds: classifications and databases.

2.1.5 Ontologies

It is nowadays common to see classifications or conceptual models to be promoted as ontologies. Ontologies contain most of the features of entity-relationship models, and thus, most parts of the kind of schemas considered above.

The distinctive feature of ontologies is the existence of a model theoretic

semantics: ontologies are logic theories, see for details [108]. Thus, their interpretation is not left to the users that read the diagrams or to the database management systems implementing them, it is specified explicitly by set of equations. The semantics provides the rules for interpreting the syntax. It does not provide the meaning directly but constrains the possible interpretations of what is declared.

Ontologies are expressed in an ontology language. There are a large variety of languages for expressing ontologies [222], for example, OWL [217, 52], an ontology language recommended by the W3C. Fortunately, most of these languages share the same kinds of entities, often with different names but comparable interpretations.

2.2 Types of heterogeneity

The goal of matching ontologies is to reduce heterogeneity between them. Heterogeneity does not lie solely in the differences of ultimate goals of the applications according to which they have been designed or in the expression formalisms in which ontologies have been encoded. There have been many different classifications to types of heterogeneity [11, 212, 36, 127, 106, 120, 125, 14, 233, 128, 71, 48, 115, 93, 29]. Some of them focus on mismatches [128], others rather mention interoperability levels [71]. We consider here the most obvious types of heterogeneity:

Syntactic heterogeneity occurs when two ontologies are not expressed in the same ontology language. This obviously happens when comparing, for instance, a classification with a conceptual model. This also happens when two ontologies are modeled by using different knowledge representation formalisms, for instance, OWL and F-logic. This kind of mismatch is generally tackled at the theoretical level when one establishes equivalences between constructs of different languages.

Thus, it is sometimes possible to translate ontologies between different ontology languages while still preserving the meaning [76].

Terminological heterogeneity occurs due to variations in names when referring to the same entities in different ontologies. This can be caused by the use of different natural languages, e.g., *Paper* vs *Articulo*, different technical sublanguages, e.g., *Paper* vs *Memo*, the use of synonyms, e.g., *Paper* vs *Article*, etc.

Conceptual heterogeneity, also called semantic heterogeneity in [71] and logical mismatch in [128], stands for the differences in modeling the same domain of interest. This can happen due to the use of different (and, sometimes, equivalent) axioms for defining concepts or due to the use of totally different concepts, e.g., geometry axiomatized with points as primitive objects or geometry axiomatized with spheres as primitive objects. Also, as noted in [128] and [231], there is a difference between the conceptualization mismatch, which relies on the differences between modeled concepts, and the explicitation mismatch, which relies on the way these concepts are expressed. Finally, in the context of conceptual differences, [15] identifies three important reasons for these to hold, namely *difference in coverage*, *difference in granularity* and *difference in perspective*.

Semiotic heterogeneity, also called pragmatic heterogeneity in [29], is concerned with how entities are interpreted by a human. Indeed, entities which have exactly the same interpretation are often interpreted by humans with regard to the context, for instance, of how they are ultimately used. This kind of heterogeneity is difficult for the computer to detect and even more difficult to solve, because it is out of its reach. The intended use of entities has a great impact on their interpretation, therefore, matching entities which are not meant to be

used in the same context is often error-prone. Given the limited grasp that a computer can have on these issues, we do not deal with semiotic heterogeneity here.

Usually, several types of heterogeneity occur together. This thesis is only concerned with reducing the terminological and (to a certain extent) conceptual types of heterogeneity, which are both often referred to as semantic heterogeneity.

2.3 Problem statement

There have been different formalizations of matching and its result, see, for example, [23, 135, 123, 29, 239]. We provide here a general definition, following the work in [214].

The *matching* operation determines the alignment A' for a pair of ontologies $\mathsf{O1}$ and $\mathsf{O2}$. There are some other parameters which can extend the definition of the matching process, namely: (*i*) the use of an input alignment A, which is to be completed by the process; (*ii*) the matching parameters, p, e.g., weights, thresholds; and (*iii*) external resources used by the matching process, r, e.g., common knowledge and domain specific thesauri. Technically, this process can be defined as follows.

The matching process can be viewed as a function f which, from a pair of ontologies $O1$ and $O2$ to match, an input alignment A, a set of parameters p and a set of oracles and resources r, returns a new alignment A' between these ontologies:

$$A' = f(O1, O2, A, p, r)$$

This can be schematically represented as illustrated in Figure 2.2.

It can be useful to specifically consider the matching of many ontologies within the same process. We call this multiple matching.

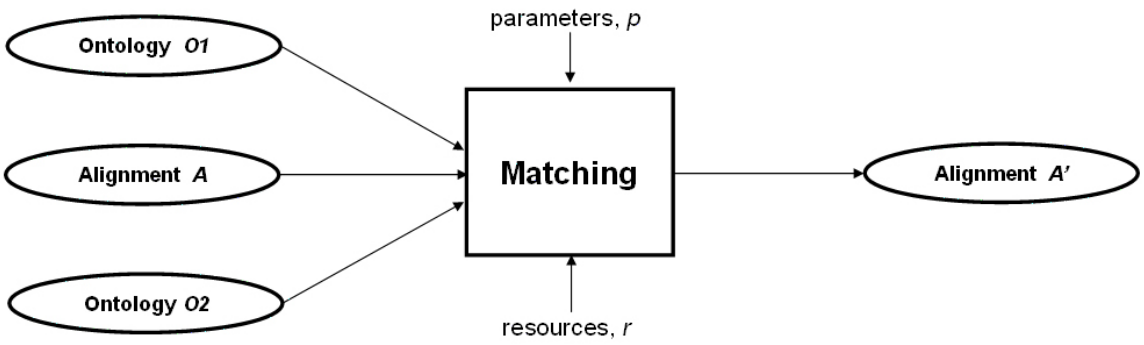


Figure 2.2: The matching process

The multiple matching process can be viewed as a function f which, from a set of ontologies to match $\{O1, \ldots On\}$*, an input multi-alignment* A*, a set of parameters* p *and a set of oracles and resources* r*, returns a new multi-alignment* A' *between these ontologies:*

$$A' = f(O1, \ldots On, A, p, r)$$

The matching process is the main subject of this thesis. However, before discussing its internals, let us first consider what it provides: the alignment.

Alignments express the correspondences between entities belonging to different ontologies. We focus here on matching between two ontologies. In case of multiple matching, the definitions can be straightforwardly extended by using n-ary correspondences. A correspondence must express the two corresponding entities and the relation that is supposed to hold between them. We provide the definition of the alignment following the work in [73, 29].

Given two ontologies, a correspondence is a 5-tuple:

$$\langle id, e_1, e_2, n, R \rangle,$$

such that

- *id is a unique identifier of the given correspondence;*

- *e_1 and e_2 are the entities (e.g., tables, XML elements, properties, classes) of the first and the second ontology, respectively;*
- *n is a confidence measure (typically in the [0 1] range) holding for the correspondence between the entities e_1 and e_2;*
- *R is a relation (e.g., equivalence ($=$), more general ($\sqsupseteq$), disjointness ($\perp$), overlapping ($\sqcap$)) holding between the entities e_1 and e_2.*

The correspondence $\langle id, e_1, e_2, n, R\rangle$ asserts that the relation R holds between the ontology entities e_1 and e_2 with confidence n. The usage of confidences is that the higher the degree, the most likely the relation holds.

Given two ontologies O1 and O2, an alignment is made up of a set of correspondences between pairs of entities belonging to O1 and O2, respectively.

For example, in Figure 1 (p.xxi), according to some matching algorithm based on linguistic and structure analysis, the confidence measure (for the fact that the equivalence relation holds) between entities with labels *Photo_and_Cameras* in O1 and *Cameras_and_Photo* in O2 could be 0.67. Suppose that this matching algorithm uses a threshold of 0.55 for determining the resulting alignment, i.e., the algorithm considers all the pairs of entities with a confidence measure higher than 0.55 as correct correspondences. Thus, our hypothetical matching algorithm should return to the user the following correspondence:

$$\langle id_{5,4}, Photo\_and\_Cameras, Cameras\_and\_Photo, 0.67, =\rangle.$$

However, the relation between the same pair of entities, according to another matching algorithm which is able to determine that both entities mean the same thing, could be exactly the equivalence relation (without computing the confidence measure). Thus, returning to the user

$$\langle id_{5,4}, Photo\_and\_Cameras, Cameras\_and\_Photo, n/a, = \rangle.$$

By analogy with mathematical functions, it is useful to define some properties of the alignments. These apply when the only considered relation is equality (=). One can ask for a total alignment with regard to one ontology, i.e., all the entities of one ontology must be successfully mapped to the other one. This property is purposeful whenever thoroughly transcribing knowledge from one ontology to another is the goal: there is no entity that cannot be translated.

One can also require the mapping to be injective with regard to one ontology, i.e., all the entities of the other ontology is part of at most one correspondence. Injectivity is useful in ensuring that entities that are distinct in one ontology remain distinct in the other one. In particular, this contributes to the reversebility of alignments.

Usual mathematical properties apply to these alignments. In particular, a total alignment from O1 to O2 is a surjective alignment from O2 to O1. A total alignment from both O1 and O2 which is injective from one of them is a bijection. In mathematical English, an injective function is said to be *one-to-one* and a surjective function to be *onto*. Due to the wide use among matching practitioners of the term *one-to-one* for a bijective, i.e., both injective and surjective, alignment, we will only use one-to-one for bijective.

In conceptual models and databases, the terms multiplicity or cardinality denote the constraints on a relation. Usual notations are 1:1, 1:m, n:1 or n:m. If we consider only total and injective property, denoted as 1 for injective and total, ? for injective, + for total and * for none, and the two possible orientations of the alignments, from O1 to O2 and from O2 to O1, the multiplicities become: ?:?, ?:1, 1:?, 1:1, ?:+, +:?, 1:+, +:1, +:+, ?:*, *:?, 1:*, *:1, +:*, *:+, *:* [72].

2.4 Summary

In this chapter, we have first described different kinds of data and conceptual models and observed an expressivity hierarchy of them. Although, there are differences between these forms of ontologies, we believe that techniques developed for matching each of them can be of a mutual benefit. In fact, on the one side, for example, schema matching is usually performed with the help of techniques trying to guess the meaning encoded in the schemas. On the other side, ontology matching systems primarily try to exploit knowledge explicitly encoded in the ontologies. In real world applications, schemas and ontologies usually have both well defined and obscure terms, and contexts in which they occur, therefore, solutions from both problems would be mutually beneficial.

Then, we focused on identifying what semantic heterogeneity is and why it requires matching. We have presented various reasons why mismatches can occur between ontologies. Their variety and the fact that they often occur together constrains to develop multiple approaches for matching ontologies. Finally, we have defined the action of matching ontologies and its result: the alignment.

Part II

State of the art: ontology matching approaches

Chapter 3

Ontology matching techniques

Having defined what the matching problem is, we overview some classifications of the techniques that can be used for solving this problem. In particular, surveys on the topic through the recent years have been provided in [198, 233, 123]; while the major contributions of the previous decades are presented in [11, 130, 219, 124, 192]. The work presented in [123] focuses on current state of the art in ontology matching. Authors review recent approaches, techniques and tools. The survey of [233] concentrates on approaches to ontology-based information integration and discusses general matching approaches that are used in information integration systems. However, none of the above mentioned works provide a comparative review of the existing ontology matching techniques and systems. On the contrary, the survey of [198] is devoted to a classification of database schema matching approaches and a comparative review of matching systems. Notice that these three works address the matching problem from different perspectives (artificial intelligence, information systems, databases) and analyze disjoint sets of systems. [214] have attempted at considering the above mentioned works together, focusing on schema-based matching methods, and aiming to provide a common conceptual basis for their analysis.

Material presented in this chapter has been developed in collaboration with Jérôme Euzenat and published in [214, 75]. Also a part of work on the topic of this chapter has been supported by the FP6 Knowledge Web[1] Network of Excellence.

In this chapter we first consider various dimensions on which a classification of matching techniques can be elaborated (§3.1). We then present our classification based on several of these dimensions (§3.2). We also discuss some alternative classifications of matching approaches that have been proposed so far in the literature (§3.3). Finally, we outline a number of plausible matching strategies used in building a matching system (§3.4).

3.1 Matching dimensions

There are many independent dimensions along which algorithms can be classified. Following the definition of the matching process in Figure 2.2 (p.29), we may primarily classify algorithms according to (*i*) the input of the algorithms, (*ii*) the characteristics of the matching process, and (*iii*) the output of the algorithms. The other characteristics, such as parameters, resources, and input alignments, are considered less important. Let us discuss these three main aspects in turn.

3.1.1 Input dimensions

These dimensions concern the kind of input on which algorithms operate. As a first dimension, algorithms can be classified depending on the data or conceptual models in which ontologies are expressed. For example, the Artemis system [39] (see §4.1.5) supports the relational, object-oriented, and entity-relationship models; Cupid [146] (see §4.1.9) supports XML and relational models; QOM [69] (see §4.2.3) supports RDF and OWL mod-

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

els. A second possible dimension depends on the kind of data that the algorithms exploit: different approaches exploit different information in the input ontologies. Some of them rely only on schema-level information, e.g., Cupid [146] (see §4.1.9), COMA [58] (see §4.1.10); others rely only on instance data, e.g., GLUE [62]; and others exploit both schema- and instance-level information, e.g., QOM [69] (see §4.2.3). Even with the same data models, matching systems do not always use all available constructs, e.g., the approach presented in this thesis (see Chapter 6), when dealing with attributes discards information about datatypes, e.g., *string* or *integer*, and uses only the attributes names. In general, some algorithms focus on the labels assigned to the entities, some consider their internal structure and the types of their attributes, and others consider their relations with other entities (see next section for details).

3.1.2 Process dimensions

A classification of the matching process could be based on its general properties, as soon as we restrict ourselves to formal algorithms. In particular, it depends on the *approximate* or *exact* nature of its computation. Exact algorithms compute the absolute solution to a problem; approximate algorithms sacrifice exactness to performance (e.g., [69]). All the techniques discussed in the remainder of the thesis can be either approximate or exact. Another dimension for analyzing the matching algorithms is based on the way they interpret the input data. We identify three large categories based on the intrinsic input, external resources, or some semantic theory of the considered entities. We call these three categories *syntactic*, *external*, and *semantic*, respectively; and discuss them in detail in the next section.

3.1.3 Output dimensions

Apart from the information that matching systems exploit and how they manipulate it, the other important class of dimensions concerns the form of the result they produce. The form of the alignment might be of importance: is it a one-to-one alignment between the ontology entities? Has it to be a final correspondence? Is any relation suitable?

Other significant distinctions in the output results have been indicated in [97]. One dimension concerns whether systems deliver a graded answer, e.g., that the correspondence holds with 98% confidence or 4/5 probability; or an all-or-nothing answer, e.g., that the correspondence definitely holds or not. In some approaches correspondences between ontology entities are determined using distance measures. This is used for providing an alignment expressing equivalence between these entities. Another dimension concerns the kind of relations between entities a system can provide. Most of the systems focus on equivalence ($=$), while a few other are able to provide a more expressive result, e.g., equivalence, subsumption ($\sqsubseteq$), incompatibility ($\perp$), see for details [31, 97, 98].

There are many dimensions that can be taken into account when attempting at classifying matching methods. In the next section we present a classification of elementary techniques that draws simultaneously on several such criteria.

3.2 A classification of matching techniques

In this section we discuss only schema-based elementary matchers. Therefore, only schema level information is considered, not instance data[2]. The

[2]Prominent solutions of instance-based ontology matching can be found in [60, 63].

exact/approximate opposition has not been used because each of the methods described below can be implemented as exact or approximate algorithm, depending on the goals of the matching system. To ground and ensure a comprehensive coverage for our classification we have analyzed state of the art approaches used for schema-based matching. The bibliography part reports a partial list of works which have been scrutinized pointing to (some of) the most important contributions. We have used the following guidelines for building our classification:

Exhaustivity. The extension of categories dividing a particular category must cover its extension (i.e., their aggregation should give the complete extension of the category);

Disjointness. In order to have a proper tree, the categories dividing one category should be pairwise disjoint by construction;

Homogeneity. In addition, the criterion used for further dividing one category should be of the same nature (i.e., should come from the same dimension). This usually helps guaranteeing disjointness;

Saturation. Classes of concrete matching techniques should be as specific and discriminative as possible in order to provide a fine grained distinction between possible alternatives. These classes have been identified following a *saturation* principle: they have been added/modified till the saturation was reached, namely taking into account new techniques did not require introducing new classes or modifying them.

Notice that disjointness and exhaustivity of the categories ensures stability of the classification, namely new techniques will not occur in between two categories. Classes of matching techniques represent the state of the art. Obviously, with appearance of new techniques, they might be extended and further detailed.

We build on the previous work of classifying automated schema matching approaches of [198]. The classification of [198] distinguishes between *elementary* (individual) matchers and *combinations* of matchers[3]. Elementary matchers comprise *instance-based* and *schema-based*, *element-* and *structure-level*, *linguistic-* and *constrained-based* matching techniques. Also *cardinality* and *auxiliary information* (e.g., thesauri, global schemas) can be taken into account.

For classifying elementary schema-based matching techniques, we introduce two synthetic classifications (see Figure 3.1), based on what we have found the most salient properties of the matching dimensions. These two classifications are presented as two trees sharing their leaves. The leaves represent classes of elementary matching techniques and their concrete examples. Two synthetic classifications are:

- *Granularity/Input Interpretation* classification is based on (*i*) granularity of match, i.e., element- or structure-level, and then (*ii*) on how the techniques generally interpret the input information;

- *Kind of Input* classification is based on the kind of input which is used by elementary matching techniques.

The overall classification of Figure 3.1 can be read both in descending (focusing on how the techniques interpret the input information) and ascending (focusing on the kind of manipulated objects) manner in order to reach the *Basic Techniques* layer. Let us discuss in turn *Granularity/Input Interpretation*, *Basic Techniques*, *Kind of Input* layers together with supporting arguments for the categories/classes introduced at each layer.

Elementary matchers are distinguished by the *Granularity/Input interpretation* layer according to the following classification criteria:

[3]Combinations of matchers are discussed in §3.4

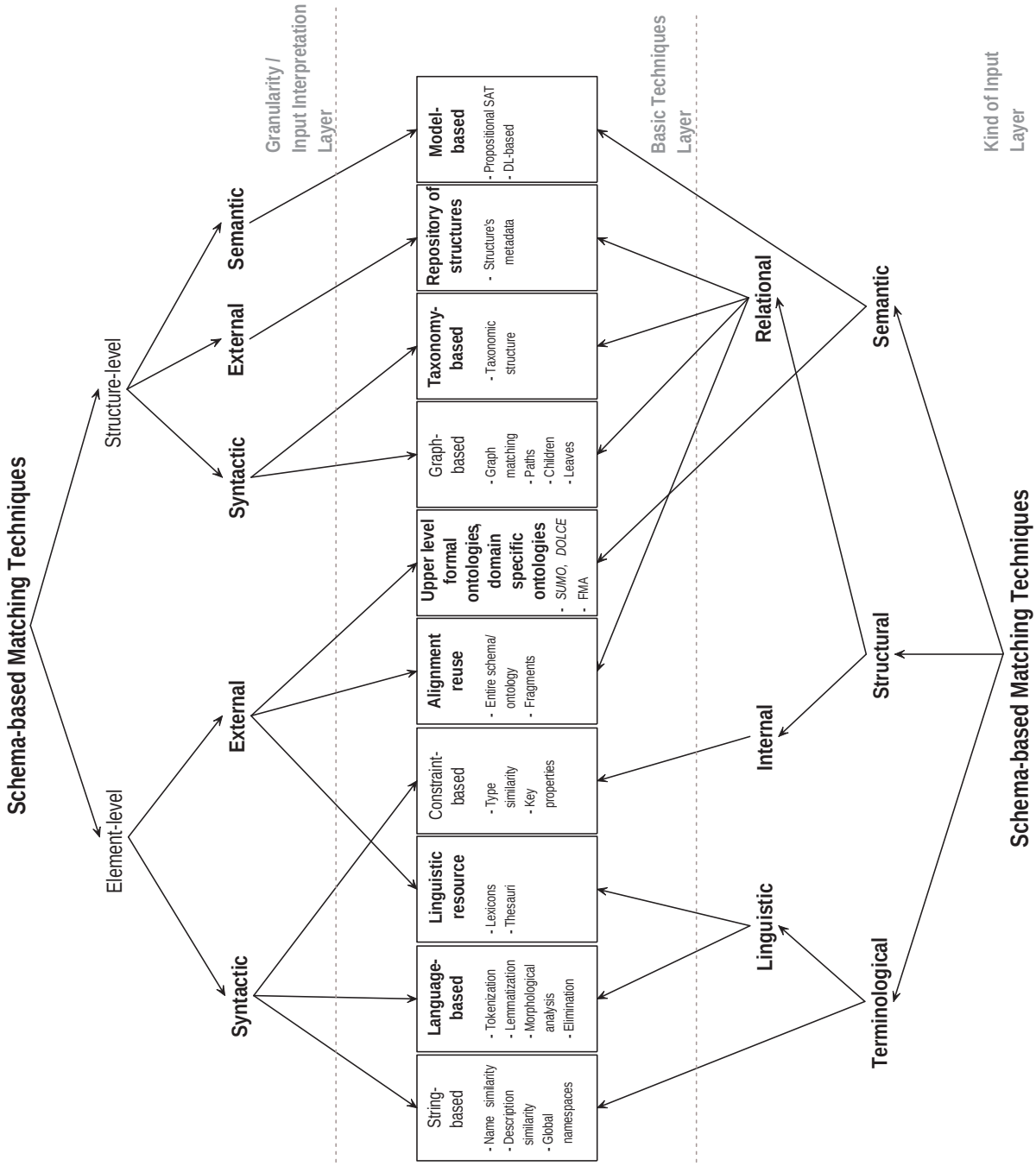


Figure 3.1: A retained classification of elementary schema-based matching techniques

- *Element-level vs structure-level.* Element-level matching techniques compute correspondences by analyzing entities in isolation, ignoring their relations with other entities. Structure-level techniques compute correspondences by analyzing how entities appear together in a structure. This criterion is the same as first introduced in [198].

- *Syntactic vs external vs semantic.* The key characteristic of the syntactic techniques is that they interpret the input as a function of its sole structure following some clearly stated algorithm. External are the techniques exploiting auxiliary (external) resources of a domain and common knowledge in order to interpret the input. These resources might be human input or some thesaurus expressing the relationships between terms. The key characteristic of the semantic techniques is that they use some formal semantics (e.g., model-theoretic semantics) to interpret the input and justify their results. In case of a semantic based matching system, exact algorithms are complete (i.e., they guarantee a discovery of all the possible alignments) while approximate algorithms tend to be incomplete.

To emphasize the differences with the initial classification of [198], the new categories/classes are marked in bold face. In particular, in the *Granularity/Input Interpretation* layer we detail further (with respect to [198]), the element- and structure-level of matching by introducing the *syntactic* vs *semantic* vs *external* criteria. The reasons of having these three categories are as follows. Our initial criterion was to distinguish between *internal* and *external* techniques. By *internal* we mean techniques exploiting information which comes only with the input ontologies. *External* techniques are as defined above. *Internal* techniques can be further detailed by distinguishing between *syntactic* and *semantic* interpretation of input, also as defined above. However, only limited, the same distinction can be intro-

duced for the *external* techniques. In fact, we can qualify some oracles, e.g., WordNet [163], SUMO [174], DOLCE [88], as syntactic or semantic, but not a user input. Thus, we do not detail *external* techniques any further and we omit in Figure 3.1 the theoretical category of *internal* techniques (as opposed to *external*). Notice that we also omit in further discussions element-level semantic techniques, since semantics is usually given in a structure, and, hence, there are no element-level semantic techniques.

Distinctions between classes of elementary matching techniques in the *Basic Techniques* layer of our classification are motivated by the way a matching technique interprets the input information in each concrete case. In particular, a label can be interpreted as a string (a sequence of letters from an alphabet) or as a word or a phrase in some natural language, a hierarchy can be considered as a graph (a set of nodes related by edges) or a taxonomy (a set of concepts having a set-theoretic interpretation organized by a relation which preserves inclusion). Thus, we introduce the following classes of elementary ontology matching techniques at the element-level: *string-based*, *language-based*, *based on linguistic resources*, *constraint-based*, *alignment reuse*, and *based on upper level and domain specific formal ontologies*. At the structure-level we distinguish between: *graph-based*, *taxonomy-based*, *based on repositories of structures*, and *model-based techniques*.

The *Kind of Input* layer classification is concerned with the type of input considered by a particular technique:

- The first level is categorized depending on which kind of data the algorithms work on: strings (*terminological*), structure (*structural*) or models (*semantics*). The two first ones are found in the ontology descriptions, the last one requires some semantic interpretation of the ontology and usually uses some semantically compliant reasoner to deduce the correspondences.

- The second level of this classification decomposes further these categories if necessary: *terminological* methods can be *string-based* (considering the terms as sequences of characters) or based on the interpretation of these terms as linguistic objects (*linguistic*). The structural methods category is split into two types of methods: those which consider the *internal* structure of entities (e.g., attributes and their types) and those which consider the relation of entities with other entities (*relational*).

Notice that following the above mentioned guidelines for building a classification the *terminological* category should be divided into *linguistic* and *non-linguistic* techniques. However, since *non-linguistic* techniques are all *string-based*, this category has been discarded.

We discuss below the main classes of the *Basic Techniques* layer (also indicating in which matching systems they are exploited) according to the above classification in more detail. The order follows that of the *Granularity/Input Interpretation* classification and these techniques are divided in two sections concerning element-level techniques (§3.2.1) and structure-level techniques (§3.2.2). Finally, in Figure 3.1, techniques which are marked in italic (techniques based on upper level ontologies) have not been implemented in any matching system yet. However, we are arguing why their appearance seems reasonable in the near future.

3.2.1 Element-level techniques

String-based techniques

These techniques are often used in order to match names and name descriptions of ontology entities. They consider strings as sequences of letters in an alphabet. They are typically based on the following intuition: the more similar the strings, the more likely they denote the same concepts.

A comparison of different string matching techniques, from *distance* like functions to *token-based distance* functions can be found in [47]. Usually, distance functions map a pair of strings to a real number, where a smaller value of the real number indicates a greater similarity between the strings. Some examples of string-based techniques which are extensively used in matching systems are *prefix*, *suffix*, *edit distance*, and *n-gram*:

- *Prefix.* This test takes as input two strings and checks whether the first string starts with the second one. *Prefix* is efficient in matching cognate strings and similar acronyms (e.g., *int* and *integer*), see, for example [146, 58, 159, 99]. This test can be transformed in a smoother distance by measuring the relative size of the prefix and the ratio.

- *Suffix.* This test takes as input two strings and checks whether the first string ends with the second one (e.g., *phone* and *telephone*), see, for example [146, 58, 159, 99].

- *Edit distance.* This distance takes as input two strings and computes the edit distance between the strings. That is, the number of *insertions*, *deletions*, and *substitutions* of characters required to transform one string into another, normalized by the length of the longest string. For example, the edit distance between *NKN* and *Nikon* is 0.4. Some of matching systems exploiting the given technique are discussed in [58, 181, 99].

- *N-gram.* This test takes as input two strings and computes the number of common n-grams (i.e., sequences of n characters) between them. For example, $trigram(3)$ for the string *nikon* are *nik*, *iko*, *kon*. Thus, the distance between *nkon* and *nikon* would be 1/3. Some of matching systems exploiting the given test are discussed in [58, 99].

Language-based techniques

These techniques consider names as words in some natural language (e.g., English). They are based on Natural Language Processing (NLP) techniques exploiting morphological properties of the input words.

- *Tokenization.* Names of entities are parsed into sequences of *tokens* by a tokenizer which recognizes punctuation, cases, blank characters, digits, etc. For example, *Hands-Free_Kits* becomes ⟨*hands, free, kits*⟩ [99].

- *Lemmatization.* The strings underlying tokens are morphologically analyzed in order to find all their possible basic forms. For example, *Kits* becomes *Kit* [99].

- *Elimination.* The tokens that are articles, prepositions, conjunctions, and so on, are marked (by some matching algorithms, e.g., [146]) to be discarded.

Usually, the above mentioned techniques are applied to names of entities before running string-based or lexicon-based techniques in order to improve their results. However, we consider these language-based techniques as a separate class of matching techniques, since they can be naturally extended, for example, in a distance computation (by comparing the resulting strings or sets of strings).

Constraint-based techniques

These are algorithms which deal with the internal constraints being applied to the definitions of entities, such as types, cardinality of attributes, and keys. We omit here a discussion of matching keys as these techniques appear in our classification without changes with respect to the original

publication [198]. However, we provide a different perspective on matching datatypes and cardinalities.

- *Datatypes comparison* involves comparing the various attributes of a class with regard to the datatypes of their value. Contrary to objects that require interpretations, the datatypes can be considered objectively and it is possible to determine how a datatype is close to another (ideally this can be based on the interpretation of datatypes as sets of values and the set-theoretic comparison of these datatypes, see [226, 227]). For instance, the datatype *day* can be considered closer to the datatype *workingday* than the datatype *integer*. This technique is used in [78].

- *Multiplicity comparison* attribute values can be collected by a particular construction (set, list, multiset) on which cardinality constraints are applied. Again, it is possible to compare the so constructed datatypes by comparing (*i*) the datatypes on which they are constructed and (*ii*) the cardinality that are applied to them. For instance, a set of between 2 and 3 children is closer to a set of 3 people than a set of 10-12 flowers (if children are people). This technique is used in [78].

Linguistic resources

Linguistic resources, such as common knowledge or domain specific thesauri are used in order to match words (in this case names of ontology entities are considered as words of a natural language) based on linguistic relations between them (e.g., synonyms, hyponyms).

- *Common knowledge thesauri.* The approach is to use common knowledge thesauri to obtain meaning of terms used in ontologies. For

example, WordNet [163, 80] is an electronic lexical database for English (and other languages [4, 232]), where various *senses* (possible meanings of a word or expression) of words are put together into sets of synonyms. Relations between ontology entities can be computed in terms of bindings between WordNet senses, see, for instance [97, 31]. For example, in Figure 1 (p.xxi), a sense-based matcher may learn from WordNet (with a prior morphological preprocessing of labels performed) that *Camera* in O1 is a hypernym for *Digital Camera* in O2, and, therefore conclude that entity *Digital_Cameras* in O2 should be subsumed by the entity *Photo_and_Cameras* in O1. Another type of matchers exploiting thesauri is based on their structural properties, e.g., WordNet hierarchies. In particular, hierarchy-based matchers measure the distance, for example, by counting the number of arcs traversed, between two concepts in a given hierarchy, see [101]. Several other distance measures for thesauri have been proposed in the literature, e.g., [200, 197].

- *Domain specific thesauri.* These thesauri usually store some specific domain knowledge, which is not available in the common knowledge thesauri, (e.g., proper names) as entries with synonym, hypernym and other relations. For example, in Figure 1, entities *NKN* in O1 and *Nikon* in O2 are treated by a matcher as synonyms from a domain thesaurus look up: syn key - "NKN:Nikon = syn" [146].

Alignment reuse

These techniques represent an alternative way of exploiting external resources, which record alignments of previously matched ontologies. For instance, when we need to match ontology o' and o'', given the alignments between o and o', and between o and o'' from the external resource, storing

previous match operations results. The alignment reuse is motivated by the intuition that many ontologies to be matched are similar to already matched ontologies, especially if they are describing the same application domain. These techniques are particularly promising when dealing with large ontologies consisting of hundreds and thousands of entities. In these cases, first, large match problems are decomposed into smaller subproblems, thus generating a set of ontology fragments matching problems. Then, reuse of previous match results can be more effectively applied at the level of ontology fragments rather than at the level of entire ontologies. The approach was first introduced in [198] and later was implemented as two matchers, i.e., (*i*) reuse alignments of entire ontologies, or (*ii*) their fragments [58, 8, 199].

Upper level and domain specific formal ontologies

These techniques use as external sources of knowledge upper level and domain specific formal ontologies. Examples of the upper level ontologies are the Suggested Upper Merged Ontology (SUMO) [174] and Descriptive Ontology for Linguistic and Cognitive Engineering (DOLCE) [88]. The key characteristic of these ontologies is that they are logic-based systems, and therefore, matching techniques exploiting them can be based on the analysis of interpretations. Thus, these are semantic techniques. For the moment, we are not aware of any matching systems which use these kind of techniques. However, it is quite reasonable to assume that this will happen in the near future. In fact, for example, the DOLCE ontology aims at providing a formal specification (axiomatic theory) for the top level part of WordNet. Therefore, systems exploiting WordNet now in their matching process (and aware of some of its limitations [89]) might also consider using DOLCE as a potential extension.

Domain specific formal ontologies can also be used as external sources of background knowledge. Such ontologies are focusing on a particular domain and use terms in a sense that is relevant only to this domain and which is not related to similar concepts in other domains. For example, in the anatomy domain, an ontology such as The Foundational Model of Anatomy (FMA)[4] can be used as the context for the other medical ontologies to be matched (as long as it is known that the reference ontology covers the ontologies to be matched). This can be particularly useful for providing the missing structure when matching poorly structured resources [2].

3.2.2 Structure-level techniques

Graph-based techniques

These are graph algorithms which consider the input as labeled graphs. The applications (e.g., database schemas, or ontologies) are viewed as graph-like structures containing terms and their inter-relationships. Finding the correspondences between elements of such graphs corresponds to solving a form of the graph homomorphism problem [91]. Usually, the similarity comparison between a pair of nodes from the two ontologies is based on the analysis of their positions within the graphs. The intuition behind is that, if two nodes from two ontologies are similar, their neighbors might also be somehow similar. Below, we present some particular matchers representing this intuition.

- *Graph homomorphism.* There have been done a lot of work on graph (tree) matching in graph theory and also with respect to ontology matching applications, see, for example, [210, 211]. Graph homomorphism is a combinatorial problem that can be computationally expensive. It is usually solved by approximate methods. In ontology

[4] http://sig.biostr.washington.edu/projects/fm/AboutFM.html

matching, the problem is encoded as an optimization problem (finding the graph matching minimizing some distance like the dissimilarity between matched objects) which is further resolved with the help of a graph matching algorithm. This optimization problem is solved through a fix-point algorithm (improving gradually an approximate solution until no improvement is made). Examples of such algorithms are given in [159] and [78]. Some other (particular) matchers handling directed acyclic graphs (DAGs) and trees are *children*, *leaves*, and *relations*.

- *Children.* The (structural) similarity between inner nodes of the graphs is computed based on similarity of their children nodes, that is, two non-leaf schema elements are structurally similar if their immediate children sets are highly similar. A more complex version of this matcher is implemented in [58].

- *Leaves.* The (structural) similarity between inner nodes of the graphs is computed based on similarity of leaf nodes, that is, two non-leaf schema elements are structurally similar if their leaf sets are highly similar, even if their immediate children are not [146, 58].

- *Relations.* The similarity computation between nodes can also be based on their relations. For example, in one of the possible ontology encodings of schemas of Figure 1 (p.xxi), if class *Photo_and_Cameras* relates to class *NKN* by relation *hasBrand* in one ontology, and if class *Digital_Cameras* relates to class *Nikon* by relation *hasMarque* in the other ontology, then knowing that classes *Photo_and_Cameras* and *Digital_Cameras* are similar, and also relations *hasBrand* and *hasMarque* are similar, we can infer that *NKN* and *Nikon* may be similar too [143].

Taxonomy-based techniques

These are also graph algorithms which consider only the specialization relation. The intuition behind taxonomic techniques is that *is-a* links connect terms that are already similar (being a subset or superset of each other), therefore their neighbors may be also somehow similar. This intuition can be exploited in several different ways:

- *Bounded path matching.* Bounded path matchers take two paths with links between classes defined by the hierarchical relations, compare terms and their positions along these paths, and identify similar terms, see, for instance [181]. For example, in Figure 1 (p.xxi), given that element *Digital_Cameras* in O2 should be subsumed by the element *Photo_and_Cameras* in O1, a matcher would suggest *FJFLM* in O1 and *FujiFilm* in O2 as an appropriate match.

- *Super(sub)-concepts rules.* These matchers are based on rules capturing the above stated intuition. For example, if super-concepts are the same, the actual concepts are similar to each other. If sub-concepts are the same, the compared concepts are also similar [55, 69].

Repository of structures

Repositories of structures store ontologies and their fragments together with pairwise similarity measure, e.g., coefficients in the [0 1] range between them. Notice that unlike the alignment reuse, repository of structures stores only similarities between ontologies, not alignments. In the following, to simplify the presentation, we call ontologies or their fragments as structures. When new structures are to be matched, they are first checked for similarity against the structures which are already available in the repository. The goal is to identify structures which are sufficiently similar to be worth matching in more detail, or reusing already existing align-

ments, thus, avoiding the match operation over the dissimilar structures. Obviously, the determination of similarity between structures should be computationally cheaper than matching them in full detail. The approach of [199], to matching two structures proposes to use some metadata describing these structures, such as structure name, root name, number of nodes, maximal path length, etc. These indicators are then analyzed and are aggregated into a single coefficient, which estimates similarity between them. For example, two ontologies may be found as an appropriate match if they both have the same number of nodes.

Model-based

These are algorithms which handle the input based on its semantic interpretation (e.g., model-theoretic semantics). Thus, they are well grounded deductive methods. Examples are propositional satisfiability (SAT) and description logics (DL) reasoning techniques.

- *Propositional satisfiability (SAT).* This method is the core of the approach presented in this thesis. As from [31, 97, 98, 32], the approach is to decompose the graph (tree) matching problem into the set of node matching problems. Then, each node matching problem, namely pairs of nodes with possible relations between them is translated into a propositional formula of form:

$$Axioms \rightarrow rel(context_1, context_2),$$

  and checked for *validity*. The *Axioms* part encodes background knowledge (e.g., *Digital_Cameras* $\rightarrow$ *Cameras* codifies the fact that *Digital_Cameras* is less general than *Cameras*), which is used as premises to reason about relations *rel* (e.g., $=$, $\sqsubseteq$, $\sqsupseteq$, $\perp$) holding between the nodes $context_1$ and $context_2$ (e.g., node 7 in O1 and node 12 in O2 of

Figure 1 (p.xxi)). A propositional formula is valid if and only if its negation is unsatisfiable. The unsatisfiability is checked by using SAT solvers. Notice that SAT deciders are correct and complete decision procedures for propositional satisfiability, and therefore, they can be used for an exhaustive check of all the possible correspondences.

- *DL-based techniques.* The SAT-based approach computes the satisfiability of theory merging both ontologies along an alignment. Propositional language used for codifying matching problems into propositional validity problems is limited in its expressivity, namely it allows for handling only unary predicates. Thus, it cannot handle, for example, binary predicates, such as properties or roles. However, the same procedure can be carried within description logics (expressing properties). In description logics, the relations (e.g., $=$, $\sqsubseteq$, $\sqsupseteq$, $\perp$) can be expressed as a function of subsumption [33]. In fact, first merging two ontologies (after renaming) and then testing each pair of concepts and roles for subsumption is enough for aligning terms with the same interpretation (or with a subset of the interpretations of the others). For instance, suppose that we have one ontology introducing classes *company*, *employee* and *micro-company* as a company with at most 5 employees, and another ontology introducing classes *firm*, *associate* and *SME* as a firm with at most 10 associates. If we know that all *associate*s are *employee*s and we already have established that *firm* is equivalent to *company*, then we can deduce that a *micro-company* is a *SME*.

There are examples in the literature of DL-based techniques used in relevant to ontology matching applications. For example, in spatio-temporal database integration scenario, as first motivated in [193] and later developed in [218, 194] the inter-schema correspondences are initially proposed

by the integrated schema designer and are encoded together with input schemas in $\mathcal{ALCRP}(\mathcal{S}_2 \oplus \mathcal{T})$ language. Then, DL reasoning services are used to check the satisfiability of the two source schemas and the set of inter-schema correspondences. If some objects are found unsatisfied, then the inter-schema correspondences should be reconsidered.

Another example, is when DL-based techniques are used in query processing scenario [162]. The approach assumes that correspondences between pre-existing domain ontologies are already specified in a declarative manner (e.g., manually). User queries are rewritten in terms of pre-existing ontologies and are expressed in Classic [28], and further evaluated against real world repositories, which are also subscribed to the pre-existing ontologies. An earlier approach for query answering by terminological reasoning is described in [12].

Finally, a very similar problem to ontology matching is addressed within the system developed for matchmaking in electronic marketplaces [54]. Demand D and supply S requests are translated from natural language sentences into Classic [28]. The approach assumes the existence of a predefined domain ontology T, which is also encoded in Classic. Matchmaking between a supply S and a demand D is performed with respect to the predefined domain ontology T. Reasoning is performed with the help of the NeoClassic reasoner in order to determine the *exact match* ($T \models (D \sqsubseteq S)$) and ($T \models (S \sqsubseteq D)$), *potential match* (if $D \sqcap S$ is satisfiable in T), and *nearly miss* (if $D \sqcap S$ is unsatisfiable in T). The system also provides a logically based matching results rank operation.

3.3 Other classifications

Let us now consider some other available classifications of matching techniques.

[66] introduced a classification based on two orthogonal dimensions. These can be viewed as horizontal and vertical dimensions. The horizontal dimension includes three layers that are built one on top of another:

Data layer: This is the first layer. Matching between entities is performed here by comparing only data values of simple or complex datatypes.

Ontology layer: This is the second layer which, in turn, is further divided, following the cake of [22], into four levels. These are semantic nets, description logics, restrictions and rules. For example, at the level of semantic nets, ontologies are viewed as graphs with concepts and relations, and, therefore, matching is performed by comparing only these. The description logics level brings a formal semantics account to ontologies. Matching at this level includes, for example, determining taxonomic similarity based on the number of subsumption relations separating two concepts. This level also takes into account instances of entities, therefore, for example, assessing concepts to be the same, if their instances are similar. Matching at the levels of restrictions and rules is typically based on the idea that if, e.g., similar rules between entities exist, these entities can be regarded as similar. This typically requires processing of higher order relations.

Context layer: Finally, this layer is concerned with the practical usage of entities in the context of an application. Matching is performed here by comparing the usages of entities in ontology-based applications. One of the intuitions behind such matching methods is that similar entities are often used in similar contexts.

The vertical dimension represents specific *domain knowledge* which can be situated at any layer of the horizontal dimension. Here, the advantage of external resources of domain specific knowledge, e.g., Dublin Core[5] for the bibliographic domain, is considered for assessing the similarity between entities of ontologies.

[61] classifies matching techniques into (*i*) rule-based and (*ii*) learning-based. Typically, rule-based techniques work with schema-level information, such as entity names, datatypes and structures. Some examples of rules are that two entities match if their names are similar or if they have the same number of neighbor entities. Learning-based approaches often work with instance-level information, thereby performing matching, for example, by comparing value formats and distributions of data instances underlying the entities under consideration. However, learning can also be done at the schema-level and from the previous matches [59].

[237, 13] classify matching methods into three categories following the cognitive theory of meaning and communication between agents:

Syntactic: This category represents methods that use purely syntactic methods to compute alignments. Some examples of such methods include string-based techniques, e.g., edit distance between strings and graph-based techniques.

Pragmatic: This category represents methods that rely on comparison of data instances underlying the entities under consideration in order to compute alignments. Some examples of such methods include automatic classifiers, e.g., Bayesian classifier [64, 149, 60] and formal concepts analysis [90, 223].

Conceptual: This category represents methods that work with concepts and compare their meanings in order to compute alignments. Some

[5] http://dublincore.org/

examples of such methods include techniques exploiting external thesauri, such as WordNet, in order to compare meanings among the concepts under consideration.

There were also some classifications mixing the process dimension of matching together with either input dimension or output dimension. For example, [56] extends the work of [198] by adding a *reuse-oriented* category of techniques on top of schema-based vs. instance-based separation, meaning that reuse-oriented techniques can be applied at schema and instance level. However, these techniques can also include some input information, such as user input or alignments obtained from previous match operations.

[97] classified matching approaches into *syntactic* and *semantic*. At the matching process dimension these correspond to syntactic and conceptual categories of [237] respectively. However, these have been also constrained by a second condition dealing with the output dimension: syntactic techniques return coefficients in the [0 1] range, while semantic techniques return logical relations, such as equivalence and subsumption.

Finally, we notice that the more the ontology matching field progresses, the wider the variety of techniques that come into use at different levels of granularity.

3.4 Matching strategies

The basic techniques presented earlier in §3.2 are the building blocks on which a matching solution is built. In particular, the following aspects of building a working matching system have to be taken into account as well:

- How to organize the combination of various basic matching algorithms. A natural way of composing the basic matchers consists of improving the matching through the use of sequential composition [146, 39]. Another way to combine algorithms consists of running several different

algorithms independently and aggregating their results: this is called parallel composition [58, 69].

- How to involve the user in the loop. There are at least three areas in which users can be involved in a matching solution: (*i*) obviously, by providing initial alignments (and parameters) to the matchers, (*ii*) by dynamically combining matchers [160, 8, 56], and (*iii*) by providing feedback to the matchers in order for them to adapt their results [67, 25].

- How to extract the final alignment. This can be done by using various thresholds, e.g., hard threshold (retains all the correspondence above a given threshold) [58, 69], or graph matching algorithms [19, 141], more precisely weighted bipartite graph matching or covering [87].

Besides the above mentioned aspects, development of a matching strategy also covers the use of learning and probabilistic algorithms for learning from data the best method and the best parameters for matching [60, 67, 205], dealing with circularities and developing a strategy for computing these similarities in spite of cycles and non linearity in the constraints governing similarities [159, 78].

3.5 Summary

There is a variety of techniques that can be used for ontology matching. The classification discussed in this chapter provides a common conceptual basis, and, hence, can be used for comparing (analytically) different existing ontology matching systems as well as for designing a new one, taking advantages of state of the art solutions. Also, classifications of matching methods provide some guidelines which help in identifying families of matching techniques.

This chapter showed the difficulty of having a clear cut classification of algorithms. We provided two such classifications based on granularity and input interpretation on one side and the kind of input on the other side. We also briefly outlined a number of issues to be addressed when assembling components of a matching system. This indicated that the craft of ontology matching systems is a delicate art of combining basic matchers in the most advantageous way.

Chapter 4

Overview of matching systems

This chapter is devoted to an overview of existing matching systems which have emerged during the last decade. There have already been done some comparisons of a number of matching systems, in particular in [193, 198, 57, 123, 176, 61, 214]. Our purpose here is not to compare them in full detail, although we give some comparisons, but rather to show their variety, in order to demonstrate in how many different ways the methods presented in the previous chapter have been practically exploited. We present the matching systems in light of the classifications of Chapter 3. We also point to concrete basic matchers and matching strategies used in the considered systems.

In order to facilitate the presentation we follow two rules. First, the year of the system appearance is considered. Then, if there are some evolutions of the system or very similar systems, these are discussed close to each other. Since the main focus of this thesis is on schema-based matching, instance-based systems (e.g., LSD [60], GLUE [63], Automatch [21], sPLMap [175]) as well as meta-matching systems (APFEL [67], eTuner [205]) were excluded from the consideration, see [75] for an overview. We have also excluded from consideration the systems which assume that align-

ments have already been established, and use this assumption as a prerequisite of running the actual system. These approaches include such information integration systems as: Tsimmis [44], Observer [162], SIMS [3], Kraft [196], Picsel [105], DWQ [38], AutoMed [35], and InfoMix [136].

Material presented in this chapter has been developed in collaboration with Jérôme Euzenat and published in [214, 75]. Also some work on the topic of this chapter has been supported by the FP6 Knowledge Web[1] Network of Excellence.

The structure of this chapter is as follows. We first describe systems which focus on schema-level information (§4.1). Then, we present systems which exploit both schema-level and instance-level information (§4.2).

4.1 Schema-based systems

Schema-based systems, according to the classification of Chapter 3, are those which rely mostly on schema-level input information for performing ontology matching.

4.1.1 Hovy (University of Southern California)

[119] describes a number of heuristics used to match large-scale ontologies, such as *Sensus* and *Mikrokosmos*, in order to combine them in a single reference ontology. In particular, were used three types of matchers based on (*i*) concept names, (*ii*) concept definitions, and (*iii*) taxonomy structure. For example, the name matcher splits composite-word names into separate words and then compares substrings in order to produce a similarity score. Specifically, the name matcher score is computed as the sum of the square of the number of letters matched, plus 20 points if words are exactly equal or 10 points if end of match coincides. For instance, using this strategy,

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

the comparison between *Free World* and *World* results in 35 points score, while the comparison between *cuisine* and *vine* results in 19 points score. The definition matcher compares the English definitions of two concepts. Here, both definitions are first separated into individual words. Then, the number and the ratio of shared words in two definitions is computed in order determine the similarity between them. Finally, results of all the matchers are combined based on experimentally obtained formula. The combined scores between concepts from two ontologies are sorted in descending order and are presented to the user for establishing a cutoff value as well as for approving or discarding operations, results of which are saved for later reuse.

4.1.2 TransScm (Tel Aviv University)

TransScm [166] provides data translation and conversion mechanisms between input schemas based on schema matching. First, by using rules, the alignment is produced in a semi-automatic way. Then, this alignment is used to translate data instances of the source schema to instances of the target schema. Input schemas are internally encoded as labeled graphs, where some of the nodes may be ordered. Nodes of the graph represent schema elements, while edges stand for the relations between schema elements or their components. Matching is performed between nodes of the graphs top-down and in one-to-one fashion. Matchers are viewed as rules. For example, (according to the *identical* rule) two nodes match if their labels are found to be synonyms based on the built-in thesaurus; see for a list of the available rules [240]. The system combines rules sequentially based on their priorities. It tries to find for the source node a unique *best* matching target node, or determine a mismatch. In case (*i*) there are a number of matching candidates, among which the system cannot choose the best one, or (*ii*) if the system cannot match or mismatch a source node

to a target node with the given set of rules, user involvement is required. In particular, users with the help of a graphic user interface can add, disable or modify rules to obtain the desired matching result. Then, instances of the source schema are translated to instances of the target schema according to the match rules. For the example of the *identical* rule, translation includes copying the source node components.

4.1.3 DIKE (Università di Reggio Calabria, Università di Calabria)

DIKE (Database Intensional Knowledge Extractor) is a system supporting the semi-automatic construction of cooperative information systems (CISs) from heterogeneous databases [189, 187, 188, 186]. It takes as input a set of databases belonging to the CIS. It builds a kind of mediated schema (called data repository or global structured dictionary) in order to provide a user-friendly integrated access to the available data sources. DIKE focuses on entity-relationship schemas. The matching step is called the extraction of inter-schema knowledge. It is performed in a semi-automatic way. Some examples of inter-schema properties that DIKE can find are *terminological properties*, such as synonyms, homonyms among objects, namely entities and relationships, or type conflicts, e.g., similarities between different types of objects, such as entities, attributes, relationships; *structural properties*, such as object inclusion; *subschema similarities*, such as similarities between schema fragments. With each kind of property is associated its plausibility coefficient in the [0 1] range. The properties with a lower plausibility coefficient than a dynamically derived threshold are discarded, whereas others are accepted. DIKE works by computing sequentially the above mentioned properties. For example, synonyms and homonyms are determined based on information from external resources, such as WordNet, and by analyzing the distances of objects in the neigh-

borhood of the objects under consideration. Also, some weights are used to produce a final coefficient. Then, type conflicts are analyzed and resolved by taking as input the results of synonyms and hyponyms analysis.

4.1.4 SKAT and ONION (Stanford University)

SKAT (Semantic Knowledge Articulation Tool) is a rule-based system that semi-automatically discovers mappings between two ontologies [169]. Internally, input ontologies are encoded as graphs. Rules are provided by domain experts and are encoded in first order logic. In particular, experts specify initially desired matches and mismatches. For example, a rule *President* = *Chancellor*, indicates that we want *President* to be an appropriate match to *Chancellor*. Apart from declarative rules, experts can specify matching procedures that can be used to generate the new matches. Thus, experts have to approve or reject the automatically suggested matches, thereby producing the resulting alignment. Matching procedures are applied sequentially. Some examples of these procedures are: string-based matching, e.g., two terms match if they are spelled similarly, and structure matching, e.g., structural graph slices matching, such as considering nodes near the root of the first ontology against nodes near the root of the second ontology.

ONION (ONtology compositION) is a successor system. It discovers mappings between multiple ontologies semi-automatically. The ultimate goal of matching is to enable a unified query answering over the involved ontologies [170]. Input ontologies (the system handles RDF files) are internally represented as labeled graphs. The alignment is viewed as a set of *articulation rules*. The semi-automated algorithm for resolving the terminological heterogeneity of [168] forms the basis of the *articulation generator*, ArtGen, for the ONION system. ArtGen, in turn, can be viewed as an evolution of the SKAT system with some added matchers. Thus, it exe-

cutes a set of matchers and suggests articulation rules to the user. A human expert can either accept, modify or delete the suggestions. The expert can also indicate the new matches that the articulation generator might have missed. ArtGen works sequentially, first by performing linguistic matching and then structure-based matching. During the linguistic matching phase, concept names are represented as sets of words. The linguistic matcher compares all possible pairs of words from any two concepts of both ontologies and assigns a similarity score in [0 1] to each pair. The matcher uses a word similarity table generated by a thesaurus-based or corpus-based matcher called the *word relator* to determine the similarity between pairs of words. The similarity score between two concepts is the average of the similarity scores (ignoring scores of zero) of all possible pairs of words in their names. If this score is higher than a given threshold, ArtGen generates a match candidate. Structure-based matching is performed based on the results of the linguistic matching. It looks for structural isomorphism between subgraphs of the ontologies, taking into account some linguistic clues (see also §4.1.9 for a similar technique). The structural matcher tries to match only the unmatched pairs from the linguistic matching, thereby complementing its results.

4.1.5 Artemis (Università di Milano, Università di Modena e Reggio Emilia)

Artemis (Analysis of Requirements: Tool Environment for Multiple Information Systems) [39] was designed as a module of the MOMIS mediator system [18, 17] for creating global views. It performs affinity-based analysis and hierarchical clustering of source schema elements. Affinity-based analysis represents the matching step: in a sequential manner it calculates the name, structural and global affinity coefficients exploiting a common thesaurus. The common thesaurus is built with the help of ODB-Tools [16],

WordNet or manual input. It represents a set of intensional and a set of extensional relationships which depict intra- and inter-schema knowledge about classes and attributes of the input schemas. Based on global affinity coefficients, a hierarchical clustering technique categorizes classes into groups at different levels of affinity. For each cluster it creates a set of global attributes and the global class. Logical correspondence between the attributes of a global class and source schema attributes is determined through a mapping table.

4.1.6 H-Match (Università degli Studi di Milano)

H-Match [41] is an automated ontology matching system. It was designed to enable knowledge discovery and sharing in the settings of open networked systems, in particular within the Helios peer-to-peer framework [40]. The system handles ontologies specified in OWL. Internally, these are encoded as graphs using the H-model representation [40]. H-Match inputs two ontologies and outputs (one-to-one or one-to-many) correspondences between concepts of these ontologies with the same or closest intended meaning. The approach is based on a similarity analysis through affinity metrics, e.g., term to term affinity, datatype compatibility, and thresholds. H-Match computes two types of affinities (in the [0 1] range), namely *linguistic* and *contextual* affinity. These are then combined by using weighting schemas, thus yielding a final measure, called *semantic* affinity. Linguistic affinity builds on top of a thesaurus-based approach of the Artemis system (§4.1.5). In particular, it extends the Artemis approach (*i*) by building a common thesaurus involving such relations among WordNet synsets as meronymy or coordinate terms, and (*ii*) by providing an automatic handler of compound terms (i.e., those composed by more than one token) that are not available from WordNet. Contextual affinity requires consideration of the neighbor concepts, e.g., linked via taxonomical or mereological rela-

tions, of the actual concept.

One of the major characteristics of H-Match is that it can be dynamically configured for adaptation to a particular matching task. Notice that in dynamic settings complexity of a matching task is not known in advance. This is achieved by means of four matching models. These are: *surface*, *shallow*, *deep*, and *intensive*, each of which involves different types of constructs of the ontology. Computation of a linguistic affinity is a common part of all the matching models. In case of the surface model, linguistic affinity is also the final affinity, since this model considers only names of ontology concepts. All the other three models take into account various contextual features and therefore contribute to the contextual affinity. For example, the shallow model takes into account concept properties, whereas the deep and the intensive models extend previous models by including relations and property values, respectively. Each concept involved in a matching task can be processed according to its own model, independently from the models applied to the other concepts within the same task. Finally, by applying thresholds, correspondences with semantic (final) affinity higher than the cut-off threshold value are returned in the final alignment.

4.1.7 Anchor-Prompt (Stanford Medical Informatics)

Anchor-Prompt [181] is an extension of Prompt, also formerly known as SMART, and is an ontology merging and alignment tool with a sophisticated prompt mechanism for possible matching terms [179]. Prompt handles ontologies expressed in such knowledge representation formalisms as OWL and RDF Schema. Anchor-Prompt is a sequential matching algorithm that takes as input two ontologies, internally represented as graphs and a set of anchors-pairs of related terms, which are identified with the help of string-based techniques, such as edit-distance, or defined by a user

or another matcher computing linguistic similarity. Then the algorithm refines them by analyzing the paths of the input ontologies limited by the anchors in order to determine terms frequently appearing in similar positions on similar paths. Finally, based on the frequencies and user feedback, the algorithm determines matching candidates.

4.1.8 OntoBuilder (Technion Israel Institute of Technology)

OntoBuilder is a system for information seeking on the web [171]. A typical situation the system deals with is, for example, when a user is searching for a car to be rented. Obviously, the user would like to compare prices from multiple providers in order to make an informed decision. Thus, the same input information has to be typed in many times. OntoBuilder operates in two phases, namely: (*i*) ontology creation (the so called *training* phase) and (*ii*) ontology adaptation (the so called *adaptation* phase). During the training phase an initial ontology (in which a user's data needs are encoded) is created by extracting it from a visited web-site of, e.g., *AVIS* car rental company. The adaptation phase includes on-the-fly matching and interactive merging operations of the related ontologies with the actual (initial) ontology. Ontology creation is out of the scope of this thesis. Hence, we concentrate only on the ontology adaptation phase. During the adaptation phase the user suggests the web sites (s)he would like to further explore, e.g., the *Hertz* car rental company. Each such a site goes through the ontology extraction process, thus, resulting in a candidate ontology, which is then merged into the actual ontology. To support this, the best match for each existing term in the actual ontology (to terms from the candidate ontology) is selected. Selection strategy employs thresholds. The matching algorithm works in a term to term fashion. It sequentially executes a number of matchers. Some examples of the matchers used here are removing noisy characters and stop terms and substring matching. If

all else fails, a thesaurus look-up is performed. Finally, mismatched terms are presented to the user for manual matching. Some further matchers such as those for precedence matching were introduced in a later work in [86]. Also top-k mappings as alternative for single best matching (i.e., top-1 category) was proposed in [85].

4.1.9 Cupid (University of Washington, Microsoft Corporation, University of Leipzig)

Cupid [146] implements a sequential algorithm comprising linguistic and structural schema matching techniques, and computing similarity coefficients with the assistance of domain specific thesauri. Input schemas are encoded as graphs. Nodes represent schema elements and are traversed in a combined bottom-up and top-down manner. The matching algorithm consists of three phases and operates only with tree-structures, to which non-tree cases are reduced. The first phase (linguistic matching) computes linguistic similarity coefficients between schema element names (labels) based on morphological normalisation, categorization, string-based techniques, such as common prefix, suffix tests, and thesauri look-up. The second phase (structural matching) computes structural similarity coefficients weighted by leaves which measure the similarity between contexts in which elementary schema elements occur. The third phase (mapping elements generation) aggregates the results of the linguistic and structural matching through a weighted sum and generates a final alignment by choosing pairs of schema elements with weighted similarity coefficients which are higher than a threshold.

4.1.10 COMA and COMA++ (University of Leipzig)

COMA (COmbination of MAtching algorithms) [58] is a schema matching tool based on parallel composition of matchers. It provides an extensible library of matching algorithms, a framework for combining obtained results, and a platform for the evaluation of the effectiveness of the different matchers. As in [58], COMA contains six elementary matchers, five hybrid (i.e., combinations of elementary methods) matchers, and one reuse-oriented matcher. Most of them implement string-based techniques, such as affix, n-gram, edit distance; others share techniques with Cupid (thesauri look-up, etc.). Reuse-oriented is an original matcher, which tries to reuse previously obtained results for entire new schemas or for its fragments. Schemas are internally encoded as directed acyclic graphs, where elements are the paths. This aims at capturing contexts in which the elements occur. Distinct features of the COMA tool in respect to Cupid are a more flexible architecture and the possibility of performing iterations in the matching process. It presumes interaction with users who approve obtained matches and mismatches to gradually refine and improve the accuracy of match. COMA++ is built on top of COMA by elaborating in more detail the alignment reuse operation, provides a more efficient implementation of the COMA algorithms and a graphical user interface [58, 56].

4.1.11 Similarity Flooding (Stanford University, University of Leipzig)

The Similarity Flooding (SF) [159] approach is based on the ideas of similarity propagation. Schemas are presented as directed labeled graphs; grounding on the OIM specification [156]. The algorithm manipulates them in an iterative fix-point computation to produce an alignment between the nodes of the input graphs. The technique starts from string-based compar-

ison, such as common prefix, suffix tests, of the vertices labels to obtain an initial alignment which is refined within the fix-point computation. The basic concept behind the similarity flooding algorithm is the similarity spreading from similar nodes to the adjacent neighbors through propagation coefficients. From iteration to iteration the spreading depth and a similarity measure are increasing till the fix-point is reached. The result of this step is a refined alignment which is further filtered to finalize the matching process.

4.1.12 CtxMatch and CtxMatch2 (University of Trento, ITC-IRST)

CtxMatch [31, 30, 33] represents the first instantiation of the semantic matching approach [97], namely the approach developed in this thesis. It translates the ontology matching problem into the logical validity problem and computes logical relations, such as equivalence, subsumption between concepts and properties. CtxMatch is a sequential system. At the element level it uses only WordNet to find initial matches for classes as well as for properties. At the structure level, it exploits description logic reasoners, such as Pellet[2] or FaCT[3] to compute the final alignment in a way similar to what is presented in Chapter 3 when discussing methods based on description logics.

4.1.13 DCM framework (University of Illinois at Urbana-Champaign)

MetaQuerier [43] is a middleware system that assists users in finding and quering multiple databases on the web. It exploits the Dual Correlation Mining (DCM) matching framework to facilitate source selection according

[2] http://www.mindswap.org/2003/pellet/

[3] http://www.cs.man.ac.uk/∼horrocks/FaCT

to user search keywords [116]. Unlike other works, the given approach takes as input multiple schemas and returns alignments between all of them. This setting is called *holistic* schema matching. DCM automatically discovers complex mappings, e.g., {*author*} corresponds to {*first name, last name*}, between attributes of the web query interfaces in the same domain of interest, e.g., books. As the name (DCM) indicates, schema matching is viewed as *correlation mining*. The idea is that co-occurrence patterns often suggest complex matches. That is, *grouping attributes*, such as *first name* and *last name*, tend to co-occur in query interfaces. Technically, this means that those attributes are positively correlated. Contrary, attribute names which are synonyms, e.g., *quantity* and *amount*, rarely co-occur, thus representing an example of negative correlation between them. Matching is performed in two phases. During the first phase (matching discovery), a set of matching candidates is generated by mining first positive and then negative correlations among attributes and attribute groups. Also, some thresholds and a specific correlation measure such as the H-measure are used. During the second phase (matching construction), by applying some strategies of ranking, e.g., scoring function, rules, and selection, such as iterative greedy selection, the final alignment is produced.

4.2 Mixed systems

The following systems take advantage of both schema-level and instance-level input information if they are both available.

4.2.1 SEMINT (Northwestern University, NEC, The MITRE Corporation)

SEMantic INTegrator (SEMINT) is a tool based on neural networks to assist in identifying attribute correspondences in heterogeneous databases

[137, 138]. It supports access to a variety of database systems and utilizes both schema- and instance-level information to produce rules for matching corresponding attributes automatically. The approach works as follows. First, it extracts from two databases all the necessary information (features or discriminators) which is potentially available and useful for matching. This includes normalized schema information, e.g., field specifications, such as datatypes, length, constraints, and statistics about data values, e.g., character patterns, such as ratio of numerical characters, ratio of white spaces, and numerical patterns, such as mean, variance, standard deviation. Second, by using a neural network as a classifier (self-organizing map algorithm), it groups the attributes based on similarity of the features for a single (the first) database. Then, it uses a back-propagation neural network for learning and recognition. Based on the previously obtained clusters, the learning is performed. Finally, using a trained neural network on the first database features and clusters, the system recognizes and computes similarities between the categories of attributes from the first database and the features of attributes from the second database, thus, generating a list of match candidates, which are to be inspected and confirmed or discarded by a human user.

4.2.2 Clio (IBM Almaden, University of Toronto)

Clio is a system for managing and facilitating data transformation and integration tasks within heterogeneous environments [164, 165, 172, 113]. Clio handles relational and XML schemas. As a first step, the system transforms the input schemas into an internal representation, which is a nested relational model. The Clio approach is focused on making the alignment operational. It is assumed that the matching step (namely, identification of the so-called *value correspondences*) is performed with the help of a schema matching component or manually by the user. The built-in schema match-

ing algorithm of Clio combines in a sequential manner instance-based attribute classification via a variation of a Naive Bayes classifier [107, 64, 149] and string matching between elements names, e.g., by using edit distance measure. Then, taking the n:m value correspondences (the alignment) together with constraints coming from the input schemas, Clio compiles these into an internal query graph representation. In particular, an interpretation of the input correspondences is given. Thus, a set of logical mappings with formal semantics is produced. To this end, Clio also supports mappings composition [79]. Finally, the query graph can be serialized into different query languages, e.g., SQL, XSLT, XQuery, thus enabling actual data to be moved from a source to a target, or to answer queries. The system, besides trivial transformations, aims at discovering complex ones, such as the generation of keys, references, join conditions.

4.2.3 NOM and QOM (University of Karlsruhe)

NOM (Naive Ontology Mapping) [69] and QOM (Quick Ontology Mapping) [68] are components of the FOAM framework [66].

NOM adopts the idea of parallel composition of matchers from COMA (§4.1.10). Some innovations with respect to COMA are in the set of elementary matchers based on rules, exploiting explicitly codified knowledge in ontologies, such as information about super- and sub-concepts, super- and sub-properties, etc. At present the system supports 17 rules. For example, a rule states that if super-concepts are the same, the actual concepts are similar to each other. These rules use many terminological and structural techniques.

QOM (Quick Ontology Mapping) [68] is a variation of the NOM system dedicated to improve the efficiency of the system. The approach is based on the idea that the loss of quality in matching algorithms is marginal (to a standard baseline); however improvement in efficiency can be tremendous.

This fact allows QOM to produce correspondences fast, even for large-size ontologies. QOM is grounded on matching rules of NOM. However, for the purpose of efficiency the use of some rules, e.g., the rules that traverse the taxonomy, have been restricted. QOM avoids the complete pairwise comparison of trees in favor of a (n incomplete) top-down strategy, thereby focusing only on promising matching candidates. The similarity measures produced by basic matchers (matching rules) are refined by using a sigmoïd function, thereby emphasizing high individual similarities and de-emphasizing low individual similarities. They are then aggregated through weighted average. Finally, with the help of thresholds, the final alignment is produced.

4.2.4 OLA (INRIA Rhône-Alpes, Université de Montréal)

OLA (OWL Lite Aligner) [78] is an ontology matching system which is designed with the idea of balancing the contribution of each of the components that compose an ontology, e.g., classes, constraints, data instances. OLA handles ontologies in OWL. It first compiles the input ontologies into graph structures, unveiling all relationships between entities. These graph structures produce the constraints for expressing a similarity between the elements of the ontologies. The similarity between nodes of the graphs follows two principles: (i) it depends on the category of node considered, e.g., class, property, and (ii) it takes into account all the features of this category, e.g., superclasses, properties.

The distance between nodes in the graph are expressed as a system of equations based on string-based, language-based and structure-based similarities. These distances are almost linearly aggregated (they are linearly aggregated modulo local matches of entities). For computing these distances, the algorithm starts with base distance measures computed from labels and concrete datatypes. Then, it iterates a fix-point algorithm until

no improvement is produced. From that solution, an alignment is generated which satisfies some additional criterion on the alignment obtained and the distance between matched entities.

4.2.5 Corpus-based matching (University of Washington, Microsoft Research, University of Illinois at Urbana-Champaign)

[144] proposed an approach to schema matching which, besides input information available from schemas under consideration, also exploits some domain specific knowledge via an external corpus of schemas and mappings. The intuition behind the approach is based on the use of corpus in information retrieval, where similarity between queries and concepts is determined based on analyzing large corpora of text. In schema matching settings, such a corpus can be initialized with a small number of schemas obtained, for example, by using available standard schemas in the domain of interest (see, for instance, XML.org[4] and OASIS.org[5]) and should eventually evolve in time with new matching tasks.

Since the corpus is intended to have a number of different representations of each concept in the domain, it should facilitate learning these variations in the elements and their properties. The corpus is exploited in two ways. First, to obtain an additional evidence about each element being matched by including evidence from similar elements in the corpus. Second, in the corpus, similar elements are clustered and some statistics for clusters are computed, such as neighborhood and ordering of elements. These statistics are ultimately used to build constraints that facilitate selection of the correspondences in the resulting alignment.

The approach handles web forms and relational schemas and focuses on

[4] http://www.xml.org/

[5] http://www.oasis-open.org/

one-to-one alignments. It works in two logical phases. Firstly, schemas under consideration are matched against the corpus, thereby augmenting these with possible variations of their elements based on knowledge available from the corpus. Secondly, augmented schemas are matched against each other. In both cases the same set of matchers is applied. In particular, basic matchers, called learners, include: (*i*) a *name learner*, (*ii*) a *text learner*, (*iii*) a *data instance learner*, and (*iv*) a *context learner*. These matchers mostly follow the ideas of techniques used in LSD [59] and Cupid (§4.1.9). For example, the *name learner* exploits names of elements. It applies tokenization and n-grams to the names in order to create training examples. The matcher itself is a text classifier, such as Naive Bayes. In addition, the name learner, in order to determine similarity between element names string, uses edit distance. The *data instance learner* determines whether the values of instances share common patterns, same words, etc. Also, a matcher for an automatic combination of the results produced by basic matchers, called *metalearner*, uses *logistic regression* with the help of *stacking* technique [224] in order to learn its parameters. Finally, by using constraints obtained based on the statistics from the corpus, some filtering of the candidate correspondences is performed in order to produce the final alignment.

4.3 Summary

The panorama of systems considered in this chapter has multiplied the diversity of basic techniques by the variety of strategies for combining them introduced in the previous chapter. However, there are a number of constant features that are shared by the majority of systems. Also, usually each individual system innovates on a particular aspect. Let us summarize some global observations concerning the presented systems:

- Most of the systems under consideration deal with particular ontology types, such as DTDs, relational schemas and OWL ontologies. Only a small number of systems aim at being generic, i.e., handle multiple types of ontologies. Some examples include Cupid (§4.1.9), COMA and COMA++ (§4.1.10), Similarity Flooding (§4.1.11) and the approach proposed in this thesis.

- Most of the approaches take as input a pair of ontologies, including the approach proposed in this thesis, while only a small number of systems take as input multiple ontologies, e.g., DCM (§4.1.13).

- Most of the approaches handle only tree-like structures, including the approach proposed in this thesis, while only a small number of systems handle graphs. Some examples of the latter include Cupid (§4.1.9), COMA and COMA++ (§4.1.10), and OLA (§4.2.4).

- Most of the systems focus on discovery of one-to-one alignments, while only a small number of systems have tried to address the problem of discovering more complex correspondences, such as one-to-many, e.g., the approach proposed in this thesis, and many-to-many, e.g., DCM (§4.1.13).

- Most of the systems focus on computing confidence measures in [0 1] range, most often standing for the fact that the equivalence relation holds between ontology entities. Only a small number of systems compute logical relations between ontology entities, such as equivalence, subsumption. Some examples of the latter include CtxMatch (§4.1.12) and the approach proposed in this thesis.

Table 4.1 summarizes how some of the matching systems considered in this chapter cover the solution space in terms of the classification of Chapter 3.

Table 4.1: Basic matchers used by different systems

	Element-level		Structure-level	
	Syntactic	**External**	**Syntactic**	**Semantic**
Hovy §4.1.1	string-based, language-based	-	taxonomic structure	-
TranScm §4.1.2	string-based	built-in thesaurus	taxonomic structure, matching of neighbourhood	-
DIKE §4.1.3	string-based, domain compatibility	WordNet	matching of neighbourhood	-
SKAT §4.1.4	string-based	auxiliary thesaurus, corpus-based	taxonomic structure, matching of neighbourhood	-
Artemis §4.1.5	domain compatibility, language-based	common thesaurus (CT)	matching of neighbours via CT, clustering	-
H-Match §4.1.6	domain compatibility, language-based, domains and ranges	common thesaurus (CT)	matching of neighbours via CT, relations	-
Anchor-Prompt §4.1.7	string-based, domains and ranges	-	bounded paths matching: (arbitrary links), taxonomic structure	-
OntoBuilder §4.1.8	string-based, language-based	thesaurus look up	-	-
Cupid §4.1.9	string-based, language-based, datatypes, key properties	auxiliary thesauri	tree matching weighted by leaves	-
COMA & COMA++ §4.1.10	string-based, language-based, datatypes	auxiliary thesauri, alignment reuse, repository of structures	DAG (tree) matching with a bias towards various structures (e.g., leaves)	-
SF §4.1.11	string-based, datatypes, key properties	-	iterative fix-point computation	-
CtxMatch §4.1.12	string-based, language-based	WordNet	-	based on description logics
DCM §4.1.13	-	-	correlation mining	-
SEMINT §4.2.1	neural network, datatypes, value patterns	-	-	-
Clio §4.2.2	string-based, language-based, Naive Bayes	-	-	-
NOM & QOM §4.2.3	string-based, domains and ranges	application-specific vocabulary	matching of neighbours, taxonomic structure	-
OLA §4.2.4	string-based, language-based, datatypes	WordNet	iterative fix-point computation, matching of neighbours, taxonomic structure	-

For example as from Table 4.1, OLA (§4.2.4) exploits string-based element-level matchers, a matcher based on WordNet, iterative fix-point computation, etc. Table 4.1 also testifies that ontology matching research so far was mainly focused on syntactic and external techniques. In fact, many systems rely on the same string-based techniques. Similar observation can be also made concerning the use of WordNet as an external resource of common knowledge. In turn, semantic techniques have rarely been exploited, this is only done by the approach proposed in this thesis and CtxMatch (§4.1.12).

Having considered some of the recent schema-based matching systems, it is important to notice that the matching operation typically constitutes only one of the steps towards the ultimate goal of, e.g., ontology integration, data integration, and web service composition. To this end, we would like to mention some existing infrastructures, which use matching as one of its integral components. Some examples include: Chimaera [150], MAFRA [49, 142], Rondo and Moda [161, 160, 158], Prompt Suite [180, 177], Alignment API [73], GeRoMe [126], Protoplasm [24], COMA++ [58, 56] and ModelGen [6, 7]. The goal of such infrastructures is to enable a user with a possibility of performing such high-level tasks, e.g., given a product request expressed in terms of the catalog $C1$, return the products satisfying the request from the marketplaces $MP1$ and $MP2$. Moreover, use matching component $M5$, and translate instances by using component $T3$.

Part III

Schema-based semantic matching

Chapter 5

Semantic matching

We think of *matching* as an operation that takes two graph-like structures (e.g., classifications, XML schemas or ontologies) and produces an alignment between nodes of two graphs that correspond semantically to each other.

Many various solutions of matching have been proposed so far. This work concentrates on a schema-based solution, namely a matching approach exploiting only the schema information, thus not considering instances. The reason behind our choice is that schema information is always available, while this is not the case with instance information: (*i*) in traditional applications, such as schema integration, instance data may not be available due to the security concerns [46], (*ii*) in some of the emerging applications, such as two agents meeting or looking for the web service integration, there are no instances given beforehand. Finally in some applications, for example, dealing with masterpieces [229], instances are the image data, which will require a specific solution. A schema-based solution, in principle, can be used in the above mentioned cases. Therefore, schema-based solutions potentially have a wider applicability rather than instance-based solutions.

In this thesis we propose the so-called *semantic matching* schema-based approach. This approach is based on two key ideas. The first is that correspondences are calculated between entities of ontologies by computing logical relations (e.g., equivalence, subsumption, disjointness), instead of computing coefficients rating match quality in the [0 1] range, as it is the case in many other approaches [146, 58, 159, 78]. The second idea is that the relations are determined by analyzing the meaning which is codified in the entities and the structures of ontologies. In particular, labels at nodes, written in natural language, are automatically translated into propositional formulas which explicitly codify the labels' intended meaning. This allows the translation of the matching problem into a propositional validity problem, which can then be efficiently resolved using sound and complete propositional satisfiability deciders.

Material presented in this chapter has been developed in collaboration with Mikalai Yatskevich and published in [96, 97, 98]. Algorithms presented in this chapter have been implemented by Mikalai Yatskevich within the S-Match system. Therefore, implementation details are out of scope of this chapter, see [103] for details.

In this chapter we first present basic motivations behind the proposed approach (§5.1). Then, we discuss the semantic matching by intuitions and examples as well as we state the problem technically (§5.2). Finally, we provide the main macro steps of the algorithm realizing the semantic matching approach (§5.3).

5.1 Generic and general matching

We assume that all forms of ontologies, e.g., database schemas, classifications, and formal ontologies (§2.1), can be represented as graphs. There-

fore, the matching problem can be decomposed into two steps:

- extract graphs from the input ontologies,
- match the resulting graphs [1].

This allows for the statement and solution of a generic matching problem, very much along the lines of what is done in Cupid [146] (§4.1.9) and COMA [58] (§4.1.10). From a technical perspective, development of a generic matcher aims at handling different forms of ontologies, e.g., relational schemas, XML schemas, classifications and OWL ontologies (§2.1), and being general-purpose means being able to serve for many applications, e.g., ontology integration (Chapter 1). These are the motivations behind the unified treatment of matching that we take in our approach and the position of considering matching being a separate operation, as opposed to considering merging or mediating being the primitive ones.

Let us define the notion of matching graphs more precisely.

A mapping element is a 4-tuple $\langle ID_{ij}, n1_i, n2_j, R\rangle$, *i=1,...,N1; j=1,...,N2; where* ID_{ij} *is a unique identifier of the given mapping element;* $n1_i$ *is the i-th node of the first graph, N1 is the number of nodes in the first graph;* $n2_j$ *is the j-th node of the second graph, N2 is the number of nodes in the second graph; and R specifies a similarity relation which may hold between the nodes* $n1_i$ *and* $n2_j$.

Note that the definition of mapping element above is a simplified version of the correspondence (§2.3, p.29).

Matching is the process of discovering mapping elements between two graphs through the application of a matching algorithm.

There exist two approaches to graph matching, namely *exact* matching and *inexact* or *approximate* matching, see, for instance [211]. Both of

[1] Note that this step is different from what is called graph matching in the graph theory [19, 141], although may include it as an integral component (§3.4).

them can be stated as subgraph matching problems: find all occurrences of a pattern graph P of m nodes as a subgraph of a graph G of n nodes, $m \leq n$. In the case of exact matching we look for subgraphs S of G that are identical to P. In inexact matching some errors are acceptable. For obvious reasons we are interested in inexact matching.

5.2 Semantic matching: the idea

As the name of the approach indicates, in semantic matching the key intuition is to match meanings (concepts). Thus, in order to emphasize this choice (and, hence, be more specific than in §5.1), mapping elements are computed as 4-tuples $\langle ID_{ij}, C1_i, C2_j, R\rangle$, where $C1_i$ is the *concept* at the i-th node of the first graph; $C2_j$ is the *concept* at the j-th node of the second graph; and R specifies a similarity relation in the form of a *semantic relation* between the *concepts* at the given nodes. Possible R's between concepts at nodes are equivalence ($=$), more specific/general ($\sqsubseteq$, $\sqsupseteq$), and disjointness ($\perp$).

The other approaches which have been grouped under the heading of *syntactic matching* in §3.3 (p.58) often focus on matching labels of nodes (being more important than other available information, such as datatypes and cardinalities) and look for the similarity using syntax driven techniques. Thus, in contrast to semantic matching, in the case of syntactic matching, mapping elements can be viewed as 4-tuples $\langle ID_{ij}, L1_i, L2_j, R\rangle$, where $L1_i$ is the *label* at the i-th node of the first graph; $L2_j$ is the *label* at the j-th node of the second graph; and R specifies a similarity relation in the form of a *coefficient*, which measures the similarity between the *labels* of the given nodes. Typical examples of R are confidence measures, for instance, similarity coefficients in the [0 1] range [146].

In semantic matching, when we match two nodes, the concepts we analyze depend not only on the concept attached to the node (the concept denoted by the label of the node), but also on the position of the node in the graph. Thus, we analyze the meaning which is codified in the entities and the structures of ontologies.

Then, the key idea is that labels, which are written in natural language, should be translated into an internal language, the language used to express concepts. The internal language should have precisely defined syntax and semantics, thus avoiding all the problems related to the ambiguities of natural language. In particular, we have chosen as an internal language a *propositional* concept language, whose expressivity turns out to be good enough, i.e., to have no or little loss in meaning, when encoding natural language labels used in classifications and schemas [147, 95]. Finally, this allows the translation of the matching problem into a propositional validity problem, which can then be efficiently resolved using *sound* and *complete* state of the art propositional satisfiability deciders. The advantage of using SAT deciders is that they allow for an exhaustive check of all the possible mapping elements and choosing only the correct ones.

5.2.1 Concept of a label and Concept at a node

In order to introduce two important notions behind the approach, let us consider two classifications of Figure 5.1.

The trivial but key observation is that labels in classifications are used to define the set of documents one would like to classify under the node holding the label. Thus, when we write *Images* (see the root node of O1 in Figure 5.1), we do not really mean "images", but rather "the documents which are (about) images". Analogously, when we write *Europe* (see the root node of O2 in Figure 5.1), we mean "the documents which are about Europe". In other words, a label has an intended meaning, which is what

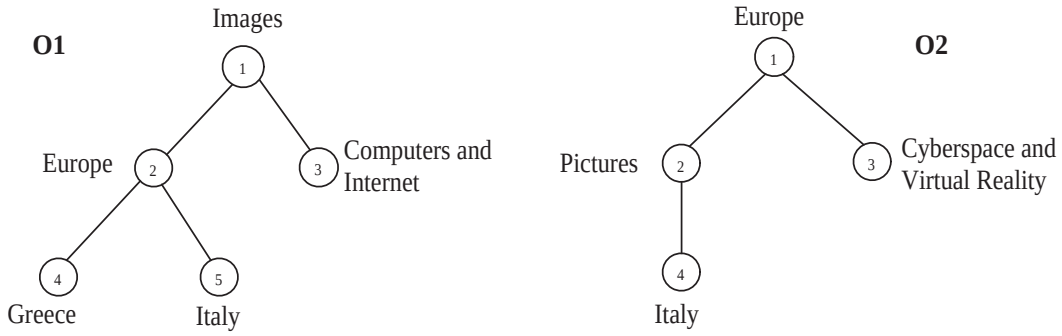


Figure 5.1: Simple catalog matching problem

this label means in the world. However, when using labels for classification purposes, we use them to denote the set of documents which talk about their intended meaning. This consideration allows us to generalize the example definitions of *Images* and *Europe* and to define the concept of a label.

Concept of a label denotes the set of documents that are about what the label means in the world.

Two observations. First, while the semantics of a label are the real world semantics, the semantics of the concept of a label are in the space of documents; the relation being that the documents in the extension of the concept of a label are about what the label means in the real world. Second, concepts of labels depend only on the labels themselves and are independent of where in a graph they are positioned.

Graphs (trees) add structure which allows us to perform the classification of documents more effectively. Let us consider, for instance, the node with label *Europe* in O1. This node stands below the node with label *Images* and, therefore, following what is standard practice in classification, one would classify under this node the set of documents which are images and which are about Europe. Thus, generalizing to graphs (trees) and nodes the idea that the extensions of concepts range in the space of

documents, we can define the concept at a node.

Concept at a node denotes the set of documents that we would classify under this node, given it has a certain label and it is positioned in a certain place in the graph.

More precisely, as the above example has suggested, a document, to be classified in a node, must be in the extension of the concepts of the labels that contribute to its meaning, e.g., all the nodes above it, and of the node itself. Notice that this captures exactly our intuitions about how to classify and access documents within classifications.

Let us now consider some general examples, which make the consequences of the observations described above clearer. For any example we also report the results produced by the state of the art matcher, Cupid [146] (§4.1.9), which exploits sophisticated syntactic matching techniques.

Let us introduce some notation (see Figure 5.1). Numbers are the unique identifiers of nodes. We use "C" for concepts of labels and concepts at nodes. Also we use "$C1$" and "$C2$" to distinguish between concepts of labels and concepts at nodes in graph 1 and graph 2, respectively. Thus, in $\mathsf{O1}$, $C1_{Italy}$ and $C1_5$ are, respectively, the concept of the label *Italy* and the concept at node 5. Also, to simplify the presentation, whenever it is clear from the context we assume that the concept of a label can be represented by the label itself. In this case, for example, C_{Italy} becomes denoted as *Italy*. We sometimes use subscripts to distinguish between graphs in which the given concept of a label occurs. For instance, $Italy_1$, means that the concept of the label *Italy* belongs to the graph $\mathsf{O1}$.

Analysis of siblings. Let us consider Figure 5.2. Structurally the graphs shown in Figure 5.2 differ in the order of siblings. Suppose that we want to match node 5 in $\mathsf{O1}$ with node 2 in $\mathsf{O2}$.

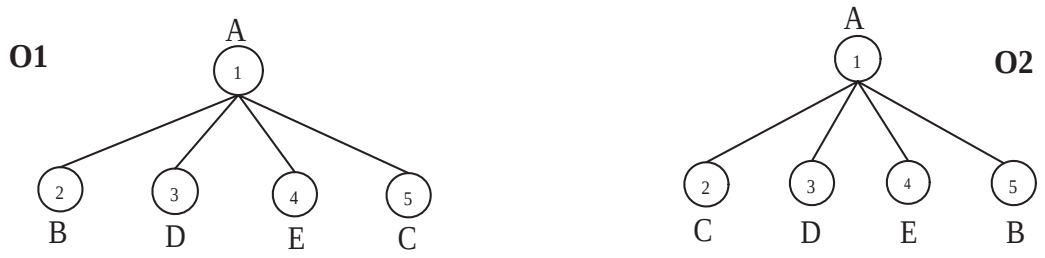


Figure 5.2: Analysis of siblings

Cupid correctly processes this situation, and as a result, the similarity coefficient between labels at the given nodes equals to 0.8, thereby indicating for an appropriate match. This is because $A_1 = A_2$, $C_1 = C_2$ and we have the same structures on both sides. A semantic matching approach compares concepts $A \sqcap C$ in O1 with $A \sqcap C$ in O2 and produces $C1_5 = C2_2$.

Analysis of ancestors. Let us consider Figure 5.3. Suppose that we want to match nodes 5 from O1 and 1 from O2.

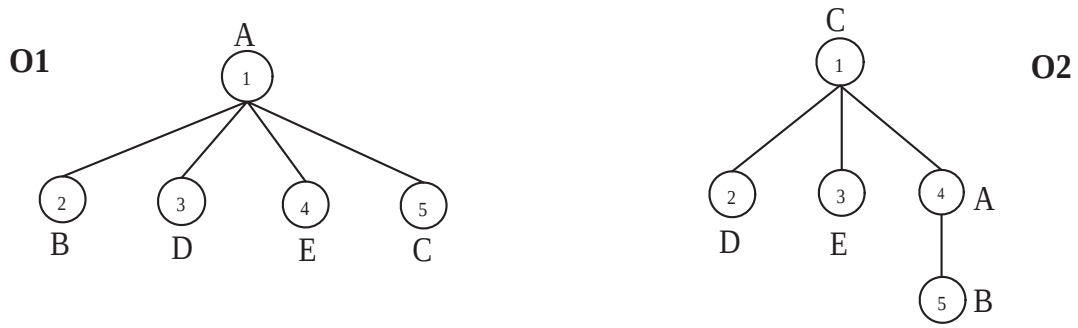


Figure 5.3: Analysis of ancestors. Case 1

Cupid does not find a correspondence between the nodes under consideration, due to the differences in structure of the given graphs, namely matching a leaf node with a root node. In semantic matching, the concept of label of node 5 in O1 is $C1_C$, while the concept at node 5 in O1 is $C1_5 = C1_A \sqcap C1_C$. The concept at node 1 in O2 is $C2_1 = C2_C$. By comparing the concepts of labels at nodes 5 in O1 and 1 in O2 we have

that, being identical, they denote the same concept, namely $C1_C = C2_C$. Thus, the concept at node 5 in O1 is a subset of the concept at node 1 in O2, namely $C1_5 \sqsubseteq C2_1$.

Let us complicate the example shown in Figure 5.3 by allowing for an arbitrary number of nodes between ancestors, see Figure 5.4. The asterisk means that an arbitrary number of nodes are allowed between nodes 1 and 5 in O2. Suppose that we want to match nodes 5 in O1 and 5 in O2.

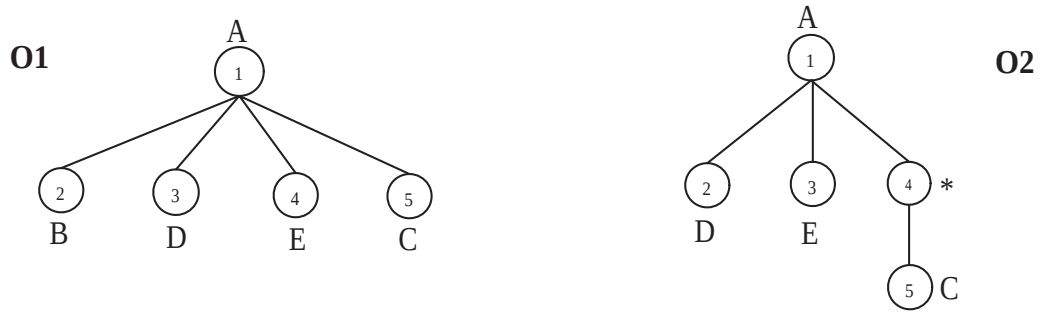


Figure 5.4: Analysis of ancestors. Case 2

Cupid finds out that the similarity coefficient between labels C_1 and C_2 is 0.86, thereby indicating for an appropriate match. This is because of the identity of labels ($A_1 = A_2$, $C_1 = C_2$), and due to the fact that nodes 5 in O1 and 5 in O2 are leaves. Notice how Cupid treats very differently the two situations represented here (Figure 5.4) and in the previous example (Figure 5.3), even if, from a semantic point of view, they are similar. Following semantic matching, the concept at node 5 in O1 is $C1_5 = C1_A \sqcap C1_C$, while the concept at node 5 in O2 is $C2_5 = C2_A \sqcap * \sqcap C2_C$. Since we have that $C1_A = C2_A$ and $C1_C = C2_C$, then $C2_5 \sqsubseteq C1_5$.

5.2.2 Semantic matching: problem statement

Having introduced the basic notions and motivations we proceed with the definition of the semantic matching problem.

A mapping element is a 4-tuple $\langle ID_{ij}, n1_i, n2_j, R\rangle$, *i=1,...,N1; j=1,...,N2; where* ID_{ij} *is a unique identifier of the given mapping element;* $n1_i$ *is the i-th node of the first graph, N1 is the number of nodes in the first graph;* $n2_j$ *is the j-th node of the second graph, N2 is the number of nodes in the second graph; and R specifies a semantic relation which may hold between the concepts at nodes* $n1_i$ *and* $n2_j$. *Possible semantic relations include: equivalence* (=), *more general* ($\sqsupseteq$), *less general* ($\sqsubseteq$), *and disjointness* ($\perp$).

Thus, for instance, the concepts at two nodes are equivalent if they have the same extension, they mismatch if their extensions are disjoint, and so on for the other relations.

We order these relations as they have been listed, according to their binding strength, from the strongest to the weakest, with less general and more general having the same binding power. Thus, equivalence is the strongest binding relation since the mapping element tells us that the concept at the second node has exactly the same extension as the first, more general and less general relations give us a containment information with respect to the extension of the concept at the first node, disjointness provides a containment information with respect to the extension of the complement of the concept at the first node.

When none of the relations holds, the special "idk" (I do not know) relation should be returned. This is an explicit statement that we are unable to compute any of the declared (four) relations. This should be interpreted as either there is not enough background knowledge, and therefore, we cannot explicitly compute any of the declared relations or, indeed, none of those

relations hold according to an application.

Semantic matching can then be defined as the following problem.

Given two graphs G1 and G2 compute the N1 × N2 mapping elements $\langle ID_{i,j}, n1_i, n2_j, R' \rangle$, *with* $n1_i \in$ *G1,* i*=1,...,N1,* $n2_j \in$ *G2,* j*=1,...,N2 and* R' *the strongest semantic relation holding between the concepts at nodes* $n1_i$ *and* $n2_j$.

Since we look for the $N1 \times N2$ correspondences, the cardinality of mapping elements we are able to determine is $1 : m$. Also, these, if necessary, can be decomposed straightforwardly into mapping elements with the 1:1 cardinality.

Thus, for example, considering the concepts at the two root nodes of O1 and O2 in Figure 5.1 we have the following mapping element:

$$\langle ID_{1,1}, n1_1, n2_1, idk \rangle.$$

This is an obvious consequence of the fact that the set of images has a non empty intersection with the set of documents which are about Europe and no stronger relation exists. Building a similar argument for node 2 in O1 and node 2 in O2 of Figure 5.1, and supposing that the concepts of the labels *Images* and *Pictures* are synonyms, we compute instead

$$\langle ID_{2,2}, n1_2, n2_2, = \rangle.$$

Finally, considering also the node 2 in O1 and the nodes with labels *Europe* and *Italy* in O2 of Figure 5.1, we have the following mapping elements:

$$\langle ID_{2,1}, n1_2, n2_1, \sqsubseteq \rangle,$$
$$\langle ID_{2,4}, n1_2, n2_4, \sqsupseteq \rangle.$$

5.3 Semantic matching: the algorithm

We focus on tree-like structures, e.g., classifications, and XML schemas. real world schemas are seldom trees, however, there are (optimized) techniques, transforming a graph representation of a schema into a tree representation, e.g., the graph-to-tree operator of Protoplasm [24]. From now on we assume that a graph-to-tree transformation can be done by using existing systems, and therefore, we focus on other issues instead.

5.3.1 An overview of the algorithm

We consider two simple XML schemas shown in Figure 5.5. Notation follows the one introduced in §5.2.1 (p.91).

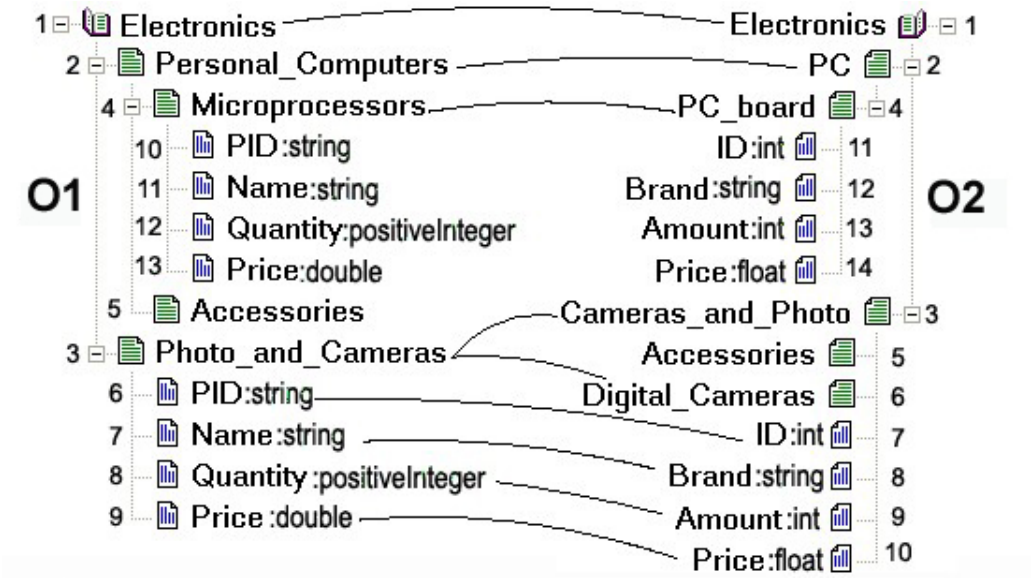


Figure 5.5: Two XML schemas and some of the mapping elements

The algorithm takes as input two tree-like structures and computes as output a set of mapping elements in four macro steps:

Step 1: for all labels L in two trees, compute concepts of labels, C_L.

Step 2: for all nodes N in two trees, compute concepts at nodes, C_N.

Step 3: for all pairs of labels in two trees, compute relations among C_L's.

Step 4: for all pairs of nodes in two trees, compute relations among C_N's.

The first two steps represent the pre-processing phase, while the third and the fourth steps are the element level and structure level matching, respectively (Chapter 3). It is important to notice that Step 1 and Step 2 can be done once, independently of the specific matching problem. Step 3 and Step 4 can only be done at run time, once two trees which must be matched have been chosen.

During Step 1 we compute the meaning of a label at a node (in isolation) by taking as input a label, by analyzing its real world semantics (e.g., using WordNet [163]), and by returning as output a concept of the label. Thus, for example, by writing $C_{Cameras\_and\_Photo}$ we move from the natural language ambiguous label *Cameras_and_Photo* to the concept $C_{Cameras\_and\_Photo}$, which codifies explicitly its intended meaning, namely the data which is about cameras and photo.

During Step 2 we analyze the meaning of the positions that the labels of nodes have in a tree. By doing this we extend concepts of labels to concepts at nodes. This is required to capture the knowledge residing in the structure of a tree, namely the context in which the given concept of label occurs [94]. For example, in O2, when we write C_6 we mean the concept describing all the data instances of the electronic photography products which are digital cameras.

Step 3 is concerned with acquisition of "world" knowledge. Relations between concepts of labels are computed with the help of element level semantic matchers. These matchers take as input two concepts of labels and produce as output a semantic relation (e.g., equivalence, more/less general) between them. For example, from WordNet [163] we can derive that PC and personal computer are synonyms, and therefore, $Personal\_Computer_1 = PC_2$.

Step 4 is concerned with the computation of the relations between concepts at nodes. This problem cannot be resolved by exploiting static knowledge sources only. We have (from Step 3) background knowledge, codified as a set of relations between concepts of labels occurring in two trees. This knowledge constitutes the background theory (axioms) within which we reason. We need to find a semantic relation (e.g., equivalence, more/less general) between the concepts at any two nodes in two trees. However, these are usually complex concepts obtained by suitably combining the corresponding concepts of labels. For example, suppose we want to find a relation between $C1_2$ (which, intuitively, stands for the concept of electronic products which are personal computers) and $C2_2$ (which, intuitively, stands for the concept of electronic products which are PCs). In this case, we should realize that they have the same extension, and therefore, that they are equivalent.

The rest of this section concentrates on technical details of each of four macro steps of the algorithm we have outlined above.

5.3.2 Step 1. Compute *concepts of labels*

During this step we compute concepts of labels for all labels in two trees. A natural choice is to take the label itself as a placeholder for its concept. For instance, the label *camera* is the best string which can be used with the

purpose of characterizing "all documents which are about cameras". This is also what we have done in the previous examples: we have taken labels to stand for their concepts. Collapsing the notions of label and of concept of label is in fact a reasonable assumption, which has been implicitly made in many previous works on syntactic matching (see, e.g., [146, 58]).

However, it has a major drawback since labels are most often written in some not well defined subset of natural language and, as a result, natural language presents many ambiguities. For instance, there are many possible different ways to state the same concept (as we have with *Quantity* and *Amount*); dually, the same sentence may mean many different things (e.g., think of the label *camera* again, being a photographic camera or television camera); *Quantity* and *Amount*, though being different words, for our purposes have the same classification role.

Among the key ideas underlying semantic matching is the one that labels, which are written in natural language, are translated into a propositional concept language, such as propositional description logic language [10]. Specifically, atomic formulas are atomic concepts, written as single words or multi-words. Complex formulas are obtained by combining atomic concepts using the logical operators, such as conjunction ($\sqcap$), disjunction ($\sqcup$), and negation ($\neg$). Note that negation can only be applied to atomic concepts. There are also comparison operators, such as less general ($\sqsubseteq$), more general ($\sqsupseteq$), and equivalence ($=$). The interpretation of these operators is the standard set-theoretic interpretation[2].

The reasons for choosing a simple propositional description logics language are as follows. First, given its set-theoretic interpretation, it "maps" naturally to the real world semantics. Second, natural language labels

[2]Note that we do not introduce any new knowledge representation formalism here. We rely on the existing one, which is a propositional description logic. Therefore, we limit its presentation to an informal discussion, which we believe is appropriate according to the whole contribution of the approach. Also note that in practice we straightforwardly translate the natural language labels into propositional logic formulas [103].

used in classifications and XML schemas are usually short expressions or phrases having simple structure. These phrases can be converted into a formula in our knowledge representation formalism with no or little loss in the meaning [95]. Finally, these formulas can be converted into equivalent formulas in a propositional logic language with boolean semantics. Thus, technically, concept of a label is the propositional formula which stands for the set of data instances (documents) that one would classify under a label it encodes.

We compute atomic concepts, as they are denoted by atomic labels (namely, labels of single words or multi-words), as the senses provided by WordNet [163]. In the simplest case, an atomic label generates an atomic concept. However, atomic labels with multiple senses or labels with multiple words generate complex concepts. The translation process from labels to concepts follows the ideas of [31] where the main steps are as follows (note that the first two steps are common to many matching approaches):

Tokenization. Labels at nodes are parsed into tokens by a tokenizer which recognises punctuation, cases, digits, etc. Thus, for instance, *Photo_and_Cameras* becomes $\langle photo, and, cameras \rangle$.

Lemmatization. Tokens at labels are further lemmatized, namely they are morphologically analyzed in order to find all their possible basic forms. Thus, for instance, *cameras* is associated with its singular form, *camera*.

Building atomic concepts. WordNet is queried to extract the senses of lemmas of tokens identified during the previous step. For example, the label *Cameras* has the only one token *cameras*, and one lemma *camera*, and from WordNet we find out that *camera* has two senses.

Atomic formulas are WordNet [163] *senses* of lemmas obtained from single words (e.g., cameras) or multi-words (e.g., digital cameras).

Building complex concepts. When existing, all tokens that are prepositions, punctuation marks, conjunctions (or strings with similar roles) are translated into logical connectives and used to build complex concepts out of the atomic concepts built previously. Thus, for instance, commas and conjunctions are translated into disjunctions, prepositions like *of* and *in* are translated into conjunctions, and so on. For example, the concept of label *cameras and photo* is computed as follows: $C_{Cameras\_and\_Photo} = \langle Cameras, senses_{WN\#2} \rangle \sqcup \langle Photo, senses_{WN\#1} \rangle$, where $senses_{WN\#2}$ is taken to be disjunction of the two senses that WordNet attaches to *Cameras*, and similarly for *Photo*. Notice that the natural language conjunction "and" has been translated into the logical disjunction "$\sqcup$" [147].

5.3.3 Step 2. Compute *concepts at nodes*

During this step we compute concepts at nodes for all nodes in two trees. We analyze the meaning of the *positions* of labels at nodes in a tree. By doing this concepts of labels are extended to concepts at nodes. This is required to capture the knowledge residing in the structure of a tree, namely the context in which the given concept at label occurs [94, 92].

Technically, concepts of nodes are written in the same propositional logical language as concepts of labels. Thus, concept at a node is the propositional formula which represents the set of data instances (documents) which one would classify under a node, given that it has a certain label and that it is in a certain position in a tree. XML schemas and classifications are hierarchical structures where the path from the root to a node uniquely identifies that node (and also its meaning). Thus, following an *access crite-*

rion semantics [109], the logical formula for a concept at node is defined as a conjunction of concepts of labels located in the path from the given node to the root. For example, $C2_6 = Electronics_2 \sqcap Cameras\_and\_Photo_2 \sqcap Digital\_Cameras_2$, which encodes the concept at node 6 in O2, describing all the data instances of the electronic photography products which are digital cameras.

5.3.4 Step 3. Compute relations among *concepts of labels*

During this step we compute relations among atomic concepts of labels for all pairs of labels in two trees. By doing this we build a theory or domain knowledge for the given input two ontologies codified as a set of semantic relations between atomic concepts of labels occurring in two trees. This is the background theory within which we reason.

Relations between atomic concepts of labels could be computed by using any element level matchers discussed in Chapter 3. However, most of those techniques, e.g., string-based, have to be modified in order to return (instead of a similarity measure) a semantic relation R, as defined in §5.2.2.

For example, $Beverages_1$ can be found less general than $Food_2$. In fact, according to WordNet, *beverages* is hyponym (subordinate word) of *food*. Notice, in WordNet *beverages* has 1 sense, while *food* has 3 senses. Some sense filtering techniques have to be used to discard the irrelevant senses for the given context, see [103] for details. Similarly, by using string-based matchers (common suffix) we can find that PID_1 is equivalent to ID_2.

The result of step 3 is a matrix of the relations holding between atomic concepts of labels. A part of this matrix for the example of Figure 5.5 is shown in Table 5.1.

Table 5.1: The matrix of semantic relations holding between atomic concepts of labels

	$Cameras_2$	$Photo_2$	$Digital\_Cameras_2$
$Photo_1$	idk	$=$	idk
$Cameras_1$	$=$	idk	$\sqsupseteq$

5.3.5 Step 4. Compute relations among *concepts at nodes*

During this step we compute relations among concepts at nodes for all pairs of nodes in two trees. This problem cannot be solved simply by asking an oracle, such as WordNet, containing static knowledge. The situation is far more complex, being as follows:

- We have background knowledge or theory computed after Step 3 for the given input two ontologies, namely a set of semantic relations between atomic concepts of labels occurring in two trees.

- Concepts of labels and concepts at nodes are codified as complex propositional formulas. In particular, concepts at nodes are conjunctions of concepts of labels, while concepts of labels, in turn, could be full propositional formulas. We have them computed from Step 1 and Step 2.

- We need to find a semantic relation, namely equivalence, more/less general, disjointness, between the concepts at any two nodes in two trees. We translate all the semantic relations into propositional connectives in the obvious way, namely: equivalence ($=$) into equivalence ($\leftrightarrow$), more general ($\sqsupseteq$) and less general ($\sqsubseteq$) into implication ($\leftarrow$ and $\rightarrow$, respectively), disjointness ($\bot$) into negation ($\neg$) of the conjunction ($\wedge$).

- Build a matching formula for each pair of concepts from two ontologies. The criterion for determining whether a relation holds between two concepts is the fact that it is entailed by the premises (theory). Therefore, a matching query is created as a formula of the following form:

$$Axioms \rightarrow rel(context_1, context_2) \tag{5.1}$$

for each pair of concepts for which we want to test the relation. $context_1$ is the concept at node under consideration in tree 1, while $context_2$ is the concept at node under consideration in tree 2. rel (within $=$, $\sqsubseteq$, $\sqsupseteq$, $\perp$) is the semantic relation (suitably translated into a propositional connective) that we want to prove holding between $context_1$ and $context_2$. The $Axioms$ part is the conjunction of all the relations (suitably translated) between atomic concepts of labels mentioned in $context_1$ and $context_2$. For example, the task of matching $C1_3$ and $C2_6$, requires the following axioms:

$$(Electronics_1 \leftrightarrow Electronics_2) \wedge (Cameras_1 \leftrightarrow Cameras_2) \wedge$$
$$(Photo_1 \leftrightarrow Photo_2) \wedge (Cameras_1 \leftarrow Digital\_Cameras_2).$$

- Check for validity of formula (5.1), namely that it is true for all the truth assignments of all the propositional variables occurring in it. A propositional formula is valid if and only if its negation is unsatisfiable. The unsatisfiability is checked by using a SAT solver.

Technically, we initially reformulate the tree matching problem into a set of node matching problems (one problem for each pair of nodes). Finally, we translate each node matching problem into a propositional validity problem. Let us discuss in detail the tree matching algorithm, see

Algorithm 1 for the pseudo-code.

Lines 1-12 define variables and datatypes. cLabel and cNode are used to memorize concepts of labels and concepts at nodes, respectively. The other names of the variables either follow in an obvious way the notions which have already been introduced, or will be explained at time of their use. In line 30, the treeMatch function inputs two trees of Nodes (source and target). It starts from the element level matching. Thus, in line 35, the matrix of relations holding between atomic concepts of labels (cLabsMatrix) is populated by the fillCLabsMatrix function. Two loops are run over all the nodes of source and target trees in lines 50-111 and 53-110 in order to formulate all the node matching problems. Then, for each node matching problem, a pair of propositional formulas encoding concepts at nodes and relevant relations holding between concepts of labels are taken by using the getCnodeFormula and extractRelMatrix functions, respectively. The former are memorized as context_1 and context_2 in lines 52 and 55. The latter are memorized in relMatrix in line 80. In order to reason about relations between concepts at nodes, the premises (axioms) are built in line 81. These are a conjunction of atomic concepts of labels which are related in relMatrix. Finally, in line 82, the relations holding between the concepts at nodes are calculated by nodeMatch and are reported in line 150 (cNodesMatrix).

A part of the cNodesMatrix matrix for the example of Figure 5.5 is shown in Table 5.2.

Table 5.2: The matrix of semantic relations holding between concepts at nodes (the matching result)

	$C2_1$	$C2_2$	$C2_3$	$C2_4$	$C2_5$	$C2_6$
$C1_3$	$\sqsubseteq$	idk	$=$	idk	$\sqsupseteq$	$\sqsupseteq$

nodeMatch translates each node matching problem into a propositional validity problem. It checks for sentence validity by proving that its nega-

Algorithm 1 The tree matching algorithm

```
1: Node: struct of
2:         int nodeId;
3:         String label;
4:         String cLabel;
5:         String cNode;
6:         Node parent;
7:         AtomicConceptOfLabel[ ] ACOL;
8: AtomicConceptOfLabel: struct of
9:                         int id;
10:                        String token;
11:                        String[ ] wnSenses;
12: String[ ][ ] cLabsMatrix, cNodesMatrix;

30: String[ ][ ] treeMatch(Tree of Nodes source, target)
31: Node sourceNode, targetNode;
32: int i, j;
33: String[ ][ ] relMatrix;
34: String axioms, context1, context2;
35: cLabsMatrix = fillCLabMatrix(source, target);

50: for each sourceNode ∈ source do
51:    i = getNodeId(sourceNode);
52:    context1 = getCnodeFormula(sourceNode);
53:    for each targetNode ∈ target do
54:       j = getNodeId(targetNode);
55:       context2 = getCnodeFormula(targetNode);

80:       relMatrix = extractRelMatrix(cLabsMatrix, sourceNode, targetNode);
81:       axioms = mkAxioms(relMatrix);
82:       cNodesMatrix[i][j] = nodeMatch(axioms, context1, context2);

110:   end for
111: end for
150: return cNodesMatrix;
```

tion is unsatisfiable. The algorithm uses, depending on a matching task, either ad hoc reasoning techniques [102], or standard DPLL-based SAT solvers [131, 51, 50]. From the example in Figure 5.5, trying to prove that $C2_6$, which is defined as $(Electronics_2 \wedge (Cameras_2 \vee Photo_2) \wedge Digital\_Cameras_2)$, is less general than $C1_3$, which, in turn, is defined as $(Electronics_1 \wedge (Photo_1 \vee Cameras_1))$, requires constructing formula (5.2), negation of which turns out to be unsatisfiable, and therefore, we can conclude that the less general relation holds.

$$\begin{aligned}&((Electronics_1 \leftrightarrow Electronics_2) \wedge (Photo_1 \leftrightarrow Photo_2) \wedge \\ &(Cameras_1 \leftrightarrow Cameras_2) \wedge (Digital\_Cameras_2 \rightarrow Cameras_1)) \rightarrow \\ &((Electronics_2 \wedge (Cameras_2 \vee Photo_2) \wedge Digital\_Cameras_2) \rightarrow \\ &(Electronics_1 \wedge (Photo_1 \vee Cameras_1)))\end{aligned} \tag{5.2}$$

5.4 Summary

In this chapter we have identified semantic matching as the new approach for performing generic matching. We discussed the main motivations behind the approach as well as its key notions. Then, the main four macro steps of the semantic matching algorithm has been presented and described with the help of examples and pseudo-code.

Chapter 6

Semantic matching with attributes

So far we have focused only on simple concept hierarchies. However, if we deal with, e.g., XML schemas, their elements may have attributes, which, in turn, may require an additional treatment. This chapter discusses an extension of the semantic matching approach for handling attributes. Material presented in this chapter has been published in [99, 103].

We first describe the idea of how to handle attributes in the settings of the semantic matching approach (§6.1). Then, we substantiate it by considering two possible alternatives when dealing with attributes, namely exploiting datatypes (§6.2) and ignoring datatypes (§6.3).

6.1 The idea of the approach

We discuss our approach for handling attributes with the help of example of Figure 5.5 (p.96). Attributes are $\langle attribute - name, type \rangle$ pairs associated with elements. Names for the attributes are usually chosen such that they describe the roles played by the domains in order to ease distinguishing between their different uses. For example, in O1 of Figure 5.5, the attributes PID and $Name$ are defined on the same domain *string*, but

their intended uses are the internal (unique) product identification and representation of the official product's names, respectively. There are no strict rules telling us when data should be represented as elements, or as attributes, and obviously there is always more than one way to encode the same data. For example, in O1, $PIDs$ are encoded as *string*s, while in O2, IDs are encoded as *int*s. However, both attributes serve for the same purpose of the unique product's identification. These observations suggest two possible ways to perform semantic matching with attributes: (*i*) taking into account datatypes, and (*ii*) ignoring datatypes.

The semantic matching approach is based on the idea of matching concepts, not their direct physical implementations, such as elements or attributes. If names of attributes and elements are abstract entities, therefore, they allow for building arbitrary concepts out of them. Instead, datatypes, being concrete entities, are limited in this sense. Thus, a plausible way to match attributes using the semantic matching approach is to discard the information about datatypes. In order to support this claim, let us consider both cases in turn.

6.2 Exploiting datatypes

In order to reason with datatypes we have created a *datatype ontology*, $\boldsymbol{O_D}$, specified in OWL [217, 52]. It describes the most often used XML schema built-in datatypes and relations between them. The backbone taxonomy of $\boldsymbol{O_D}$ is based on the following rule: *the specialization relation holds between two datatypes if and only if their value spaces are related by set inclusion.* Some examples of axioms of $\boldsymbol{O_D}$ are: *float* $\sqsubseteq$ *double*, *int* $\perp$ *string*, *anyURI* $\sqsubseteq$ *string*, and so on. Let us discuss how datatypes are plugged within four macro steps of the semantic matching algorithm (§5.3).

Steps 1,2. Compute concepts of labels and concepts at nodes. In order to handle attributes, we extend our knowledge representation formalism with the quantification construct and datatypes. Thus, we compute concepts of labels and concepts at nodes as formulas in description logics, in particular, using $\mathcal{ALC}(\mathcal{D})$ [10]. For example, $C1_7$, namely, the concept at node describing all the string data instances which are the names of electronic photography products is encoded as follows:

$$C1_7 = Electronics_1 \sqcap (Photo_1 \sqcup Cameras_1) \sqcap \exists Name_1.string.$$

Step 3. Compute relations among concepts of labels. During this step we add a *Datatype* element level matcher. It takes as input two datatypes, it queries $\boldsymbol{O_D}$ and retrieves a semantic relation between them. For example, from axioms of $\boldsymbol{O_D}$, the *Datatype* matcher can learn that *float* $\sqsubseteq$ *double*, and so on.

Step 4. Compute relations among concepts at nodes. In the case of attributes, the node matching problem is translated into a DL formula, which is further checked for its validity using sound and complete procedures. The system we use is Racer [112]. From the example in Figure 5.5, trying to prove that $C2_{10}$ is less general than $C1_9$, requires constructing the following formula:

$$\begin{aligned}
&((Electronics_1 = Electronics_2) \sqcap (Photo_1 = Photo_2) \sqcap \\
&(Cameras_1 = Cameras_2) \sqcap (Price_1 = Price_2) \sqcap (float \sqsubseteq double)) \sqcap \\
&(Electronics_2 \sqcap (Cameras_2 \sqcup Photo_2) \sqcap \exists Price_2.float) \sqcap \\
&\neg(Electronics_1 \sqcap (Photo_1 \sqcup Cameras_1) \sqcap \exists Price_1.double)
\end{aligned} \tag{6.1}$$

It turns out that formula (6.1) is unsatisfiable. Therefore, $C2_{10}$ is less general than $C1_9$. However, this result is not what the user expects. In

fact, both $C1_9$ and $C2_{10}$ describe prices of electronic products, which are photo cameras. The storage format of *prices* in O1 and O2 (i.e., *double* and *float*, respectively) is not an issue at this level of detail.

Thus, another semantic solution of taking into account datatypes would be to build abstractions out of the datatypes, e.g., *float, double, decimal* should be abstracted to type *numeric*, while *token, name, normalizedString* should be abstracted to type *string*, and so on. However, even such abstractions do not improve the situation, since we may have, for example, an *ID* of type *numeric* in the first schema, and a conceptually equivalent *ID*, but of type *string*, in the second schema. If we continue building such abstractions, we result in having that *numeric* is equivalent to *string* in the sense that they are both datatypes.

The last observation suggests that for the semantic matching approach to be correct, we should assume, that all the datatypes are equivalent. Technically, in order to implement this assumption, we should add corresponding axioms (e.g., *float* = *double*) to the premises of formula (5.1), see p.104. On the one hand, with respect to the case of not considering datatypes (see §6.3), such axioms do not affect the matching result from the quality viewpoint. On the other hand, datatypes make the matching problem computationally more expensive by requiring to handle the quantification construct.

6.3 Ignoring datatypes

In this case, information about datatypes is discarded. For example, $\langle Name, string\rangle$ becomes $Name$. Then, the semantic matching algorithm builds concepts of labels out of attribute's names in the same way as it does in the case of element's names, and so on (§5.3). A part of the cNodesMatrix with relations holding between attributes for the example of Figure 5.5

is presented in Table 6.1. Notice that this solution allows the computation of mapping elements not only between the attributes, but also between attributes and elements.

Table 6.1: Attributes: the matrix of semantic relations holding between concepts at nodes (the matching result)

	$C2_7$	$C2_8$	$C2_9$	$C2_{10}$
$C1_6$	$=$	idk	idk	idk
$C1_7$	idk	$\sqsupseteq$	idk	idk
$C1_8$	idk	idk	$=$	idk
$C1_9$	idk	idk	idk	$=$

6.4 Summary

In this chapter we have presented how attributes are handled within the semantic matching settings. We have argued that a plausible way to match attributes using the semantic matching approach is to discard the information about datatypes.

The task of determining mapping elements typically represents a first step towards the ultimate goal of, for example, schema integration, data integration, agent communication, query answering, and so on (Chapter 1). Although information about datatypes will be necessary for accomplishing an ultimate goal, we do not discuss this issue any further since in this thesis we concentrate only on the alignment discovery task.

Chapter 7

Iterative semantic matching

In this chapter we present an extension of the semantic matching approach to deal in a fully automated way with the lack of background knowledge in matching tasks. The key idea is to use semantic matching iteratively. The benefits of this extension include: saving some of the pre-match efforts, improving the quality of match via iterations, and enabling the future reuse of the newly discovered knowledge.

Material of this chapter has been developed in collaboration with Mikalai Yatskevich. In particular, some algorithms presented here exploit a library of element level semantic matchers first introduced in [101]. Note that the library of [101] does not constitute a contribution of this thesis. The algorithms presented in this chapter provide the settings that allow for a practical use for some of those highly contextual element level semantic matchers. More precisely, in terms of [101], these are the element level matchers that have the third approximation level. We also note that those element level semantic matchers technically existed before in the S-Match system, however (being highly approximate) without a practical application [103]. Material of this chapter has been published in [100].

In this chapter we first provide some motivations behind the iterative semantic matching. This is done by examples of the problem of the lack of background knowledge in matching tasks and some of its possible solutions (§7.1). Then, we present the main building blocks of the iterative semantic matching algorithm (§7.2), while its details, namely, algorithms for critical points discovery (§7.3) and critical points resolution (§7.4) are discussed in sequel.

7.1 Motivation: lack of knowledge

Recent industrial-strength evaluations of matching systems, see, e.g., [9, 77], show that lack of background knowledge, most often domain specific knowledge, is one of the key problems of matching systems these days. In fact, for example, should *PO* match *Post Office*, *Purchase Order*, or *Project Officer*? At present, most state of the art systems, for the tasks of matching thousands of nodes, perform not with such high values of *recall*[1], namely ∼30%, as in cases of toy examples, where the recall was most often around 90%. Also, contributing to this problem, [148] shows that complex matching solutions requiring months of algorithms design and development on big tasks may perform as badly as a baseline matcher requiring one hour burden.

In order to understand better the above observations, let us consider a preliminary analytical comparative evaluation of some state of the art matching systems together with a baseline solution[2] on three large (hundreds and thousands of entities in each ontology) real world test cases. Some indicators of the test cases complexity are given in Table 9.1 (p.148)

[1]Recall is a completeness measure of matching results, see §9.1, p.145 for a definition.

[2]This matcher does simple string comparison among sets of labels on the paths from nodes under consideration to the roots of the input trees, see [9].

and correspond to the matching tasks #6, 7, and 8 of the table.

These test cases were first introduced in [9] and used in the OAEI-2005 ontology matching evaluation campaign[3]. As match quality measures we focus here on recall which is a completeness measure. It varies in the [0 1] range; the higher the value, the smaller the set of correct correspondences which have not been found (§9.1). The summarized evaluation results for all the three matching tasks are shown in Figure 7.1. Notice that the results for such matching systems as *OMAP*, *CMS*, *Dublin20*, *Falcon*, *FOAM*, *OLA*, and *ctxMatch2*, were taken from OAEI-2005, see [77], while evaluation results for the *baseline* matcher and *S-Match* were taken from [9]. As Figure 7.1 shows, none of the considered matching systems performs with a value of recall which is higher than 32%.

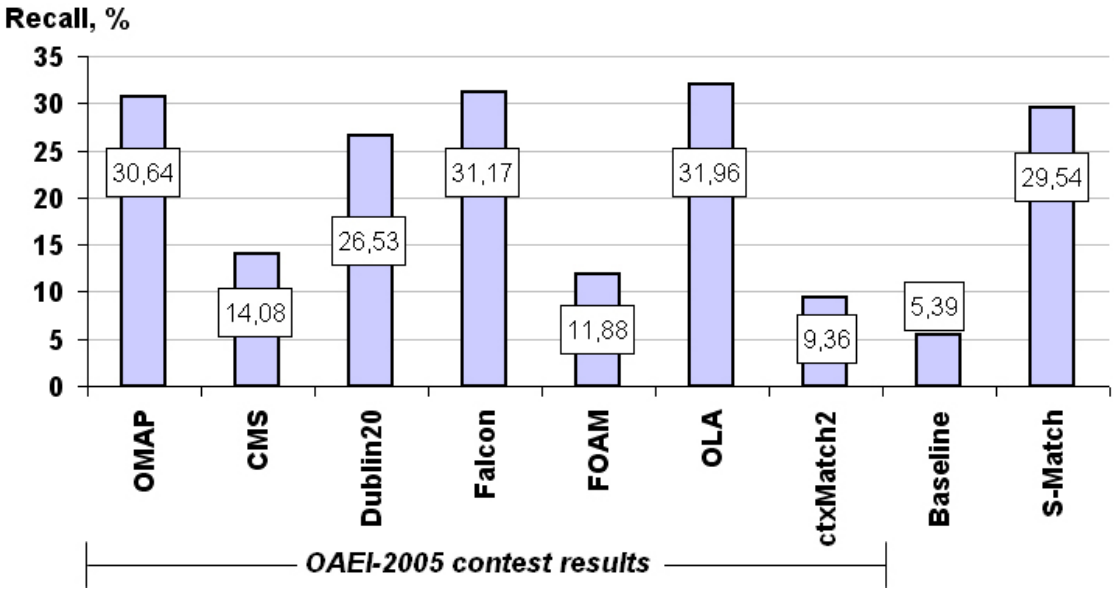


Figure 7.1: Analytical comparative evaluation

There are multiple strategies to attack the problem of the lack of background knowledge. One of the most often used methods so far is to declare the missing axioms manually as a pre-match effort. Some other plausible strategies include: (*i*) extending stop word lists, (*ii*) expanding acronyms,

[3]See for details, http://oaei.ontologymatching.org/2005/

(*iii*) reusing the previous match results, (*iv*) querying the web, (*v*) using, if available, domain specific sources of knowledge, and so on.

7.2 The iterative tree matching algorithm

We first discuss the idea behind the approach and how the tree matching algorithm (§5.3.5) should be modified in order to suitably enable iterations. Then, we present the main building blocks of the iterative tree matching algorithm, namely, algorithms for critical points discovery and critical points resolution. The algorithms are discussed via a running example. We consider lightweight ontologies O1 and O2 shown in Figure 7.2, which are small parts of *Google* and *Looksmart*. Notation follows the one introduced in §5.2.1 (p.91).

Figure 7.2: Fragments of Google and Looksmart

7.2.1 The idea in a nutshell

We propose a fully automated solution to address the problem of the lack of knowledge by using semantic matching iteratively. The idea is to repeat *Step 3* (§5.3.4) and *Step 4* (§5.3.5) of the matching algorithm for some critical (hard) matching tasks. In particular, the result of SAT is analyzed. We identify critical points in the matching process, namely mapping elements with the *idk* relation where a stronger relation (e.g., more general, equivalence) should have taken place. We attack critical points by exploiting sophisticated element level matchers of [101] (see also §3.2.1) which use the *deep* information encoded in WordNet, e.g., its internal structure. Then, taking into account the newly discovered knowledge as additional axioms, we re-run SAT solver on a critical task. Finally, if SAT returns false, we save the newly discovered knowledge, thereby enabling its future reuse.

7.2.2 The iterative tree match algorithm

The iterative tree matching algorithm is shown as Algorithm 2. The numbers on the left indicate where the new code must be positioned in Algorithm 1 (§5.3.5, p.106).

Algorithm 2 The iterative tree matching algorithm

```
13: Boolean[ ][ ] cPointsMatrix;

100:    if (cPointsDiscovery(sourceNode, targetNode) == true) then
101:        cPointsMatrix[i][j] = true;
102:      ResolveCpoint(sourceNode, targetNode, context1, context2);
103:    end if
```

In line 13, we introduce cPointsMatrix which memorizes critical points. Semantic matching algorithm works in a top-down manner, and hence, mismatches among the top level classes of ontologies imply further mismatches

between their descendants. Thus, the descendants should be processed only after the critical point at those top level nodes has been resolved. This is ensured by suitably positioning the new functions (enabling iterations) in a double loop of Algorithm 1. Hence, in line 100, we check with the help of cPointsDiscovery function if the nodes under consideration are the critical point. If they indeed represent the critical point, they are (memorized and) resolved by using the ResolveCpoint function (line 102). In the example of Figure 7.2, critical points which are determined include, for instance, $\langle C1_2, C2_3\rangle$, $\langle C1_3, C2_3\rangle$, and $\langle C1_4, C2_4\rangle$.

An updated cNodesMatrix, after running the iterative tree matching algorithm, is presented in Table 7.1. Comparing it with the non-iterative matching algorithm result, which is further reported in Table 7.3, we can see that having identified and resolved the $\langle C1_3, C2_3\rangle$ critical point, we also managed to discover the new correspondences, namely between $C2_3$ and $C1_9$, $C1_{10}$, $C1_{11}$.

Table 7.1: Recomputed cNodesMatrix: relations among concepts at nodes

	$C1_1$	$C1_2$	$C1_3$	$C1_4$	$C1_5$	$C1_9$	$C1_{10}$	$C1_{11}$
$C2_1$	$=$	$\sqsupseteq$	$\sqsupseteq$	$\sqsupseteq$	$\sqsupseteq$	$\sqsupseteq$	$\sqsupseteq$	$\sqsupseteq$
$C2_3$	$\sqsubseteq$	$=$	$=$	idk	idk	$\sqsupseteq$	$\sqsupseteq$	$\sqsupseteq$

Having computed all the mapping elements for a given pair of ontologies, the identified critical relations are validated by a human user. In particular, user decides if the type of relation determined automatically is appropriate for the given pair of ontologies. For example, is it appropriate that $Games_1 \leftrightarrow Entertainment_2$ or a weaker relation, namely $Games_1 \rightarrow Entertainment_2$, should have taken place ? User decides either to use this relation once (only for this pair of ontologies) or to save it in a domain specific oracle in order to enable its future reuse.

Finally, it is worth noting that iterative semantic matching algorithm

amounts to *robustness* of the semantic matching. In fact, even if non-iterative semantic matching determines a (false) top level mismatch, this can be discovered and resolved by applying Algorithm 2. Thus, avoiding further propagation of possible mismatches between the descendants of the initially mismatched top level nodes.

7.3 The critical points discovery algorithm

The algorithm for discovering critical points is based on the following intuitions:

- each *idk* relation in cNodesMatrix is potentially a critical point, but it is not always the case;
- since critical points arise due to lack of background knowledge, the clue is to check whether some other nodes located below the critical nodes (those representing a critical point) are related somehow. In case of a positive result the actual nodes are indeed the critical point; they represent a false alarm otherwise.

Algorithm 3 formalizes these intuitions. In particular, the first condition mentioned above is checked in line 4. Verifying the second condition is more complicated. We call a relation holding between descendants of the potentially critical nodes a *support relation*. The support relation holds if there exists atomic concept of label (sACOL) in the descendants of sourceNode which is related in cLabsMatrix (by any semantic relation, except *idk*) to any atomic concept of label (tACOL) in the descendants of targetNode. This condition is checked in a double loop in lines 7-13. Finally, if both conditions are satisfied, the cPointsDiscovery function concludes that the nodes under consideration are the critical point (line 10). Under the given

Algorithm 3 The critical points discovery algorithm

```
1: Boolean cPointsDiscovery(Node sourceNode, targetNode)
2: Node[ ] sDescendant, tDescendant;
3: ACOL sACOL, tACOL;
4: if (cNodesMatrix[sourceNode.nodeID][targetNode.nodeID]==“idk”)
   then
5:    sDescendant = getSubTree(sourceNode);
6:    tDescendant = getSubTree(targetNode);
7:    for each sACOL ∈ sDescendant.ACOL do
8:       for each tACOL ∈ tDescendant.ACOL do
9:          if cLabsMatrix[sACOL.id][tACOL.id] != “idk” then
10:            return true;
11:         end if
12:      end for
13:   end for
14: else
15:   return false;
16: end if
```

critical points discovery strategy, performing such a look up over the cLabsMatrix makes sense, obviously, only when sourceNode and targetNode are non-leaf nodes.

For example, suppose we want to match $C1_3$ and $C2_3$. Parts of cLabsMatrix and cNodesMatrix with respect to the given matching task are shown in Table 7.2 and Table 7.3. Notice that the relations in Table 7.2 were computed by applying element level matchers of [101], namely *WordNet*, *prefix*, *suffix*, *edit distance*, and *n-gram* (see also §3.2.1) in the order as they were stated [103].

Table 7.2: cLabsMatrix: relations holding among atomic concepts of labels

	TOP_1	$Games_1$	$Board\_Games_1$
TOP_2	$=$	idk	idk
$Entertainment_2$	idk	idk	idk
$Games_2$	idk	$=$	$\sqsupseteq$

Table 7.3: cNodesMatrix: relations holding among concepts at nodes

	$C1_1$	$C1_2$	$C1_3$	$C1_4$	$C1_5$	$C1_9$	$C1_{10}$	$C1_{11}$
$C2_1$	$=$	$\sqsupseteq$	$\sqsupseteq$	$\sqsupseteq$	$\sqsupseteq$	$\sqsupseteq$	$\sqsupseteq$	$\sqsupseteq$
$C2_3$	$\sqsubseteq$	idk	idk	idk	idk	idk	idk	idk

In cNodesMatrix (Table 7.3) the relation between $C1_3$ and $C2_3$ is idk[4]. In cLabsMatrix (Table 7.2) there is a support relation for the given matching problem, e.g., $Board\_Games_1 \sqsubseteq Games_2$. Therefore, relation between $C1_3$ and $C2_3$ represents the critical point and we should reconsider the relation holding between $Games_1$ and $Entertainment_2$ in cLabsMatrix.

Finally, it is worth noting that this algorithm also properly handles nodes, which are indeed dissimilar, e.g., $C1_5$ and $C2_2$ are determined not to be the critical point.

7.4 The critical points resolution algorithm

Let us discuss how the critical points are resolved, see Algorithm 4.

The ResolveCpoint function determines relations (cRel) for the critical points. Also, by exploiting the cNodesMatrixUpdate procedure, it updates accordingly cNodesMatrix. In particular, ResolveCpoint executes element level semantic matchers of [101], which have the third approximation level, over the atomic concepts of labels by using the GetMLibRel function (line 7). These matchers include: *Hierarchy Distance (HD)*, *WordNet Gloss (WNG)*, *Extended WordNet Gloss (EWNG)*, *Gloss Comparison (GC)*, and *Extended Gloss Comparison (EGC)*. By default, they are applied following the order (ExecutionList) as stated above. These matchers produce the re-

[4]Notice that the relation is *idk* since we deal with the classifications, while if the classifications of Figure 7.2 were encoded as taxonomies (by modifying in obvious way the construction of the *Axioms*) we could immediately deduce the less general relation between the nodes under consideration.

Algorithm 4 The critical points resolution algorithm

```
1: ResolveCpoint(Node sourceNode, targetNode, String context1, context2)
2: String cRel;
3: String[ ] ExecutionList;
4: ACOL sACOL, tACOL;
5: for each sACOL ∈ sourceNode.ACOL do
6:    for each tACOL ∈ targetNode.ACOL do
7:       cRel = GetMLibRel(ExecutionList, sACOL.wnSenses, tACOL.wnSenses);
8:       cLabsMatrix[sACOL.id][tACOL.id] = cRel;
9:    end for
10: end for
11: cNodesMatrixUpdate(sourceNode, targetNode, context1, context2);
```

lations which depend heavily on the context of the matching task. Thus, they cannot be applied in all the cases. For example, these matchers can find that *cat* is equivalent to *dog* since they are both *pets*, which can be appropriate for the context of some matching tasks (the example follows next), and obviously not appropriate for the context of some other matching tasks, for instance, requiring fine-grained distinctions in the domain of *animals*.

For example, a *Hierarchy Distance* matcher computes the equivalence relation if the distance between two input senses in the WordNet hierarchy is less than a given threshold value (e.g., 3) and returns *idk* otherwise. According to WordNet, *games* and *entertainment* have a common ancestor, which is *diversion*. The distance between these concepts is 2 (1 more general link and 1 less general). Therefore, the *HD* matcher concludes that $Games_1$ is equivalent to $Entertainment_2$. If the *HD* matcher fails, which is not the case in our example, we apply the other remaining matchers in the order as stated above. We do not discuss those matchers here, since they do not constitute the contribution of this thesis, see [101] for details. The example above was given to provide a complete account of the iterative semantic matching approach.

In line 8, we update cLabsMatrix with the critical relation, cRel, such that

in all the further computations and for the current pair of nodes this relation is available. Finally, given the new axiom ($Games_1 \leftrightarrow Entertainment_2$) we recompute (line 11)[5], by re-running SAT, the relation holding between the pair of critical nodes, thus determining that $C1_3 = C2_3$.

The iterative semantic matching algorithms have been implemented within the S-Match system. We do not discuss implementation details, since the pseudo-code was given with low level details, and therefore, its implementation is straightforward.

7.5 Summary

We have presented an automated approach to attack the problem of the lack of background knowledge by applying semantic matching iteratively. The key idea is to repeat *Step 3*, namely computing the relations between atomic concepts of labels, and *Step 4*, namely computing the relations between concepts at nodes, of the semantic matching algorithm for some critical (hard) matching tasks. The algorithms realizing the approach have been discussed with the help of examples and pseudo-code.

[5] cNodesMatrixUpdate performs functionalities identical to those of lines 80-82 in Algorithm 1, see p.106.

Chapter 8

Explaining semantic matching

Matching systems may produce effective alignments that may not be intuitively obvious to human users. In order for users to trust the alignments, and thus use them, they need information about them, e.g., they need access to the sources that were used to determine semantic correspondences between ontology entities. Explanations are also useful when matching large applications with thousands of entities, e.g., business product classifications, such as *UNSPSC* and *eCl@ss*. In such cases, automatic matching solutions will find a number of plausible correspondences, and hence user input is required for performing cleaning-up of the alignment. Finally, explanations can also be viewed and applied as argumentation schemas for negotiating alignments between agents [129]. This chapter is devoted to an extension of the semantic matching approach that enables explanation of the answers it produces, thus making the matching result intelligible.

Material presented in this chapter has been developed in collaboration with Deborah McGuinness, Paulo Pinheiro da Silva and Jérôme Euzenat and published in [154, 215, 75].

In this chapter we first present the information required for providing explanations of matching (§8.1). Then, we discuss our approach to explaining semantic matching (§8.2). In turn, details of the approach are provided in sequel, including default explanations (§8.2.1), explaining the basic matchers (§8.2.2), and explaining the matching process (§8.2.3). Finally, we discuss some implementation details (§8.3).

8.1 Justifications

We have presented the matching process as the use of basic matchers combined by strategies (Chapter 3). In order to provide explanations to users it is necessary to have information on both matters. In particular, this information involves justifications on the reason why a correspondence should hold or not.

8.1.1 Information about basic matchers

When matching systems return alignments, users may not know which external sources of background knowledge were used, when these sources were updated, or whether the resulting correspondences was looked up or derived. However, ultimately, human users or agents have to make decisions about the alignments in a principled way. So, even when basic matchers simply rely on some external source of knowledge, users may need to understand where the information comes from, with different levels of detail.

Following [153], we call information about the origins of asserted facts the provenance information. Some examples of this kind of information include:

- external knowledge source name, e.g., WordNet;

- date and authors of original information;
- authoritativeness of the source, that is whether it is certified as reliable by a third party;
- name of a basic matcher, version, authors, etc. If the basic matcher relies on a logical reasoner, such as a SAT solver, e.g., SAT4J [131], some more meta-information about the reasoner may be made available:
  - the reasoning method, e.g., the Davis-Putnam-Longemann-Loveland (DPLL) procedure [51, 50];
  - properties, e.g., soundness and completeness characteristics of the result returned by the reasoner;
  - reasoner assumptions, e.g., closed world vs open world.

Additional types of information may also be provided, such as a degree of belief for an external source of knowledge from a particular community, computed by using some social network analysis techniques.

8.1.2 Process traces

Matching systems typically combine multiple matchers and the final alignment is usually a result of synthesis, abstraction, deduction, and some other manipulations of their results. Thus, users may want to see a trace of the performed manipulations. We refer to them as process traces. Some examples of this kind of information include:

- a trace of rules or strategies applied,
- support for alternative paths leading to a single conclusion,
- support for accessing the implicit information that can be made explicit from any particular reasoning path.

Users also may want to understand why a particular correspondence was not discovered, or why a discovered correspondence was ranked in a particular place, thereby being included in or excluded from the final alignment.

8.2 Explaining semantic matching: the approach

The goal of explanation is to take advantage of the previously mentioned types of information for rendering the matching process intelligible to the users. Among the key issues is to represent explanations in a simple and clear way [133].

In fact, while knowledge provenance and process traces may be enough for experts when they attempt to understand why a correspondence was returned, usually they are inadequate for ordinary users. Thus, raw justifications have to be transformed into an understandable explanation for each of the correspondences. Techniques are required for transforming raw justifications and rewriting them into abstractions that produce the foundation for what is presented to users.

Presentation support also needs to be provided for users to better understand explanations. Human users will need help in asking questions and obtaining answers of a manageable size. Additionally, agents may even need some control over requests, such as the ability to break large process traces into appropriate size portions, etc. Based on [153], requirements for process presentation may include:

- methods for breaking up process traces into manageable pieces,
- methods for pruning process traces and explanations to help the user find relevant information,

- methods for explanation navigation, including the ability to ask follow-up questions,
- methods for obtaining alternative justifications for answers,
- different presentation formats, e.g., natural language, graphs, and associated translation techniques,
- methods for obtaining justifications for conflicting answers,
- abstraction techniques.

In order to meet the above mentioned requirements the semantic matching approach [215] has been extended to use a third-party infrastructure for provenance and justification, namely the Inference Web (IW) infrastructure as well as the Proof Markup Language (PML) [151, 195]. Thus, meaningful fragments of semantic matching proofs can be loaded on demand. Users can browse an entire proof or they can restrict their view and refer only to specific, relevant parts of proofs. They can ask for provenance information related to proof elements (e.g., the origin of the terms in the proofs, the authors of the ontologies), and so on.

8.2.1 A default explanation

A default explanation of alignments should be a short, natural language, high-level explanation without any technical details. It is designed to be intuitive and understandable by ordinary users.

We concentrate on class matching and describe the approach with the help of example of a simple classification matching problem shown in Figure 5.1 (p.90). Notation follows the one introduced in §5.2.1 (p.91). Suppose an agent wants to exchange or to search for documents with another agent. The documents of both agents are stored in classifications $\mathsf{O1}$ and

O2, respectively. Recall that semantic matching takes as input these classifications, decomposes the tree matching problem into a set of node matching problems, each of which, in turn, is translated into a propositional validity problem, which is then resolved using a SAT solver.

From the example in Figure 5.1, trying to prove that the node with label *Europe* in O1 is equivalent to the node with label *Pictures* in O2, requires constructing the following formula:

$$\underbrace{((Images_1 \leftrightarrow Pictures_2) \wedge (Europe_1 \leftrightarrow Europe_2))}_{Axioms} \rightarrow$$
$$(\underbrace{(Images_1 \wedge Europe_1)}_{Context_1} \leftrightarrow \underbrace{(Europe_2 \wedge Pictures_2)}_{Context_2})$$

In this example, the negated formula is unsatisfiable, thus the equivalence relation holds between the nodes under consideration.

Suppose that agent O2 is interested in knowing why semantic matching suggested a set of documents stored under the node with label *Europe* in O1 as the result to the query – "find european pictures". A default explanation is presented in Figure 8.1.

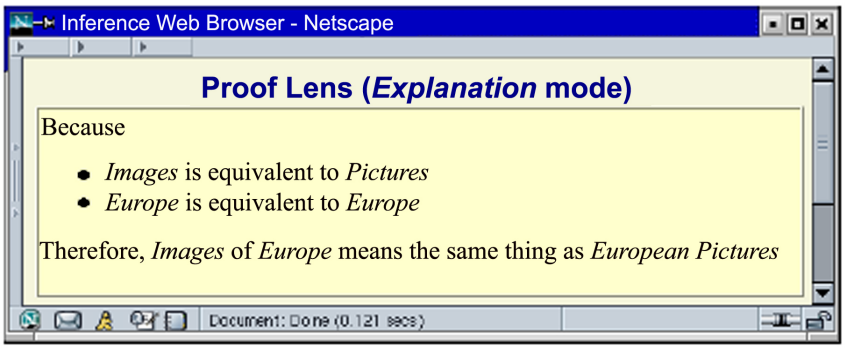


Figure 8.1: Default explanation in English

From the explanation in Figure 8.1, users may learn that *Images* in O1 and *Pictures* in O2 can be interchanged in the context of the query. Users may also learn that *Europe* in O1 denotes the same concept as *Europe (European)* in O2. Therefore, they can conclude that *Images of Europe* means the same thing as *European Pictures*.

8.2.2 Explaining basic matchers

Explaining basic matchers requires only to formulate the justification information.

Suppose that agents want to see the sources of background knowledge used in order to determine the correspondence. For example, which applications, publications, other sources, have been used to determine that *Images* is equivalent to *Pictures*. Figure 8.2 presents the source metadata for the default explanation of Figure 8.1.

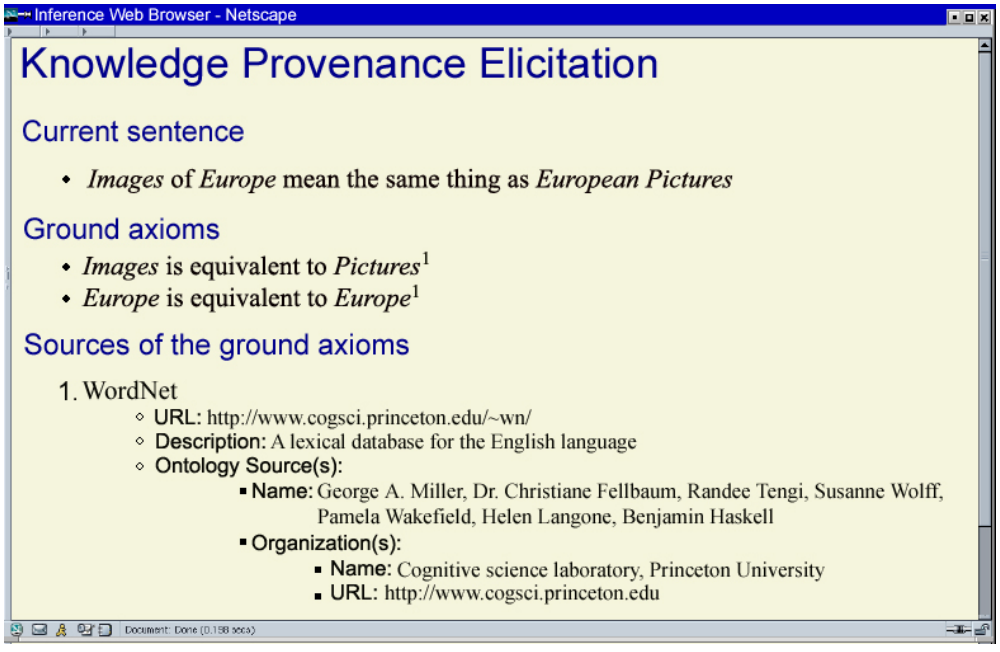


Figure 8.2: Source metadata information

In this case, both (all) the ground sentences used in the semantic matching proof came from WordNet. Using WorldNet, we learned that the first sense of the word *Pictures* is a synonym to the second sense of the word *Images*. Therefore, the semantic matching algorithm can conclude that these two words are equivalent words in the context of the answer. The meta-information about WordNet is also presented in Figure 8.2 as *sources of the ground axioms*. Further examples of explanations include providing meta-information about the other element level matchers used, i.e., those which are based not only on WordNet [101], the order in which the matchers are used, and so on.

8.2.3 Explaining logical reasoning

A complex explanation may be required if users are not familiar with or do not trust the inference engine(s) embedded in a matching system. As the web starts to rely more on information manipulations, instead of simply information retrieval, explanations of embedded manipulation or inference engines become more important. Semantic matching uses a propositional satisfiability engine, more precisely, this is the Davis-Putnam-Longemann-Loveland procedure [51, 50] as implemented in JSAT/SAT4J [131].

The task of a SAT solver is to find an assignment $\mu \in \{\top, \bot\}$ for atoms of a propositional formula φ such that φ evaluates to *true*. φ is *satisfiable* if and only if $\mu \models \varphi$ for some μ. If μ does not exist, φ is *unsatisfiable*. A *literal* is a propositional atom or its negation. A *clause* is a disjunction of one or more literals. φ is said to be in conjunctive normal form if and only if it is a conjunction of disjunctions of literals. The basic DPLL procedure recursively implements three rules: *unit resolution*, *pure literal* and *split*. We only consider the unit resolution rule to facilitate the presentation.

Let l be a literal and φ a propositional formula in conjunctive normal form. A clause is called a *unit clause* if and only if it has exactly one

unassigned literal. *Unit resolution* is an application of *resolution* to a unit clause.

$$unit\ resolution : \frac{\varphi \wedge \{l\}}{\varphi[l \mid \top]}$$

Unit resolution rule

Let us consider the propositional formula standing for the problem of testing if the concept at node with label *Europe* in O1 is less general than the concept at node with label *Pictures* in O2 of Figure 5.1. The propositional formula encoding the above stated matching problem is as follows:

$$((Images_1 \leftrightarrow Pictures_2) \wedge (Europe_1 \leftrightarrow Europe_2)) \rightarrow \\ ((Images_1 \wedge Europe_1) \rightarrow (Europe_2 \wedge Pictures_2))$$

Its intuitive reading, in turn, is as follows: "assuming that *Images* and *Pictures* denote the same concept, is there any situation such that the concept *Images of Europe* is less general than the concept *European Pictures*?". The proof of the fact that this is not the case is shown in Figure 8.3. Since the DPLL procedure of JSAT/SAT4J only handles conjunctive normal form formulas, in Figure 8.3, we show the conjunctive normal form of the above formula.

From the explanation in Figure 8.3, users may learn that the proof of the fact that the concept at node with label *Europe* in O1 is less general than the concept at node with label *Pictures* in O2 requires 4 steps and at each proof step (excepting the first one, which is a problem statement) the *unit resolution* rule is applied. Also, users may learn the assumptions that are made by JSAT/SAT4J. For example, at the second step, the DPLL procedure assigns the truth value to all instances of the atom *Europe*,

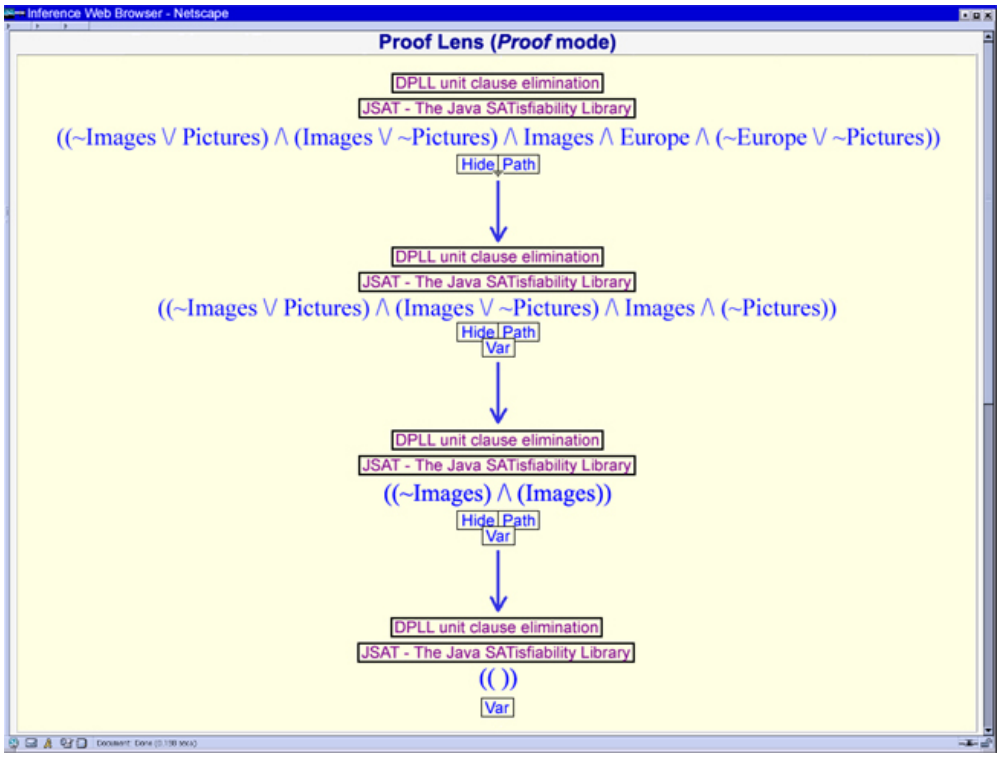


Figure 8.3: A graphical explanation of the unit clause rule

therefore making an assumption that there is a model where what an agent says about Europe is always true. According to the *unit resolution* rule, the atom *Europe* should then be deleted from the input sentence, and, hence it does not appear in the sentence of the step 2.

The explanation of Figure 8.3 represents some technical details (only the less generality test) of the default explanation in Figure 8.1. This type of explanations is the most verbose. It assumes that, even if the graphical representation of a decision tree is quite intuitive, the users have some background knowledge in logics and SAT. However, if they do not, they have a possibility to learn it by following the publications mentioned in the source metadata information of the DPLL *unit resolution* rule (*DPLL unit clause elimination*) and JSAT/SAT4J (*JSAT-The Java SATisfiability Library*).

Some further observations are to be made with respect to the other two rules. In the current version, the *pure literal* and *split* rules are explained in the same manner as the *unit resolution* rule. Two notes are to be made with respect to the *split* rule. The first is that, it is applied when we need to reason by case distinction, for example, when matching the node with label *Computers and Internet* in O1 and and the node with label *Cyberspace and Virtual Reality* in O2 of Figure 5.1. The second note is that, in the case of a satisfiable result, only a path of a decision tree standing for a successful assignment is represented. In the case of an unsatisfiable result a full decision tree is reported.

8.3 Implementation details

In order to provide these explanations, we have extended semantic matching and its implementation within the S-Match system to use the Inference Web infrastructure. Inference Web enables applications to generate portable and distributed explanations for any of their answers [151].

Figure 8.4 presents an abstract and partial view of the Inference Web infrastructure as used by S-Match. In order to use Inference Web to provide explanations, question answering systems need to produce proofs of their answers in PML, publish those proofs on the web, and provide a pointer to the last step in the proof. Inference Web also has a *registry* [152] of meta-information about proof elements, such as sources, e.g., publications, ontologies, inference engines and their rules. In the case of S-Match, the Inference Web repository contains meta-information about WordNet and JSAT/SAT4J.

In Inference Web, proof and explanation documents are formatted in PML and are composed of PML *node sets* [195]. Each node set represents a step in a proof whose conclusion is justified by any of a set of inference

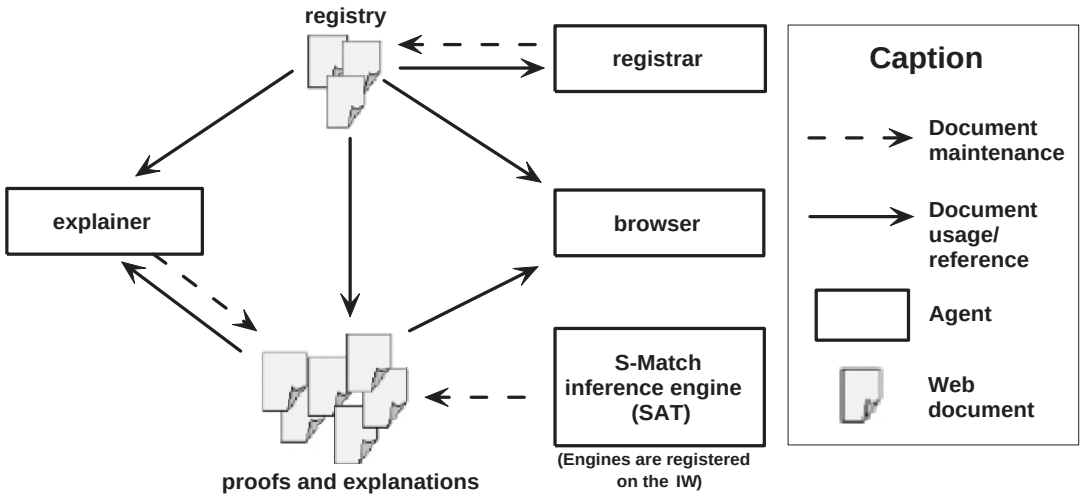


Figure 8.4: Inference Web infrastructure overview

steps associated with a node set. Node sets are OWL classes [217] and they are the building blocks of OWL documents describing proofs and explanations for application answers published on the web.

The *explainer* hides low-level information, e.g., the core reasoner rules, and exposes abstractions of the higher-level derived rules. Thus, many intermediate results can be dropped.

The *Inference Web browser* is used to present proofs and explanations. Exploiting PML properties, meaningful fragments of S-Match proofs can be loaded on demand. Users can browse an entire proof or they can limit their view and refer only to specific, relevant parts of proofs since each node set has its own URI that can be used as an entry point for proofs and proof fragments.

8.4 Summary

By extending S-Match to use the Inference Web infrastructure, we have demonstrated our approach for explaining answers from matching systems exploiting background ontological information and reasoning engines. The explanations can be presented in different styles allowing users to understand the correspondences and consequently to make informed decisions about them. The chapter also demonstrates that S-Match users can leverage the Inference Web tools, for example, for sharing, combining, browsing proofs, and supporting proof meta-information including background knowledge.

Delivering alignments to users, for inspection and revision, is an important topic not deeply developed so far in the ontology matching community. However, by using explanations, a matching system can provide users with meaningful prompts and suggestions on further steps towards the production of a desired result.

Part IV

Evaluation

Chapter 9

Evaluation setup

The increasing number of methods available for ontology matching suggests the need for evaluation of these methods. However, very few extensive experimental comparisons of algorithms are available. Matching systems are difficult to compare, but we believe that the ontology matching field can only evolve if evaluation criteria are provided. These should help system designers to assess the strengths and weaknesses of their systems as well as help application developers to choose the most appropriate algorithm.

Material presented in this chapter has been developed in collaboration with Jérôme Euzenat and Mikalai Yatskevich. Also the work on data set construction (for the evaluation of quality of the results produced by matching systems) from the cultural heritage domain has been done in collaboration with Marjolein van Gendt and Thomas Forrer along the line of the STITCH[1] project. Parts of the material of this chapter have been published in [98, 103, 75].

In this chapter we first discuss evaluation measures (9.1). Then, we present the test cases used for evaluation (9.2). Finally, we overview the matching systems used for evaluation (9.3).

[1]STITCH is funded by CATCH, a programme of the Netherlands Organization for Scientific Research NWO. See for details, http://www.cs.vu.nl/STITCH/

9.1 Evaluation measures

In order to evaluate the results of matching algorithms it is necessary to confront them with ontologies to be matched and to compare the alignment produced with a reference alignment based on some criteria.

This section is concerned with the question of how to measure the results returned by ontology matchers. It considers different possible measures for evaluating matching algorithms and systems. These include both effectiveness and efficiency measures.

9.1.1 Quality measures

The most prominent criteria are *precision* and *recall* originating from information retrieval [230] and adapted to ontology matching [57]. Precision and recall are based on the comparison of the resulting alignment A with a reference alignment R. These criteria are well understood and widely accepted.

Precision measures the ratio of correctly found correspondences (true positives) over the total number of returned correspondences (true positives and false positives), see Figure 9.1. In logical terms, precision is meant to measure the degree of correctness of the method.

Given a reference alignment R, the precision of some alignment A is a function $\mathcal{P} : \Lambda \times \Lambda \rightarrow [0\ 1]$, such that:

$$\mathcal{P}(A, R) = \frac{|R \cap A|}{|A|}.$$

Precision can also be determined without explicitly having a complete reference alignment. In this case only the correct alignments among the retrieved alignments have to be determined, namely $R \cap A$.

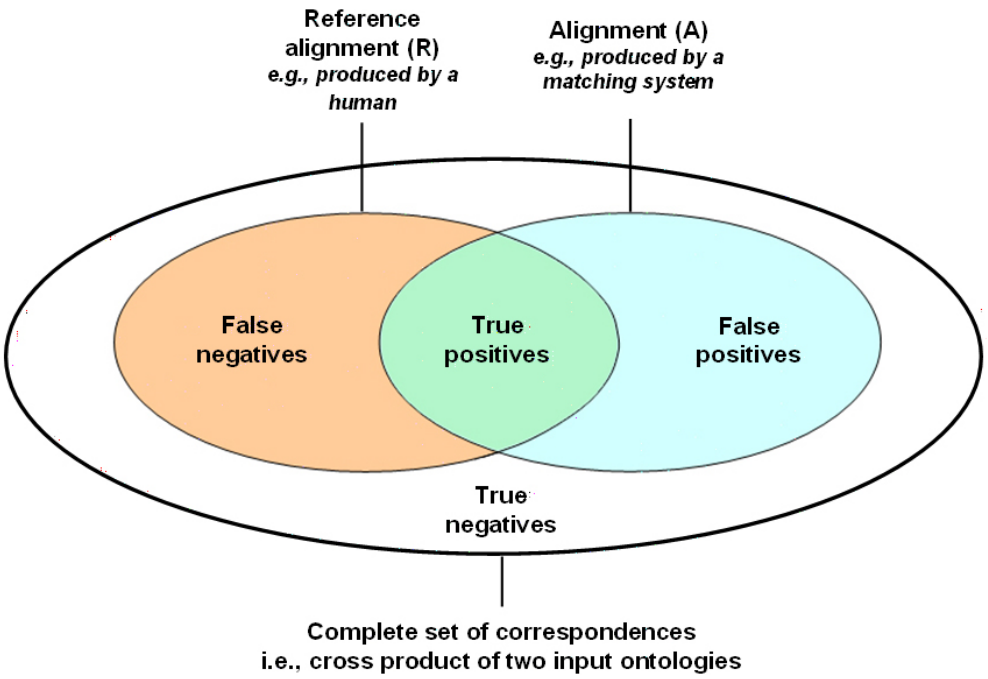


Figure 9.1: Two alignments as sets of correspondences and relations between them

Recall measures the ratio of correctly found correspondences (true positives) over the total number of expected correspondences (true positives and false negatives). In logical terms, recall is meant to measure the degree of completeness of the alignment.

Given a reference alignment R, the recall of some alignment A is a function $\mathcal{R} : \Lambda \times \Lambda \rightarrow [0\ 1]$, such that:

$$\mathcal{R}(A, R) = \frac{|R \cap A|}{|R|}.$$

Although precision and recall are the most widely and commonly used measures, when comparing systems one may prefer to have only a single measure. Moreover, systems are often not comparable based solely on precision and recall. The one which has higher recall may have a lower precision and vice versa. For this purpose, two measures are introduced

which aggregate precision and recall.

The F-measure is used in order to aggregate the results of precision and recall.

Given a reference alignment R and a number α between 0 and 1, the F-measure of some alignment A is a function $\mathcal{F}_\alpha : \Lambda \times \Lambda \rightarrow [0\ 1]$, such that:

$$\mathcal{F}_\alpha(A, R) = \frac{\mathcal{P}(A, R) \cdot \mathcal{R}(A, R)}{(1 - \alpha) \cdot \mathcal{P}(A, R) + \alpha \cdot \mathcal{R}(A, R)}.$$

If $\alpha = 1$, then the F-measure is equal to precision and if $\alpha = 0$, the F-measure is equal to recall. In between, the higher the value of α, the more importance is given to precision with regard to recall. Very often, the value $\alpha = 0.5$ is used, i.e.,

$$\mathcal{F}_{0.5}(A, R) = \frac{2 \cdot \mathcal{P}(A, R) \cdot \mathcal{R}(A, R)}{\mathcal{P}(A, R) + \mathcal{R}(A, R)},$$

which is the harmonic mean of precision and recall. It will be also used this way when computing the results of our experiments. This measure helps comparing systems by their precision and recall at the point where their F-measure is maximal.

The overall measure, also defined in [159] as matching accuracy, is the ratio of the number of errors on the size of the expected alignment. It is considered as an edit distance between an alignment and a reference alignment in which the only operation is "error correction". In this respect, it is considered as a measure of the effort required to fix the alignment. The overall is always lower than the F-measure and it ranges between $[-1\ 1]$. In fact, if precision is lower than 0.5, overall reaches a negative value, which can be interpreted that repairing the alignment is not worth the effort.

Given a reference alignment R, the overall measure of some alignment A is a function $\mathcal{O} : \Lambda \times \Lambda \rightarrow [-1\ 1]$, such that:

$$\mathcal{O}(A, R) = 1 - \frac{|(A \cup R) - (A \cap R)|}{|R|} = 1 - \frac{|R - A| + |A - R|}{|R|}.$$

9.1.2 Performance measures

Performance measures assess the resource consumption when matching ontologies. We mention some of these criteria below.

Unlike previously considered measures, performance measures depend on the processing environment and the underlying ontology management system. Thus, it is difficult to obtain objective evaluations, because they are based on the usual measures, namely processing time in seconds and memory in bytes. The important point here is that algorithms that are being compared should be run under the same conditions. We consider here two such measures.

Speed is measured by the amount of time taken by the algorithms for performing their matching tasks. It should be measured in the same conditions, i.e., same processor, same memory consumption, for all the systems. If user interaction is required, one has to ensure that only the processing time of the matching algorithm is measured.

The amount of *memory* used for performing the matching task marks another performance measure. Due to the dependency with underlying systems, it could also make sense to measure only the extra memory required in addition to that of the ontology management system, but it still remains highly dependent.

9.2 Test cases

The evaluation was performed on nine matching tasks from different application domains: a pair of catalogs (#1) and product schemas (#2), namely our running examples of Figure 5.1 (p.90) and Figure 5.5 (p.96), respectively. There are two matching tasks from a business domain (#3, 5). The first business example (#3) describes two company profiles: a Standard one and Yahoo Finance. The second business example (#5) deals with BizTalk purchase order schemas. There is one matching task from an academy domain (#4). It describes courses taught at Cornell University and at the University of Washington. There are three matching tasks on general topics (#6, 7, 8) as represented by the well-known web directories, such as Google, Yahoo, and Looksmart. Finally, the last matching task (#9) is from the cultural heritage domain. It deals with two standard thesauri used for storing masterpieces. Table 9.1 provides some indicators of the complexity of the test cases[2].

Table 9.1: Some indicators of the complexity of the test cases

#	**Matching task**	**#nodes**	**max depth**	**#labels per tree**
1	Images vs Europe (Figure 5.1, p.90)	4/5	2/2	6/5
2	Product schemas (Figure 5.5, p.96)	13/14	4/4	14/15
3	Yahoo Finance vs Standard	10/16	2/2	22/45
4	Cornell vs Washington	34/39	3/3	62/64
5	CIDX vs Excel	34/39	3/3	56/58
6	Google vs Looksmart	706/1081	11/16	1048/1715
7	Google vs Yahoo	561/665	11/11	722/945
8	Yahoo vs Looksmart	74/140	8/10	101/222
9	Iconclass vs Aria	999/553	9/3	2688/835

[2]Source files, description of the test cases, and reference alignments can be found at http://www.dit.unitn.it/∼accord/, experiments section.

As match quality measures we have used the following indicators: precision, recall, F-measure and overall (§9.1.1). In order to calculate the above mentioned quality indicators we had to obtain reference alignments. In particular, reference alignments have been manually produced with the help of BizTalk Mapper [209] used to visualize ontologies and correspondences created for the test cases #1, 2, 3, 4, 5 and 9. The size of the first five test cases is not big (dozens of nodes at most), therefore reference alignments can be produced relatively easily for them. The last test case (#9) is large, therefore producing manually reference alignment for it is time consuming and error-prone. We report our experience with building it next in §9.2.1. Finally, the test cases #6, 7, and 8 constitute the data set constructed in [9], where reference alignments have been acquired semi-automatically. This test case was used in the OAEI-2005 [77] ontology matching evaluation campaign.

9.2.1 Data set construction (#9)

We discuss our experience with building test case #9. It involves two large thesauri from the cultural heritage domain. These are *Iconclass*[3] and *Aria*[4]. The underlying documents of these thesauri are illuminated manuscripts and masterpieces. Note that this, from the matching algorithm perspective, forces to use only schema-based solutions, since instances are image data, and to the best of our knowledge at the moment there are no instance-based matching solutions working with image data. Alignment between these thesauri is ultimately used in the data integration scenario (§1.2.3), see for details [229].

The *Iconclass* thesaurus contains around 25.000 entities. One of its main purposes is an iconographical analysis. Therefore, the labels used

[3] http://www.iconclass.nl/libertas/ic?style=index.xsl

[4] http://www.rijksmuseum.nl/aria/aria_catalogs/index?lang=en

for classification purposes aim at providing precise descriptions of the underlying data. Note that these labels are gloss-like. They are much more complex than those we have considered in all the previous examples, since they have to describe what is depicted on a masterpiece. An example of a label from *Iconclass* is as follows: *city-view, and landscape with man-made constructions*.

The Rijksmuseum collection contains around 600 terms used to classify paintings and sculptures by means of the *Aria* thesaurus. Contrary to the case of *Iconclass*, labels used in *Aria* are short phrases and not gloss-like.

When building this data set we have focused only on a small part of *Iconclass* devoted to the subject of *nature* which considers *earth, and world as celestial body*. More precisely this part corresponds to the *Iconclass* fragment with index 25. We have considered the whole *Aria* thesaurus. Finally, we have cleaned the thesauri under consideration from non-English phrases, e.g., *Geen realtie met index* and *Fuoco, Carro del fuoco (Ripa)*. Table 9.1 (last row) summarizes the information about these thesauri which was ultimately used.

Our goal is to create a reference alignment between the fragment of *Iconclass* and *Aria* thesauri in order to enable evaluation of the quality of the results produced by matching systems. According to the application scenario [229], we are interested only in the *equivalence* (=), *more general* ($\sqsupseteq$) and *less general* ($\sqsubseteq$) relations. For example, correspondences with the intersection ($\sqcap$) relation have to be excluded.

The reference alignment has been produced manually. Whenever necessary, we have consulted actual data instances of the underlying information resources, namely illuminated manuscripts and masterpieces. Similar to the previously discussed test cases (#1-5) we used BizTalk Mapper [209] to visualize both thesauri and have an overview of the main themes they cover. Also, an initial set (several hundreds) of correspondences have been

manually created and visualized by using BizTalk Mapper. This approach helped us to obtain a first view over the scope of the task. For example, we have identified that in both thesauri there are parts devoted to the subject of landscapes. However, we could not complete this task by using BizTalk Mapper, since having created several hundreds of correspondences the visualization has become clumsy, see Figure 9.2. Similar observations have been also reported in [201].

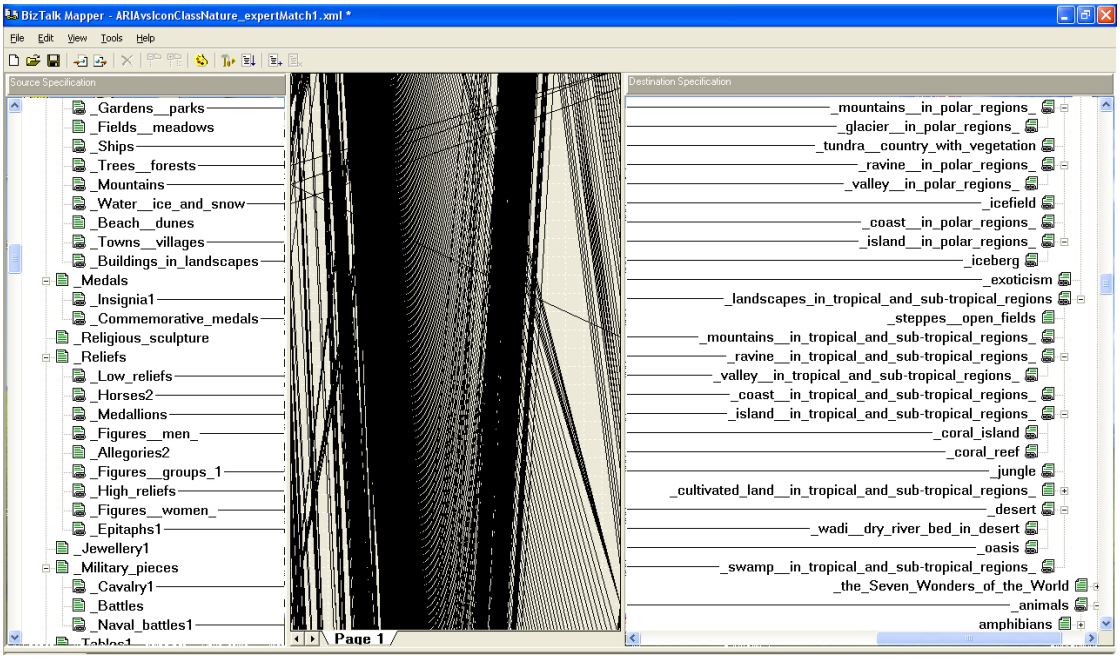


Figure 9.2: Manual matching with BizTalk Mapper

We proceeded with a plain text file to handle the correspondences. In order to ensure good enough quality of reference alignment (5% errors at most) we have split the set of all the correspondences into three parts. Let us discuss them in turn.

Part 1: *Certainly correct correspondences.* Some examples of the correspondences from this part include:

Aria: Top/Animal pieces/Birds
Iconclass: Top/Nature/earth, world as celestial body/animals

and

Aria: Top/Animal pieces/Birds
Iconclass: Top/Nature/earth, world as celestial body/animals/birds

In the first example above, the concept of *Birds* in *Aria* is more specific than *animals* in *Iconclass*, while in the second example there is the equivalence relation between the concepts under consideration.

Part 2: *Certainly incorrect correspondences.* An obvious example of the correspondence from this part is as follows:

Aria: Top/Holloware
Iconclass: Top/Nature/earth, world as celestial body/animals

Part 3: *Correspondences in the correctness of which we were uncertain.* Correspondences of this category have been analyzed one by one in order to reduce their number as much as possible and re-classify them in one of the other two categories, namely Part 1 or Part 2.
For example, the following correspondence has been initially put in Part 3 and having learned some more knowledge about jewelry, it was ultimately moved to Part 1:

Aria: Top/Accessories/Jewelry
Iconclass: Top/Nature/earth, world as celestial body/rock types; minerals and metals; soil types/rock types/precious and semiprecious stones/ precious and semiprecious stones (with NAME)/ precious and semiprecious stones: emerald

Another example, when the following correspondence has been initially put in Part 3 and having learned some more knowledge about jewelry, it was ultimately moved to Part 2:

Aria: Top/Accessories/Jewelry
Iconclass: Top/Nature/earth, world as celestial body/rock types; minerals and metals; soil types/rock types/precious and semiprecious stones/ precious and semiprecious stones (with NAME)/ precious and semiprecious stones: jasper

Note that the thesauri under consideration contain a lot of domain specific concepts that often only a domain expert can know their exact meaning. In the examples above, our initial knowledge about jewelry was not enough to assess whether *emerald* and *jasper* are actually *jewelry* or not. Finally, it is worth noting that sometimes we could not find any source of domain knowledge being precise enough to resolve our uncertainty. For example, should

Aria: Top/animal pieces/wild animals
or
Aria: Top/animal pieces/livestock
(or none ?) be matched to
Iconclass: Top/Nature/earth, world as celestial body/animals/ mammals/ hoofed animals/hoofed animals (with NAME)/hoofed animals: dromedary

Can we consider a dromedary to be fully domesticated? According to Wikipedia[5], dromedaries were domesticated between 4000 BC and 1400 BC and used for labour and dairy. Following this argument we can consider a dromedary to be a livestock. However, one can imagine an illuminated manuscript about dromedaries in wild life, thus being wild animals. Whether this question is valid or not, in our opinion, depends on the application, and, therefore, requires involvement of a domain expert to justify a correct decision. These kinds of correspondences remained in Part 3.

Table 9.2 provides a final size (number of correspondences) of each of the three parts. For the evaluation we have used only Part 1, namely the set of certainly (according to our knowledge) correct correspondences. Table 9.2 also shows that the reference alignment set is quite dense with respect to the size of the input thesauri.

	Part 1	**Part 2**	**Part 3**
#correspondences	1409	550918	120

Table 9.2: Final sizes of three parts of correspondences used for the data set construction

9.3 Systems used for evaluation

The system under a prime consideration is S-Match, which implements the ideas and algorithms presented in the previous chapters (as well as many other ideas and algorithms, for instance, a library of element level semantic matchers [101], the optimizations of the node matching algorithm [102], which will be the topic of another thesis). This system has been implemented by Mikalai Yatskevich, except parts for explanations (Chapter 8)

[5] http://en.wikipedia.org/wiki/Livestock

and iterative tree match algorithms (Chapter 7). Note that the implementation of the S-Match system is not claimed as a contribution of this thesis. The evaluation results obtained are intended to demonstrate that the ideas and algorithms developed in the thesis have been actually implemented and give proof of the concept that they are practically useful.

9.3.1 Setup for the comparative evaluation

We evaluate S-Match against three state of the art systems, namely Cupid [146] (§4.1.9), COMA [58][6] (§4.1.10), and Similarity Flooding [159] (§4.1.11) as implemented in Rondo [160]. All the systems under consideration are fairly comparable because they are all schema-based. They differ in the specific matching techniques they use and in how they compute alignments.

There are three further observations. The first observation is that Cupid, COMA, and Rondo can discover only the correspondences which express similarity between schema elements. Instead, S-Match, among the others, discovers the disjointness relation which can be interpreted as strong dissimilarity in terms of the other systems under consideration. Therefore, we did not take into account the disjointness relations (e.g., $\langle ID_{4,4}, C1_4, C2_4, \perp \rangle$ in Figure 5.5, p.96) when specifying the reference alignments. The second observation is that, since S-Match returns a matrix of relations, while all the other systems return a list of the best correspondences, we used some filtering rules. More precisely we have the following two rules: (i) discard all the correspondences where the relation is *idk*; (ii) return always the *core* relations, and discard relations whose existence is implied by the core relations. For the example of Figure 5.5 (p.96), $\langle ID_{3,3}, C1_3, C2_3, = \rangle$ should be returned, while $\langle ID_{3,5}, C1_3, C2_5, \sqsupseteq \rangle$

[6]We thank Phil Bernstein, Hong Hai Do, and Erhard Rahm for providing us with Cupid and COMA. In the evaluation we use the version of COMA described in [58]. A newer version of the system COMA++ exists but we do not have it.

should be discarded. Finally, whether S-Match returns the equivalence or subsumption relations does not affect the quality indicators. What only matters is the presence of the correspondence standing for those relations.

In our experiments each test has two degrees of freedom: *directionality* and *use of oracles*. By directionality we mean here the direction in which correspondences have been computed: from the first ontology to the second one (forward direction), or vice versa (backward direction). We report the best results obtained with respect to directionality, and use of oracles allowed. We were not able to plug a thesaurus in Rondo, since the version we have is standalone, and it does not support the use of external thesauri. Thesauri of S-Match, Cupid, and COMA were expanded with terms necessary for a fair competition (e.g., expanding *uom* into *unitOfMeasure*, a complete list is available at the URL in footnote 2, p.148).

For the comparative evaluation all the tests have been performed on a P4-1700, 512 MB of RAM, Windows XP, with no applications running but a single matching system. Also, all the tuning parameters (e.g., thresholds, strategies) of the systems were taken by default (e.g., for COMA we used *NamePath* and *Leaves* matchers combined in the *Average* strategy) for all the tests.

S-Match (non-iterative version) was run with five element level matchers, namely *WordNet*, *prefix*, *suffix*, *edit distance*, and *n-gram* [98, 101, 99] (see also §3.2.1) and SAT deciders of [131, 102] as structure level matchers which implement the semantic matching approach. String-based matchers were used with a threshold of 0.6 [101, 103].

Iterative S-Match used besides the five element level matchers mentioned above, also highly contextual element level matchers, namely: *Hierarchy Distance*, *WordNet Gloss*, *Extended WordNet Gloss*, *Gloss Comparison*, and *Extended Gloss Comparison* [101, 100] (see also §7.4).

9.3.2 Setup for the evaluation of explanations

The main goal of the experiments being conducted here is to obtain a vision of how explanations of semantic matching (Chapter 8) potentially scale to the requirements of the semantic web, providing meaningful and adjustable answers in real time.

The semantic (node) matching problem is a CO-NP hard problem, since it is reduced to the validity problem for the propositional calculus. Resolving this class of problems requires exponential time and exponentially long proof logs. However, in all the examples we have done so far proofs are not too long and seem of length polynomial in the length of the input clause. As a matter of fact, [102] shows, that when we have conjunctive concepts at nodes (e.g., $Images \wedge Europe$), these matching tasks can be resolved by the basic DPLL procedure in polynomial time; while when we have full proposition concepts at nodes (e.g., $Images \wedge (Computers \vee Internet)$), the length of the original formula can be exponentially reduced by structure preserving transformations.

In our experiments we have used three test cases, namely #1, 2, 3 of Table 9.1 (p.148). We focus on indicators characterizing explanations of mapping elements. The analysis of the quality of correspondences is beyond scope of this experiment. In the experimental study we have used the following indicators:

- Number of mapping elements determined for a pair of ontologies. As follows from the definition of semantic matching, this number should be N1 $\times$ N2, where N1 is the number of nodes in the first ontology, N2 is the number of nodes in the second ontology.

- Number of steps in a proof of a single mapping element. This indicator represents the number of PML node sets are to be created in the proof.

- Time needed to produce a proof of a single mapping element. This indicator estimates how fast the modified JSAT/SAT4J in producing IW proofs for a particular task.
- Time needed to produce a proof of all mapping elements determined by S-Match for a pair of ontologies.

In order to conduct tests in a real environment, we used the Inference Web web service of KSL at Stanford University (on a P4-2.8GHz, 1.5Gb of RAM, Linux, Tomcat web server) to generate proofs in PML, while the modified JSAT/SAT4J version was run at the University of Trento (on a P4-1.7GHz, 256 MB of RAM, Windows XP). All the tests were performed without any optimizations: for each single task submitted to JSAT/SAT4J, the IW web service was invoked, no compression methods were used while transferring files, etc.

9.4 Summary

In this chapter we have discussed some of the ontology matching evaluation criteria. In particular, we have presented quality and performance measures, test cases as well as the systems which were used for the evaluation.

We also described our experience with building manually a large data set from the cultural heritage domain for the quality evaluation of the results produced by matching systems. This data set is of high importance for the ontology matching evaluation due to the general lack of large data sets (containing hundreds and thousands of entities) allowing the measurement of the quality indicators of matching systems.

Chapter 10

Evaluation results

This chapter provides evaluation results for the test cases and systems introduced previously in §9.2 and §9.3, respectively. The results have been obtained and compared based on the measures outlined in §9.1.

Material presented in this chapter has been developed in collaboration with Mikalai Yatskevich and Paulo Pinheiro da Silva and published in [98, 215, 99, 100, 103].

In this chapter we first discuss evaluation results for the semantic matching (§10.1) and iterative semantic matching approaches (§10.2). Then, we provide evaluation results for the explanations of the semantic matching (§10.3). Finally, we briefly overview some lessons learned out of the experiments (§10.4).

10.1 Evaluation of semantic matching

We present the quality results for the tasks of Table 9.1 (p.148). For the matching tasks #2, 3, 4, 5 these are shown in Figures 10.1, 10.2, 10.3, and 10.4, respectively.

For example, in Figures 10.1 and 10.3, since all the labels at nodes

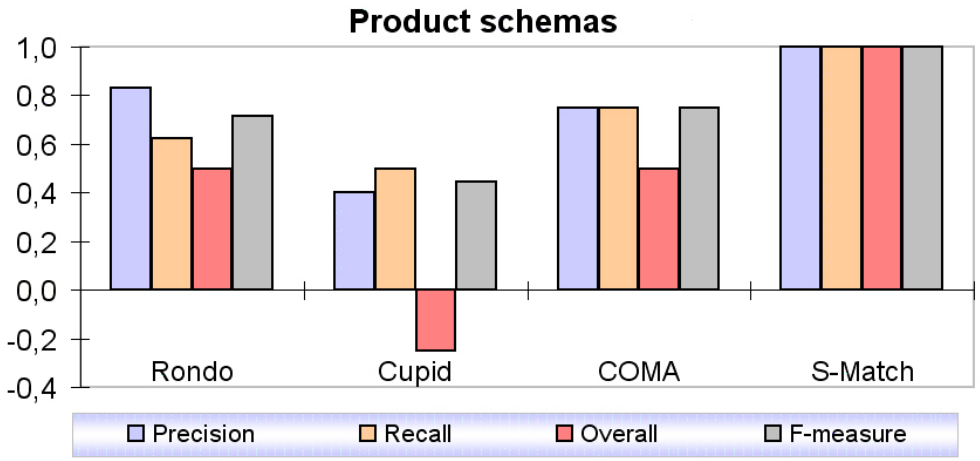


Figure 10.1: Evaluation results: Product schemas (Figure 5.5), test case #2

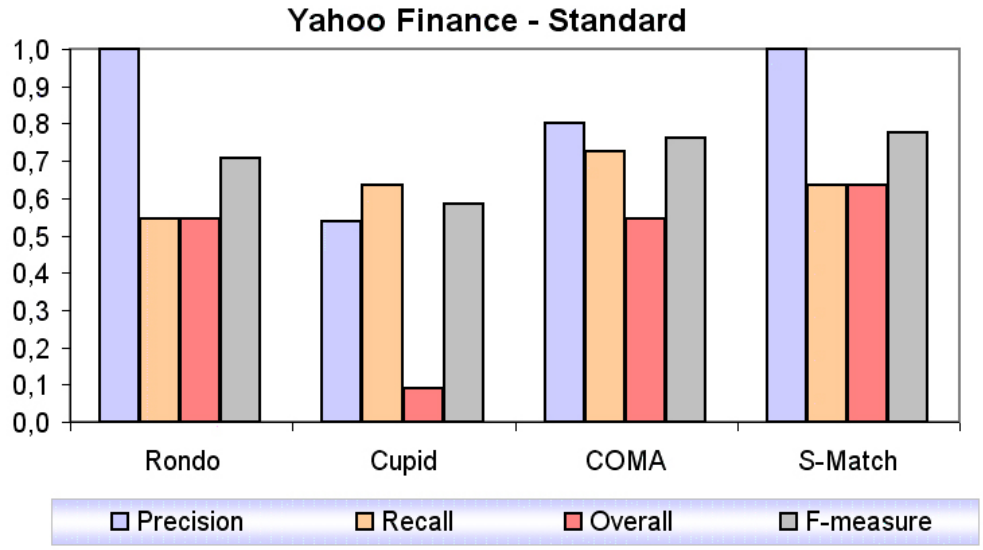


Figure 10.2: Evaluation results: Yahoo Finance vs Standard, test case #3

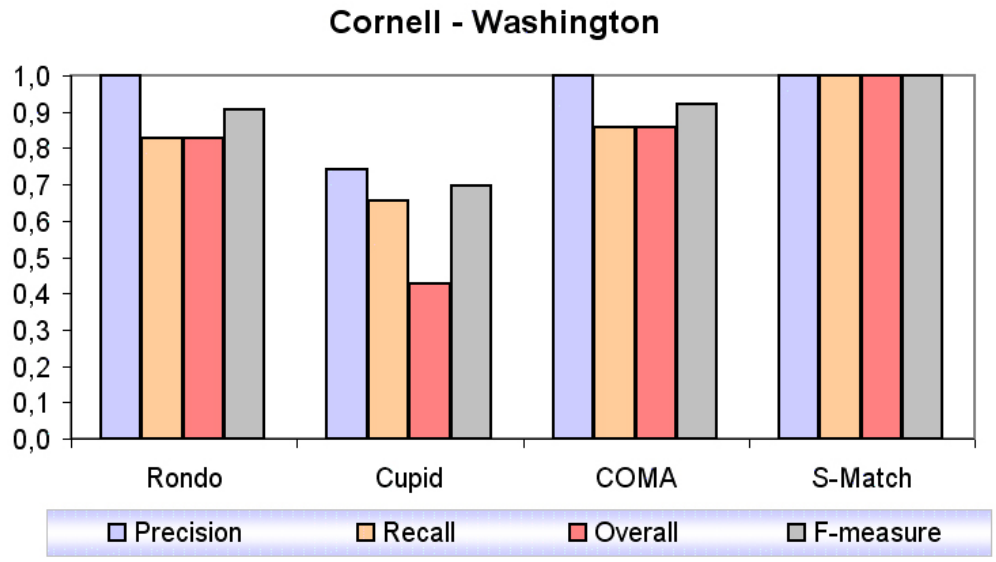


Figure 10.3: Evaluation results: Cornell vs Washington, test case #4

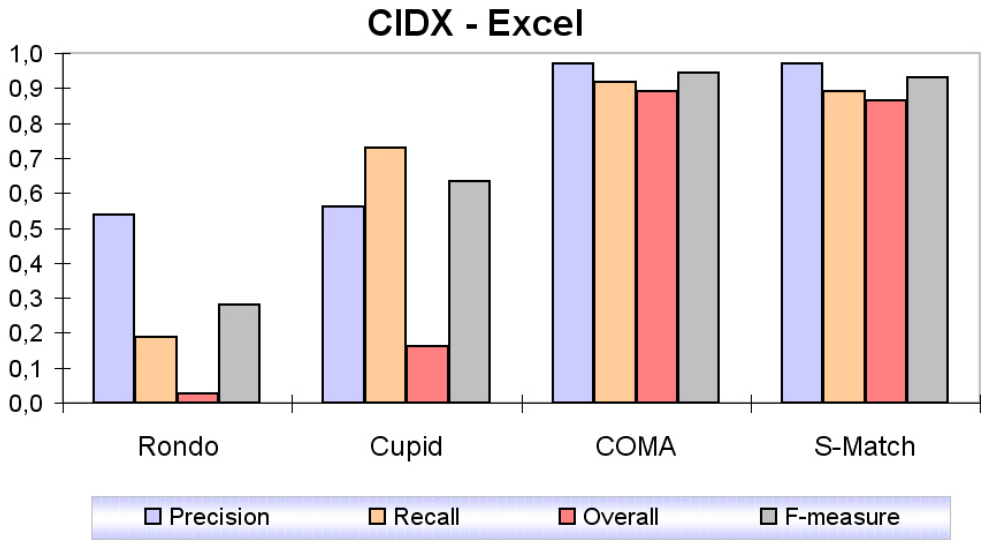


Figure 10.4: Evaluation results: CIDX vs Excel, test case #5

in the given test case were correctly encoded into propositional formulas, all the quality measures of S-Match reach their highest values. In fact, as discussed before, the propositional SAT solver is correct and complete. This means that once the element level matchers have found all and only the mapping elements, S-Match will return all of them and only the correct ones.

For pairs of business schemas, namely Yahoo Finance vs Standard and CIDX vs Excel (Figures 10.2 and Figures 10.4, respectively), S-Match performs as good as COMA and outperforms other systems in terms of quality indicators.

10.2 Evaluation of iterative semantic matching

Iterative semantic matching algorithm has been evaluated on the tasks #6, 7, 8, and 9 of Table 9.1 (p.148).

10.2.1 Evaluation results for the web directories task (#6,7,8)

As reference alignments for the tasks #6, 7, 8 we used 2265 mapping elements acquired in [9]. By construction those reference alignments represent only true positives, thereby allowing us to estimate only the recall with them. To the best of our knowledge, at the moment, there are no large data sets (besides #9) where available reference alignment allows measuring both precision and recall. Thus, in the following for the tasks #6, 7, 8 we focus mostly on analyzing the recall.

Two further observations. First, as it was already mentioned in §9.1, higher values of recall can be obtained at the expense (lower values) of precision. Thus, in order to ensure a *fair* recall evaluation, before running tests on the matching tasks #6, 7, 8, we have analyzed behavior of the

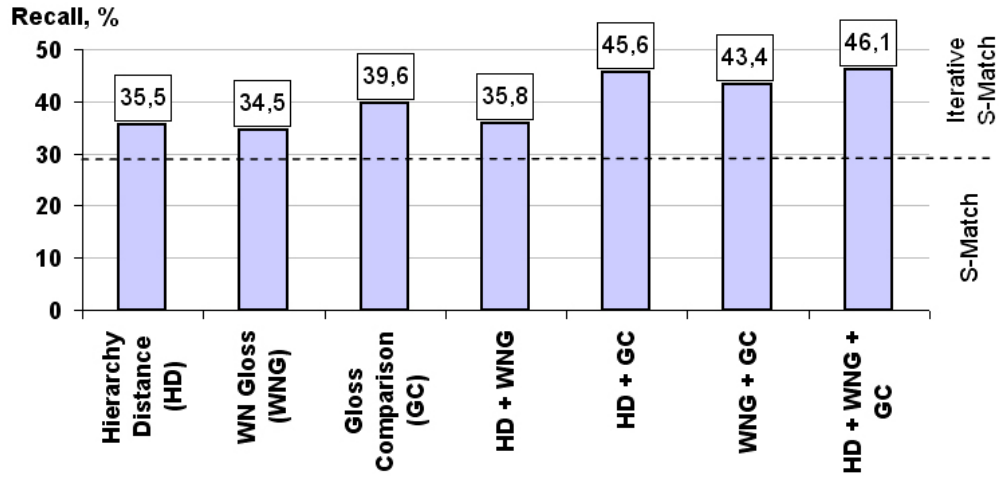


Figure 10.5: Evaluation results (absolute values), test cases #6,7,8

iterative semantic matching on the other test cases of Table 9.1 where reference alignments allowed measuring both precision and recall. Matchers decreasing precision substantially in these tests were discarded from the further evaluation. In fact, for this reason we exclude from the further considerations the *Extended Gloss Comparison* matcher. The second observation is that using matchers mentioned in §9.3 exhaustively for all the tasks, hence, omitting the critical points discovery algorithm, also leads to a significant precision decrease, thus justifying usefulness of the cPoints-Discovery algorithm (p.122).

The summarized evaluation results for the matching tasks #6, 7, 8 of Table 9.1 are shown in Figure 10.5. In particular, it demonstrates contributions to the recall of matchers mentioned in §9.3 as well as of their combinations. The *Extended WordNet Gloss* matcher performed very poorly, i.e., contributing less than 1% to the recall, hence, we do not report its results in Figure 10.5. By using a combination of the *Hierarchy Distance*, *WordNet Gloss*, and *Gloss Comparison* matchers we have improved S-Match recall results (29,5%) up to 46,1% within the iterative S-Match[1].

[1]Note that this result should be considered as a complimentary one to the results of S-Match++ reported in [9], since they address separate problem spaces.

Table 10.1: Some element level matchers used in the iterative semantic matching and their evaluation results

	HD	*GC*	*HD + GC*	*HD + WNG + GC*
Recall increase (relative), %	20	34	54	56
Threshold value	4	2	$4 \setminus 2$	$4 \setminus 1 \setminus 2$

Let us now consider *relative* characteristics of the iterative S-Match with respect to the non-iterative version, see the first row of Table 10.1 for a summary. The highest recall increase by using only a single matcher out of those mentioned in §9.3 within the iterative S-Match was achieved by the *Gloss Comparison* matcher, namely 34% over the non-iterative S-Match. The best, in this sense, combination of two matchers is being that of the *Hierarchy Distance* and *Gloss Comparison* matchers: recall increased by 54%. Finally, a combination of the *Hierarchy Distance*, *WordNet Gloss* and *Gloss Comparison* matchers resulted in the 56% recall increase with respect to the non-iterative S-Match.

Table 10.1 also reports values of thresholds used within the evaluation. These values were obtained based on the rationale behind designing matchers mentioned in §9.3 and their evaluation results.

The evaluation we have conducted shows that the problem of the lack of background knowledge is a hard one. In fact, as it turns out, not all the designed element level matchers can perform always well in real world applications, as it might (mistakenly) seem from the toy evaluations. Also, new matchers are still needed, since, for example, we could discover that $\langle C1_4, C2_4 \rangle$ in Figure 7.2 (p.118) is the critical point, however, we were unable to resolve it with the matchers mentioned in §9.3, namely to match $Home_1$ and $Hobbies\_AND\_Interests_2$ in Figure 7.2.

10.2.2 Evaluation results for the cultural heritage task (#9)

To study the behavior of semantic matching and iterative semantic matching on the data set from the cultural heritage domain (§9.2.1), we have made only a preliminary evaluation. We ran S-Match and Iterative S-Match in default configurations (§9.3). The summary of the evaluation results is presented in Table 10.2.

	Precision, %	**Recall, %**	**F-measure, %**
S-Match	44.82	6.45	11.29
Iterative S-Match	47.69	6.60	11.59

Table 10.2: Preliminary evaluation results: Iconclass vs Aria, test case #9

The results of Table 10.2 show that the task is indeed hard and challenging. Very low recall results can be explained by the fact that labels of *Iconclass* are gloss-like, while the algorithms were instead expecting labels built by only short phrases, like in all the previous test cases.

10.3 Evaluation of explanations

Figure 10.6 reports on the results of the experimental study. In particular, for each mapping element of the three test cases, it represents the number of proof steps required and the time needed to generate proofs in PML. Notice, that the proof time indicator in Figure 10.6 takes into account the time needed by the modified version of JSAT/SAT4J to produce proof information, connection time to the IW web service, time for producing and posting PML documents.

An observation of the spikes starting from the mapping element #700 in the time line of the Cornell vs Washington test case is an example of how Internet connection increases the proof time. The average proof length and

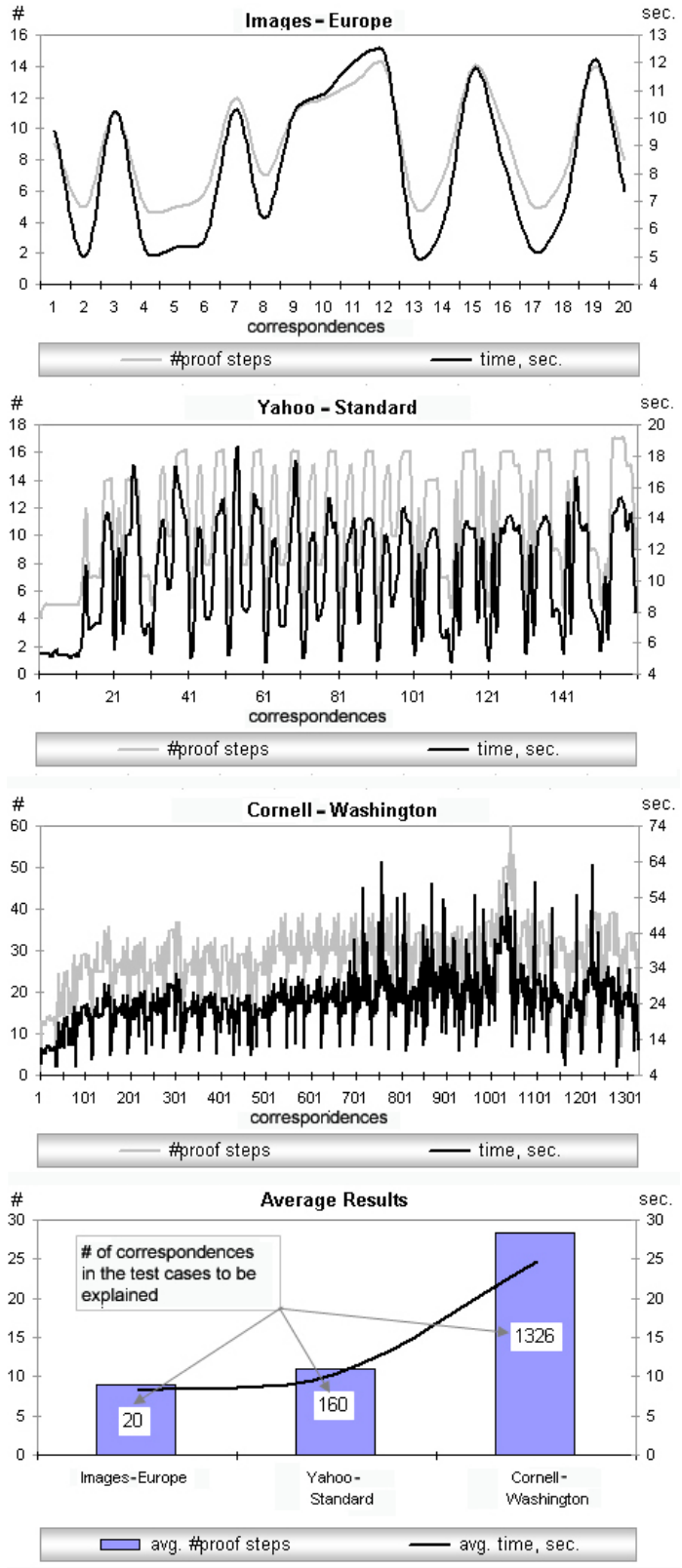


Figure 10.6: Experimental results for explanations, test cases #1,2,3

proof time for a single mapping element in the test cases of Figure 10.6 constitute 16 steps and 14 seconds. Time needed to produce proofs of all mapping elements in each test case is $2.7min.$ - 20 mapping elements; $27.7min.$ - 160 mapping elements; and $546.2min.$ - 1326 mapping elements, respectively. Notice that the modified JSAT/SAT4J version produces proof information on a single mapping element requiring, in the average, less than 1 millisecond, therefore producing proof information for all mapping elements, for instance, in the case of 1326 mappings, would require less than 1 minute. Moreover, it is hard to imagine that ordinary users will be willing to browse explanations of thousands and even hundreds of mapping elements. However, one dozen seems to be a reasonable number of mapping elements to be looked through for a short period of time.

10.4 Lessons learned

10.4.1 Evaluation of quality of the results of matching systems

Our evaluation has shown that even if matching systems can achieve good quality results on small ontologies, the situation is far from being that promising in the case of large ontologies. Also when labels are gloss-like, special techniques have to be developed to handle them. For example, we have analyzed why the recall on the data set from the cultural heritage domain was so low. After some study on the mapping elements we have find out the following general classes of mistakes done by S-Match. We illustrate them below with the help of examples.

Recognizing "noisy" labels. When working with gloss-like labels, the system tries to interpret all the labels defining an entity. However, some labels can represent "noise" for the matching algorithm. For example, given the entity *main subject: animals*, the system tries

to interpret all the labels and build a concept of it, however it is not really necessary. The labels *main* and *subject* do not technically contribute much to the meaning of the entity under consideration (which defines animals). As a consequence, the system cannot match, for example, *main subject: animals* with *animals*, because, *main* and *subject* are not related (in WordNet) to *animals*. Thus, such labels that give no or little contribution to the meaning of an entity can be considered as "noise" from the matching algorithm perspective. A proper recognition and treatment (e.g., elimination) of the "noisy" labels is thus needed.

Negation. When working with negations in gloss-like labels, the algorithm even if it interprets correctly, e.g., *other than* to be a negation, it applies it only to the next atomic concept. For example, given the label *seasons of the year represented by concepts other than personifications, human activities, landscapes or still lifes of flowers and or fruits e.g., biblical scenes*, the algorithm negates only the first atomic concept after the negation phrase, namely *personifications*. However, the intended meaning of the sentence is to negate also *human activities*, *landscapes*, etc. Thus, the system has to be improved in order to understand where to put the parentheses and negate all the necessary atomic concepts of labels. This problem appears only when dealing with gloss-like labels, since when labels are short phrases, negation usually appears to negate only one concept, e.g., *except landscapes*.

10.4.2 Evaluation of the explanations of semantic matching

Results of the experimental study of §10.3 look promising, however there are proof time issues to be addressed. For example, if a user needs explanations aimed at proof generation and manipulation need to be added. How-

ever, the experimental study we have conducted gives a preliminary vision that the explanation techniques proposed potentially scale to requirements of the semantic web, providing meaningful and adjustable answers in real time.

10.5 Summary

In this chapter we have presented comparative evaluation of semantic matching and iterative semantic matching against the other state of the art systems. We also discussed evaluation of explanations for semantic matching. The results are encouraging and empirically prove the strength of our approach.

However, as our evaluation results show, it is very difficult to know a priori the quality to expect from a matching system. Matching tasks are so different that a system can perform very well on some data and not that well on some other. This means that in order to justify the claim of a matching system to be generic, a lot of work has to be done yet, especially to address all the issues that arise when dealing with large-scale matching tasks. However, still it is necessary that evaluation data sets be as different as possible and that results be kept separate so that someone with a particular task can choose a system that performs adequately on this task.

Part V

Conclusions

Chapter 11

Summary

In this thesis we have provided a detailed account of the state of the art in ontology matching. We proposed a novel approach to ontology matching, called semantic matching, discussed its technical details and some evaluation. Specifically, the main findings of each chapter of the thesis are summarized one by one in sequel. Finally, future trends in the matching field are outlined in (the next and last) Chapter 12.

We showed that there are many applications that may need ontology matching (Chapter 1). This was the reason to consider ontology matching as a unified object of study. However, there are notable variations in the way these applications use matching. Therefore, we identified some application related differences which have to be taken into account in order to provide the best suited solution in each case.

We showed that there are various existing ways of expressing knowledge that are found in diverse applications. These ways of expressing knowledge can be viewed as different forms of ontologies that may need to be matched (Chapter 2). Unlike many other works, we aimed to treat the matching problem in a unified way and provide a common roof under the heading of ontology matching for many existing instantiations of this problem, such as schema matching, catalog matching, etc. The reason is to facilitate the

cross-fertilization. In fact, on the one side, for example, schema matching is usually performed with the help of techniques trying to guess the meaning encoded in the schemas. On the other side, ontology matching systems primarily try to exploit knowledge explicitly encoded in the ontologies. In real world applications, schemas and ontologies usually have both well defined and obscure terms, and the contexts in which they occur, therefore, solutions from both problems would be mutually beneficial. We introduced several justifications for heterogeneity in order to help the design of a matching strategy as a function of the kind of heterogeneity that has to be addressed. Finally, we technically defined the ontology matching problem.

We showed that ontology matching can take advantage of innumerable basic techniques composed and supervised in diverse ways (Chapter 3). We provided a systematic view over the available techniques by classifying them and providing some guidelines which help in identifying families of matching methods.

We reviewed existing schema-based matching systems which emerged during the last decade (Chapter 4). These were presented in light of the classifications developed in Chapter 3. We also pointed to concrete basic matcher and matching strategies used in the considered systems. We summarized some global observations concerning the presented systems and outlined a number of constant features that are shared by the majority of them.

Having analyzed in detail the state of the art we proposed an approach to ontology matching called semantic matching (Chapter 5). This has been done based on what we have found good practices in the previous approaches and what we have found missing in them, thereby mastering that gap. We discussed with the help of examples and pseudo-code the main macro steps of the algorithm that implements the semantic matching

approach.

We showed how attributes are handled within the semantic matching settings (Chapter 6). We argued that a plausible way to match attributes using the semantic matching approach is to discard the information about datatypes.

We demonstrated how to deal in a fully automated way with the lack of background knowledge in matching tasks by using semantic matching iteratively (Chapter 7). This helps saving some of the pre-match efforts, improving the quality of match via iterations, and enabling the future reuse of the newly discovered knowledge.

By extending semantic matching to use the Inference Web infrastructure, we demonstrated our approach for explaining answers from matching systems exploiting background ontological information and reasoning engines (Chapter 8). Delivering alignments to users, for inspection and revision, is an important topic not deeply developed so far in the ontology matching community.

We discussed some evaluation criteria for comparison of the results of matching algorithms (Chapter 9). We described our experience with building a large test case for the evaluation of quality results produced by matching systems. It is worth noting that this is a time-consuming and error-prone effort, however, large real world data sets for evaluation of the quality of matching results is among important and not well developed themes of ontology matching.

We performed an evaluation of the semantic matching approach, which gives proof of the concept, that it is practically useful (Chapter 10). As our comparative evaluation shows it is very difficult to know a priori the quality to expect from a matching system. Matching tasks are so different that a system can perform very well on some, usually small test cases, while not that well on some other, usually large-scale test cases. Analysis

of the mistakes done by a system opens a number of ways for further improvements.

We would like to make two final remarks. The first remark concerns some assumptions and limitations of the proposed solution. In particular, the proposed solution naturally assumes that the ontologies to be matched have a meaningful overlap, thus these are worth been matched. The proposed approach reduces the conceptual heterogeneity (see p.27) only to a certain extent, though, for example, cases such as geometry axiomatized with points as primitive objects and geometry axiomatized with spheres as primitive objects are not handled. At last, although we have aimed at producing a generic matching solution, a lot of work still needs to be done. For example, as §10.2.2 (p.165) shows, additional techniques have to be developed in order to handle properly gloss-like labels. Also we have only investigated matching of tree-like structures produced out of classifications and catalogs, while it has still to be analyzed whether the presented solution will properly handle the trees generated, e.g., out of relational schemas and the other forms of ontologies.

The second remark is to point out that although the semantic heterogeneity problem has been known and faced for decades, the ontology matching, which is a plausible solution to it, by the time the work on this thesis started, i.e., in 2002, had still been in its infancy. Therefore, besides the development of the semantic matching approach, many efforts have been invested in understanding the related to ontology matching problems and areas as well as in the rationalization of the state of the art. As a result, an extended and updated version of the general part of this thesis will appear in the first book on the topic of ontology matching [75].

Chapter 12

Future trends in the field

Anticipating the technical details of future trends in ontology matching, let us first focus on two general observations. The first observation is that most of the work on matching has been carried out among (*i*) database schemas in the world of information integration, (*ii*) XML-schemas and catalogs on the web, (*iii*) formal ontologies in artificial intelligence, semantic web, knowledge representation, and (*iv*) objects and entities in data mining. In the past, these communities were, in a sense isolated, and rarely addressed technical issues they had encountered from the multidisciplinary and cross-community viewpoints. Also, it is worth noting that during the last decade these areas have done a substantial progress in matching. However, they require other technologies to continue their growth. Thus, there has emerged such an initiative as Ontology Matching[1], which aims at increasing awareness of the existing matching efforts across the relevant communities and facilitating the cross-fertilization between them.

The second observation is that the number and variety of solutions to the matching problem keep growing at a fast pace. In particular, Figure 12.1 shows (approximately) how many works devoted to diverse aspects

[1] http://www.OntologyMatching.org

of matching have been published at various conferences all over the world in the recent years[2].

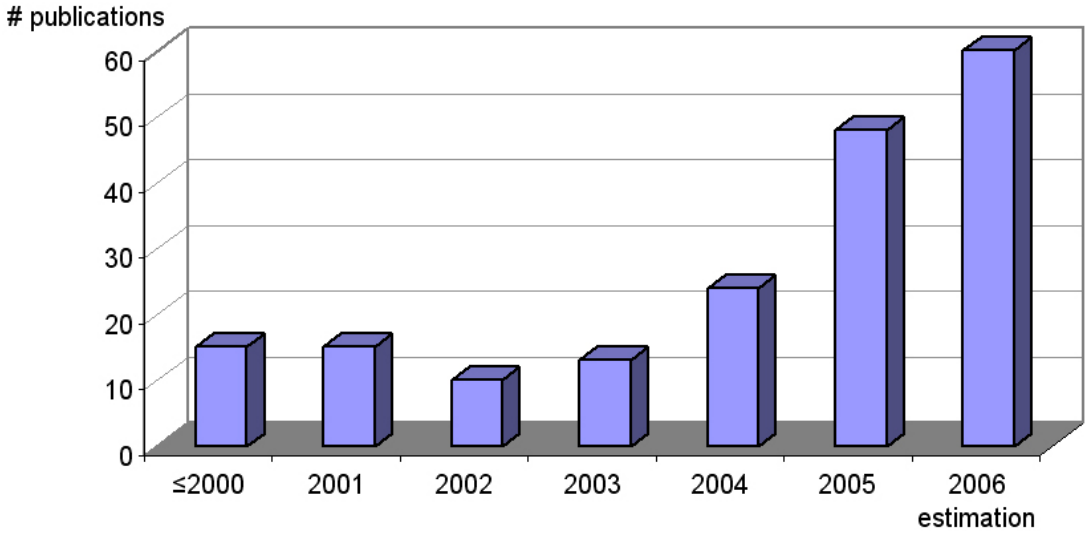


Figure 12.1: Dynamics of publications devoted to matching

In the future, we expect a continuing growth of works on matching due to the constantly increasing interest in intelligent solutions for semantic heterogeneity problem from both academia and industry.

Material presented in this chapter has been developed in collaboration with Jérôme Euzenat and Roberta Cuel and published in [75]. The work on the topic of this chapter has been supported by the FP6 Knowledge Web[3] Network of Excellence.

The rest of the chapter is organized as follows. Future trends are discussed along the lines of (*i*) trends in theories and methods (§12.1), (*ii*) trends in tools (§12.2), and (*iii*) trends in applications (§12.3). In turn, each of the three parts is detailed according to the trends of short (0-3 years), medium (3-6 years), and long (6-12 years) terms.

[2]Source: www.OntologyMatching.org, Publications section. Estimation for 2006 is based on interpolation from the three first quarters of this year.

[3]`http://knowledgeweb.semanticweb.org/`

12.1 Trends in theories and methods

Heterogeneity is typically reduced in two steps: (*i*) match two ontologies, thereby determining the alignment and (*ii*) process the alignment according to an application needs (e.g., query answering, web service composition). In this thesis we have focused only on the first step, and, therefore, here we consider the future trends only from this perspective. In particular, we discuss the future trends in matching approaches following the dimensions identified in Chapter 3, namely: (*i*) the input of the algorithms, (*ii*) the characteristics of the matching process, and (*iii*) the output of the algorithms. Finally, we discuss possible trends in the evaluation of the matching approaches.

Disregarding the timelines, there are some general trends to be mentioned, namely: gradual and incremental improvement of the existing approaches, emergence of the new approaches by modifying exiting ones (usually performed by different group(s) of people with respect to the original approaches), and emergence of the completely new approaches.

Also, notice that trends which are discussed in the short (medium) term, in general, remain valid for the forthcoming periods, though, their perfection is expected.

12.1.1 Short term

Matching approaches

Input dimensions. These dimensions concern the kind of input on which algorithms operate. We expect the following short term trends here:

- Most of the approaches tend to be more and more generic, i.e., handle multiple forms of ontologies;

- New types of input, such as plain text and query interfaces from the *deep web* [20, 117] should enter intensively into practice;
- Approaches will try to suitably handle more and more constructs available from the input (e.g., constraints).

Process dimensions. We discuss the expected trends first in basic matchers, then in matching strategies, and finally, generally, in matching approaches. Thus, the expected short term trends are:

- New types of basic automatic matchers addressing a larger variety and more sophisticated situations with respect to the current state of affairs. Some possibly emerging examples include:
  - Methods for matching glosses (comments) against labels of entities;
  - Methods for matching processes;
  - Methods for alignment reuse (e.g., by reasoning with the given correspondences to deduce the new correspondences, verify if the correspondences are still correct, and repair them if necessary);
  - Methods exploiting various (new) external resources, e.g., upper level ontologies, such as DOLCE [88], domain specific corpuses [144];
  - Approximate (e.g., semantic-based) methods.
- New libraries of matchers (or extensions of the existing libraries), which group together the basic automatic matchers based on their common characteristics, e.g., name-based matchers.
- New approaches to automate the combination of individual matchers and libraries of matchers. Some existing solutions here can be found in [59, 67]. Some possibly emerging examples are:

  - Methods for learning the optimal weight assignments, given a set of basic matchers;
  - Combining different techniques (e.g., collaborative filtering, genetic algorithms, statistics) for the optimal/near optimal weight assignments.

- New general matching solutions or default combinations of basic matchers which prove themselves equally good for most of the tasks.
- New approaches to tune automatically matching solutions in general (e.g., thresholds, weights, coefficients, which basic matchers to use). Existing examples are given in [205, 67].
- Various application specific approaches, which are particularly tailored to the input/output characteristics.
- New matching approaches investigating the quality vs. efficiency trade off.
- New ways of viewing/resolving the matching problem by reducing it to the other, already known problem. Some existing examples of these translations are graph matching [159, 78], propositional validity [30, 97], and probabilistic inference [167, 190].

Output dimensions. We expect the following short term trends: translations between alignments specified with the help of coefficients in [0 1] range and logical relations, expressiveness of alignment (atomic vs complex), language(s) for alignment (some existing examples include C-OWL [34], SWRL [118], Alignment format [73], see [208, 75] for an overview), formal semantics of alignment, scalability of alignment, framework(s) for characterizing the alignment, and application specific alignment.

Evaluation of matching approaches

We expect the following trends in evaluation of matching approaches in the short term: continues (at least annual) ontology matching contests[4], improvements of the ontology matching evaluation methodology, new data set construction methodologies, including new large real world data sets, new systematic (artificial) test (e.g., robustness to data noises), new quality measures, including combinations of precision and recall, and application specific measures.

12.1.2 Medium term

We discuss some challenges, which we believe, in the medium term (not earlier) will find appropriate solutions.

Matching approaches

Input dimensions. We expect emergence of standard(s) for the internal representations of different forms of ontologies taken as input by matching approaches.

Process dimensions. The key challenges include:

- *Knowledge incompleteness.* Recent industrial-strength evaluations of matching systems, see, e.g., [77, 9, 74], show that lack of background knowledge, most often domain specific knowledge, is one of the key problems of matching systems. In fact, most state of the art systems, for the tasks of matching thousands of entities, perform not with such high values of *recall* (∼30%) as in cases of *toy* examples, where the

[4]Matching contests of years 2004, 2005 and 2006 can be found following the links below:
2004: http://www.atl.external.lmco.com/projects/ontology/i3con.html,
2004: http://oaei.ontologymatching.org/2004/Contest/,
2005: http://oaei.ontologymatching.org/2005/,
2006: http://oaei.ontologymatching.org/2006/.

recall was most often around 80-90% [100]. Thus, we expect emergence of the frameworks leveraging the knowledge incompleteness problem, ultimately in a fully automated way. One possible solution has been proposed in this thesis.

- *Performance.* Following the above mentioned examples of the large scale evaluations, besides the effectiveness of the results, there is an issue of performance. In fact, there are applications which require at least some weak form of real time performance (to avoid having a user waiting too long for the system respond). *Execution time* indicator shows scalability properties of the matchers and their potential to become an industrial-strength systems. Also, referring to the above mentioned evaluations, the fact that some systems went out of memory on some test cases, although being fast on small and medium test cases, suggests that their performance time was achieved by using a large amount of main memory. Therefore, usage of *main memory* should also be taken into account. We expect significant improvements of the matching approaches with respect to their performance characteristics.

- *Interactive approaches (semi-automatic matching).* As from above, automatic ontology matching usually cannot be performed with a due quality, especially on the huge data sets. We believe that semi-automatic matching is a plausible way to improve the effectiveness of the results. There are tasks at which machines are good. Obviously, there are tasks at which human users are good. An important point here is to involve user only when his/her input is maximally useful.

- *Explanations and transparency.* Correspondences produced by matching systems may not be intuitively obvious to human users, and therefore, they need to be explained, see [215, 53, 129]. One possible so-

lution of how a matching system can explain its answers has been proposed in this thesis.

- *Social aspects.* The impact of social networks, web communities and direct involvement of humans (in a distributed fashion) on ontology matching has to be analyzed and distilled. Let us consider one example. Eventually, once an alignment has been determined, it can be saved, and further reused as any other data on the web [238]. Thus, on the one hand, a (large) repository of alignments has a potential to increase the effectiveness of matching systems by providing yet another source of domain specific knowledge. On the other hand, users can publish different and even contradicting alignments. Hence, one of the open problems here is how to manage the contradictory correspondences in the repositories.

Output dimensions. We expect emergence of annotations (codifying social aspects) of the alignment and standard(s) for expressing the alignment.

Evaluation of matching approaches

We expect the following trends in evaluation of matching approaches in the medium term:

- Extensive experiments across different domains with multiple test cases from each domain as well as new hard, and large real world data sets.
- More accurate evaluation measures, including user-related measures.
- Automating acquisition of reference alignments, especially for large applications.

12.1.3 Long term

Matching approaches

In the long term we expect appearance of multilingual matching approaches, i.e., those matching across multiple languages, such as English, Italian, and French. Also we expect appearance of matching approaches dealing with spatio-temporal applications [5, 194]. Finally, a substantial progress in the field should have been done by that time in general, which in turn, should cause some paradigm shifts. Thus, new visions and requirements of what is matching should appear.

Evaluation of matching approaches

Addressing the first two points mentioned above, we expect the following trends in evaluation of matching approaches in the long term: evaluation methodology for multilingual and spatio-temporal matching approaches, multilingual and spatio-temporal data sets, quality measures for multilingual and spatio-temporal matching approaches.

12.2 Trends in tools

12.2.1 Short term

We discuss the future trends in tools, distinguishing between (relevant) commercially available ones and research prototypes. Most of the commercially available matching tools focus on visualization of the input ontologies expressed in, e.g., XML, database, flat files formats, and the correspondences between them. It is also possible to specify (over the correspondences) some data transformation operations (e.g., by means of functoids) such as adding, multiplying, and dividing the values of fields in the source document and storing the result in a field in the target document. How-

ever, the matching operation itself is not automated at all, namely all the correspondences have to be specified manually. Some examples of these tools are Altova MapForce[5], BizTalk Schema Mapper[6], Cape Clear XSLT Mapper[7], Stylus Studio XSLT Mapper[8]. In the short term we expect an increase of the number of such tools.

Obviously, contrary to the commercial tools, research matching prototypes focus on automating the correspondences discovery operation and related themes. In general, majority of the research tools focus only on one of the steps of reducing the heterogeneity, namely on matching ontologies, fewer on processing the alignments, and only some of them can be called infrastructures, since they consider matching as one (among others) operations. It is early to speak about software quality in research tools. However, some positive trends are worth mentioning, such as modularity and extensibility of the architectures in most of the research prototypes. We expect gradual and incremental improvements along the lines mentioned above in the short term.

12.2.2 Medium term

We expect the following challenges of ontology matching to be addressed in the medium term: scalability of visualization of the alignment between input ontologies, user interfaces, configuration/customizing technology, industrial-strength research prototypes, including tools for matching ontologies, processing the alignment, and infrastructures.

[5] http://www.altova.com/features_xml2xml_mapforce.html
[6] http://msdn.microsoft.com/library/en-us/introduction/htm/ebiz_intro_story_jgtg.asp
[7] http://www.capescience.com/education/tutorials/index.shtml
[8] http://www.stylusstudio.com/xslt_mapper.html

12.2.3 Long term

In the long term, we expect emergence of good quality matching tools: in the sense of system characteristics, e.g., complexity, design features, performance, quality, and process characteristics, e.g., maintenance. Finally, it is worth noting that, for example, engineers of information integration systems would rather use existing matching systems than build their own. However, it is quite difficult to connect state of the art matching systems to other systems or embed them into the new environments. They are usually packaged as stand alone systems, designed for communication with a human user. In addition, they are not provided with an interface described in terms of abstract data types and logical functionality. We expect some substantial progress on the frameworks for integration of different matching systems into the new environments in the long term.

12.3 Trends in applications

12.3.1 Short term

Matching is an important operation in traditional applications, such as schema integration, data warehousing, enterprise information integration (see Chapter 1). Some examples of commercially available, e.g., EII tools, are IBM Information Integrator, Liquid Data for WebLogic from BEA systems, SAP NetWeaver, and EII platform from Denodo Technologies. However, it is worth mentioning that, even in these tools, a support for handling the semantic heterogeneity problem is still in its early stages.

We expect the above mentioned applications to play a crucial role as in the short term as in the medium and long term. For example, according to Aberdeen Group, the EII market will grow by 60% annually with around

$250M in revenue in 2005[9]. Notice that next we discuss only the new applications as an addition to those already mentioned.

12.3.2 Medium term

There is an emerging line of applications which can be characterized by their dynamics, e.g., agents, peer-to-peer systems, web services (see Chapter 1). Such applications, on the contrary to traditional ones, require a run time matching operation and take advantage of more explicit conceptual models.

We expect these applications to play an important role starting from the medium term, since the necessary technologies (e.g., run time matching) will not mature or converge earlier to support scalable solutions in, e.g., B2B and supply chains.

12.3.3 Long term

It is hard to foresee what is going to happen in a long term, since the web in particular and computer science in general are very dynamic and continuously evolving fields. Of course, in the long term, we expect different variations (e.g., P2P trading grid) of the applications mentioned so far. However, as one of the new possible scenarios, we could see embedding of the semantic matching services inside operation systems.

[9] http://www.denodo.com/english/news/2005/08_06_05.html

Bibliography

[1] Rakesh Agrawal and Ramakrishnan Srikant. On integrating catalogs. In *Proceedings of the International Conference on World Wide Web (WWW)*, pages 603–612, 2001.

[2] Zharko Aleksovski, Michel Klein, Warner ten Kate, and Frank van Harmelen. Matching unstructured vocabularies using a background ontology. In *Proceedings of the International Conference on Knowledge Engineering and Knowledge Management (EKAW)*, 2006.

[3] Yigal Arens, Chun-Nan Hsu, and Craig Knoblock. Query processing in the SIMS information mediator. In *Readings in Agents*, pages 82–90. AAAI press, 1996.

[4] Alessandro Artale, Bernardo Magnini, and Carlo Strapparava. Wordnet for Italian and its use for lexical deiscrimination. In *Proceedings of the Congress of the Italian Association for Artificial Intelligence (AI*IA)*, pages 346–356, 1997.

[5] Alessandro Artale, Christine Parent, and Stefano Spaccapietra. Modeling the evolution of objects in temporal information systems. In *Proceedings of the International Symposium on Foundations of Information and Knowledge Systems (FoIKS)*, pages 22–42, 2006.

[6] Paolo Atzeni, Paolo Cappellari, and Philip Bernstein. ModelGen: Model independent schema translation. In *Proceedings of the Inter-*

national Conference on Data Engineering (ICDE), pages 1111–1112, 2005.

[7] Paolo Atzeni, Paolo Cappellari, and Philip Bernstein. Model-independent schema and data translation. In *Proccedings of the International Conference on Extending Database Technology (EDBT)*, pages 368–385, 2006.

[8] David Aumüller, Hong-Hai Do, Sabine Maßmann, and Erhard Rahm. Schema and ontology matching with COMA++. In *Proceedings of the International Conference on Management of Data (SIGMOD), Software Demonstration*, 2005.

[9] Paolo Avesani, Fausto Giunchiglia, and Mikalai Yatskevich. A large scale taxonomy mapping evaluation. In *Proceedings of the International Semantic Web Conference (ISWC)*, pages 67–81, 2005.

[10] Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter Patel-Schneider, editors. *The description logic handbook: theory, implementations and applications*. Cambridge University Press, 2003.

[11] Carlo Batini, Maurizio Lenzerini, and Shamkant Navathe. A comparative analysis of methodologies for database schema integration. *ACM Computing Surveys*, 18(4):323–364, 1986.

[12] Howard W. Beck, Sunit K. Gala, and Shamkant B. Navathe. Classification as a query processing technique in the CANDIDE semantic data model. In *Proceedings of the International Conference on Data Engineering (ICDE)*, pages 572–581, 1989.

[13] Massimo Benerecetti, Paolo Bouquet, and Stefano Zanobini. Soundness of schema matching methods. In *Proceedings of the European Semantic Web Conference (ESWC)*, pages 211–225, 2005.

[14] Massimo Benerecetti, Paulo Bouquet, and Chiara Ghidini. Contextual reasoning distilled. *Journal of Experimental and Theoretical Artificial Intelligence (JETAI)*, 12(3):279–305, 2000.

[15] Massimo Benerecetti, Paulo Bouquet, and Chiara Ghidini. On the dimensions of context dependence: partiality, approximation, and perspective. In *Proceedings of the International and Interdisciplinary Conference on Modeling and Using Context (CONTEXT)*, pages 59–72, 2001.

[16] Domenico Beneventano, Sonia Bergamaschi, Stefano Lodi, and Claudio Sartori. Consistency checking in complex object database schemata with integrity constraints. *IEEE Transactions on Knowledge and Data Engineering*, 10(4):576–598, 1998.

[17] Sonia Bergamaschi, Domenico Beneventano, Silvana Castano, and Maurizio Vincini. MOMIS: An intelligent system for the integration of semistructured and structured data. Technical Report T3-R07, Università di Modena e Reggio Emilia, Modena (IT), 1998.

[18] Sonia Bergamaschi, Silvana Castano, and Maurizio Vincini. Semantic integration of semistructured and structured data sources. *SIGMOD Record*, 28(1):54–59, 1999.

[19] Claude Berge. *Graphes et hypergraphes*. Dunod, Paris (FR), 1970.

[20] Michael Bergman. The deep web: surfacing hidden value. *The Journal of Electronic Publishing*, 7(1), 2001.

[21] Jacob Berlin and Amihai Motro. Database schema matching using machine learning with feature selection. In *Proceedings of the International Conference on Advanced Information Systems Engineering (CAiSE)*, pages 452–466, 2002.

[22] Tim Berners-Lee, James Hendler, and Ora Lassila. The semantic web. *Scientific American*, 284(5):34–43, 2001.

[23] Philip Bernstein, Alon Halevy, and Rachel Pottinger. A vision of management of complex models. *SIGMOD Record*, 29(4):55–63, 2000.

[24] Philip Bernstein, Sergei Melnik, M. Petropoulos, and Christoph Quix. Industrial-strength schema matching. *SIGMOD Record*, 33(4):38–43, 2004.

[25] Philip Bernstein, Sergey Melnik, and John Churchill. Incremental schema matching. In *Proceedings of the International Conference on Very Large Data Bases (VLDB)*, pages 1167–1170, 2006.

[26] Philip Bernstein and Erhard Rahm. Data warehouse scenarios for model management. In *Proceedings of the Conference on Entity-Relationship Modeling (ER)*, pages 1 – 15, 2000.

[27] Grady Booch, James Rumbaugh, and Ivar Jacobson. *The Unified Modeling Language user guide*. Addison-Wesley, 1998.

[28] Alexander Borgida, Ronald Brachman, Deborah McGuinness, and Lawrence Resnick. CLASSIC: A structural data model for objects. *SIGMOD Record*, 18(2):58–67, 1989.

[29] Paolo Bouquet, Marc Ehrig, Jérôme Euzenat, Enrico Franconi, Pascal Hitzler, Markus Krötzsch, Luciano Serafini, Giorgos Stamou,

York Sure, and Sergio Tessaris. Specification of a common framework for characterizing alignment. Deliverable D2.2.1, Knowledge web NoE, 2004.

[30] Paolo Bouquet, Bernardo Magnini, Luciano Serafini, and Stefano Zanobini. A SAT-based algorithm for context matching. In *Proceedings of the International and Interdisciplinary Conference on Modeling and Using Context (CONTEXT)*, pages 66–79, 2003.

[31] Paolo Bouquet, Luciano Serafini, and Stefano Zanobini. Semantic coordination: A new approach and an application. In *Proceedings of the International Semantic Web Conference (ISWC)*, pages 130–145, 2003.

[32] Paolo Bouquet, Luciano Serafini, and Stefano Zanobini. Semantic coordination of heterogeneous classification schemas. In Steffen Staab and Heiner Stuckenschmidt, editors, *Peer-to-peer and Semantic Web*, chapter 9, pages 185–200. Springer, 2005.

[33] Paolo Bouquet, Luciano Serafini, Stefano Zanobini, and Simone Sceffer. Bootstrapping semantics on the web: meaning elicitation from schemas. In *Proceedings of the World Wide Web Conference (WWW)*, pages 505–512, 2006.

[34] Paulo Bouquet, Fausto Giunchiglia, Frank van Harmelen, Luciano Serafini, and Heiner Stuckenschmidt. Contextualizing ontologies. *Journal of Web Semantics*, 1(1):325–343, 2004.

[35] Michael Boyd, Sasivimol Kittivoravitkul, Charalambos Lazanitis, Peter McBrien, and Nikos Rizopoulos. AutoMed: A BAV data integration system for heterogeneous data sources. In *Proceedings of the International Conference on Advanced Information Systems Engineering (CAiSE)*, pages 82–97, 2004.

[36] Yuri Breitbart. Multidatabase interoperability. *SIGMOD Record*, 19(3):53–60, 1990.

[37] Michael Brodie, John Mylopoulos, and Joachim Schmidt. *On conceptual modeling*. Springer Verlag, 1984.

[38] Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. Description logics for information integration. In *Computational Logic: Logic Programming and Beyond, Essays in Honour of Robert A. Kowalski*, pages 41–60. 2002.

[39] Silvana Castano, Valeria De Antonellis, and Sabrina De Capitani di Vimercati. Global viewing of heterogeneous data sources. *IEEE Transactions on Knowledge and Data Engineering*, 13(2):277–297, 2000.

[40] Silvana Castano, Alfio Ferrara, and Stefano Montanelli. Dynamic knowledge discovery in open, distributed and multi-ontology systems: Techniques and applications. In David Taniar and Wenny Rahayu, editors, *Web Semantics and Ontology*. Idea Group Publishing, 2005.

[41] Silvana Castano, Alfio Ferrara, and Stefano Montanelli. Matching ontologies in open networked systems: Techniques and applications. *Journal on Data Semantics (JoDS)*, V:25–63, 2006.

[42] Lois Mai Chan, John Comaromi, Joan Mitchell, and Mohinder Satija. *Dewey Decimal Classifcation: A Practical Guide*. OCLC Forest Press, Dublin (OH US), 1996.

[43] Kevin Chang, Bin He, and Zhen Zhang. Toward large scale integration: Building a metaquerier over databases on the web. In *Proceedings of the Conference on Innovative Data Systems Research (CIDR)*, pages 44–55, 2005.

[44] Sudarshan Chawathe, Hector Garcia-Molina, Joachim Hammer, Kelly Ireland, Yannis Papakonstantinou, Jeffrey Ullman, and Jennifer Widom. The TSIMMIS project: Integration of heterogeneous information sources. In *Meeting of the Information Processing Society of Japan*, pages 7–18, 1994.

[45] Peter Chen. The entity-relationship model–toward a unified view of data. *ACM Transactions on database systems*, 1(1):9–36, 1976.

[46] Chris Clifton, Ed Hausman, and Arnon Rosenthal. Experience with a combined approach to attribute matching across heterogeneous databases. In *Proceedings of the IFIP Conference on Database Semantics*, 1997.

[47] William Cohen, Pradeep Ravikumar, and Stephen Fienberg. A comparison of string metrics for matching names and records. In *Proceedings of the Workshop on Data Cleaning and Object Consolidation at the International Conference on Knowledge Discovery and Data Mining (KDD)*, 2003.

[48] Oscar Corcho. *A declarative approach to ontology translation with knowledge preservation.* PhD thesis, Universidad Politécnica de Madrid, 2004.

[49] Nuno Alexandre Pinto da Silva. *Multi-dimensional service-oriented ontology mapping.* PhD thesis, Universidade de Trás-os-Montes e Alto Douro, 2004.

[50] Martin Davis, George Longemann, and Donald Loveland. A machine program for theorem proving. *Communications of the ACM*, 5(7):394–397, 1962.

[51] Martin Davis and Hilary Putnam. A computing procedure for quantification theory. *Journal of the ACM*, 7:201–215, 1960.

[52] Mike Dean and Guus Schreiber (eds.). OWL web ontology language reference. Recommendation, W3C, February 2004.

[53] Robin Dhamankar, Yoonkyong Lee, An-Hai Doan, Alon Halevy, and Pedro Domingos. iMAP: Discovering complex semantic matches between database schemas. In *Proceedings of the International Conference on Management of Data (SIGMOD)*, pages 383–394, 2004.

[54] Tommaso Di Noia, Eugenio Di Sciascio, Francesco Donini, and Marina Mongiello. A system for principled matchmaking in an electronic marketplace. In *Proceedings of the World Wide Web Conference (WWW)*, pages 321–330, 2003.

[55] Rose Dieng and Stefan Hug. Comparison of "personal ontologies" represented through conceptual graphs. In *Proceedings of the European Conference on Artificial Intelligence (ECAI)*, pages 341–345, 1998.

[56] Hong-Hai Do. *Schema matching and mapping-based data integration.* PhD thesis, University of Leipzig, 2005.

[57] Hong-Hai Do, Sergei Melnik, and Erhard Rahm. Comparison of schema matching evaluations. In *Proceedings of the GI-Workshop Web and Databases, Erfurt (DE)*, 2002.

[58] Hong-Hai Do and Erhard Rahm. COMA – a system for flexible combination of schema matching approaches. In *Proceedings of the International Conference on Very Large Databases*, pages 610–621, 2002.

[59] An-Hai Doan, Pedro Domingos, and Alon Halevy. Reconciling schemas of disparate data sources: A machine-learning approach. In *Proceeding of International Conference on Management of Data (SIGMOD)*, pages 509–520, 2001.

[60] An-Hai Doan, Pedro Domingos, and Alon Halevy. Learning to match the schemas of data sources: A multistrategy approach. *Machine Learning*, 50(3):279–301, 2003.

[61] An-Hai Doan and Alon Halevy. Semantic integration research in the database community: A brief survey. *AI Magazine, Special Issue on Semantic Integration*, 26(1):83–94, 2005.

[62] An-Hai Doan, Jayant Madhavan, Pedro Domingos, and Alon Halevy. Learning to map ontologies on the semantic web. In *Proceedings of the International World Wide Web Conference (WWW)*, pages 662–673, 2003.

[63] An-Hai Doan, Jayant Madhavan, Pedro Domingos, and Alon Halevy. Ontollogy matching: a machine learning approach. In Steffen Staab and Rudi Studer, editors, *Handbook of ontologies*, International handbooks on information systems, chapter 18, pages 385–404. Springer Verlag, Berlin (DE), 2004.

[64] Pedro Domingos and Michael Pazzani. Beyond independence: Conditions for the optimality of the simple bayesian classifier. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 105–112, 1996.

[65] Denise Draper, Alon Halevy, and Daniel Weld. The nimble integration engine. In *Proceedings of the International Conference on Management of Data (SIGMOD)*, pages 567–568, 2001.

[66] Marc Ehrig. *Ontology alignment: bridging the semantic gap.* PhD thesis, University of Karlsruhe, 2006.

[67] Marc Ehrig, Staab Staab, and York Sure. Bootstrapping ontology alignment methods with APFEL. In *Proceedings of the International Semantic Web Conference (ISWC)*, pages 186–200, 2005.

[68] Marc Ehrig and Steffen Staab. QOM – quick ontology mapping. In *Proceedings of the International Semantic Web Conference (ISWC)*, pages 683–697, 2004.

[69] Marc Ehrig and York Sure. Ontology mapping – an integrated approach. In *Proceedings of the European Semantic Web Symposium (ESWS)*, pages 76–91, 2004.

[70] Ahmed Elmagarmid, Marek Rusinkiewicz, Amith Sheth, and editors. *Management of Heterogeneous and Autonomous Database Systems.* Morgan Kaufmann, 1999.

[71] Jérôme Euzenat. Towards a principled approach to semantic interoperability. In *Proceedings of the Workshop on Ontology and Information Sharing at the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 19–25, 2001.

[72] Jérôme Euzenat. Towards composing and benchmarking ontology alignments. In *Proceedings of the Workshop on Semantic Integration at the International Semantic Web Conference (ISWC)*, pages 165–166, 2003.

[73] Jérôme Euzenat. An API for ontology alignment. In *Proceedings of the International Semantic Web Conference (ISWC)*, pages 698–712, 2004.

[74] Jérôme Euzenat, Malgorzata Mochol Pavel Shvaiko, Heiner Stuckenschmidt, Ondřej Šváb, Vojtěch Svátek, Willem Robert van Hage, and Mikalai Yatskevich. Results of the ontology alignment evaluation initiative 2006. In *Proceedings of the International Workshop on Ontology Matching (OM) at the International Semantic Web Conference (ISWC)*, 2006.

[75] Jérôme Euzenat and Pavel Shvaiko. *Ontology matching*. Springer-Verlag, Heidelberg (DE), 2007. to appear.

[76] Jérôme Euzenat and Heiner Stuckenschmidt. The 'family of languages' approach to semantic interoperability. In *Knowledge transformation for the semantic web*, pages 49–63. 2003.

[77] Jérôme Euzenat, Heiner Stuckenschmidt, and Mikalai Yatskevich. Introduction to the ontology alignment evaluation 2005. In *Proceedings of the Workshop on Integrating Ontologies at the International Conference on Knowledge Capture (K-CAP)*, 2005.

[78] Jérôme Euzenat and Petko Valtchev. Similarity-based ontology alignment in OWL-lite. In *Proceedings of the European Conference on Artificial Intelligence (ECAI)*, pages 333–337, 2004.

[79] Ronald Fagin, Phokion Kolaitis, Lucian Popa, and Wang Chiew Tan. Composing schema mappings: Second-order dependencies to the rescue. In *Proceedings of the Symposium on Principles of Database Systems (PODS)*, pages 83–94, 2004.

[80] Christiane Fellbaum. *Wordnet: An Electronic Lexical Database*. Cambridge, US: The MIT Press, 1998.

[81] Dieter Fensel. *Ontologies: A Silver Bullet for Knowledge Management and Electronic Commerce*. Springer Verlag, 2nd edition, 2004.

[82] Dieter Fensel, Holger Lausen, Axel Polleres, Jos de Bruijn, Michael Stollberg, Dumitru Roman, and John Domingue. *Enabling Semantic Web Services: The Web Service Modeling Ontology*. Springer, 2007.

[83] FIPA0037. FIPA ACL communicative act library specification. Technical report, FIPA, 2002. http://www.fipa.org/specs/fipa00037.

[84] FIPA0061. FIPA ACL message structure specification, 2002. http://www.fipa.org/specs/fipa00061.

[85] Avigdor Gal. Managing uncertainty in schema matching with top-k schema mappings. *Journal on Data Semantics (JoDS)*, VI:90–114, 2006.

[86] Avigdor Gal, Giovanni Modica, Hassan Jamil, and Ami Eyal. Automatic ontology matching using application semantics. *AI Magazine*, 26(1):21–32, 2005.

[87] David Gale and Lloyd Stowell Shapley. College admissions and the stability of marriage. *American Mathematical Monthly*, 69:5–15, 1962.

[88] Aldo Gangemi, Nicola Guarino, Claudio Masolo, and Alessandro Oltramari. Sweetening WordNet with DOLCE. *AI Magazine*, 24(3):13–24, 2003.

[89] Aldo Gangemi, Nicola Guarino, and Alessandro Oltramari. Conceptual analysis of lexical taxonomies: the case of wordnet top-level. In *Proceedings of the International Conference on Formal Ontology in Information Systems (FOIS)*, pages 285–296, 2001.

[90] Bernhard Ganter and Rudolf Wille. *Formal concept analysis: mathematical foundations*. Springer Verlag, 1999.

[91] Michael Garey and David Johnson. *Computers and intractability: a guide to the theory of NP-completeness*. W. H. Freeman & Co., 1979.

[92] Chiara Ghidini and Fausto Giunchiglia. Local models semantics, or contextual reasoning = locality + compatibility. *Artificial Intelligence*, 127(2):221–259, 2001.

[93] Chiara Ghidini and Fausto Giunchiglia. A semantics for abstraction. In *Proceedings of the European Conference on Artificial Intelligence (ECAI)*, pages 343–347, 2004.

[94] Fausto Giunchiglia. Contextual reasoning. *Epistemologia, special issue on I Linguaggi ele Macchine*, XVI:345–364, 1993.

[95] Fausto Giunchiglia, Maurizio Marchese, and Ilya Zaihrayeu. Encoding classifications into lightweight ontologies. In *Proceedings of the European Semantic Web Conference (ESWC)*, pages 80–94, 2006.

[96] Fausto Giunchiglia and Pavel Shvaiko. Semantic matching. In *Proceedings of the Workshop on Ontologies and Distributed Systems at the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 139–146, 2003.

[97] Fausto Giunchiglia and Pavel Shvaiko. Semantic matching. *The Knowledge Engineering Review*, 18(3):265–280, 2003.

[98] Fausto Giunchiglia, Pavel Shvaiko, and Mikalai Yatskevich. S-Match: an algorithm and an implementation of semantic matching. In *Proceedings of the European Semantic Web Symposium (ESWS)*, pages 61–75, 2004.

[99] Fausto Giunchiglia, Pavel Shvaiko, and Mikalai Yatskevich. Semantic schema matching. In *Proceedings of the International Conference on Cooperative Information Systems (CoopIS)*, pages 347–365, 2005.

[100] Fausto Giunchiglia, Pavel Shvaiko, and Mikalai Yatskevich. Discovering missing background knowledge in ontology matching. In *Proceedings of the European Conference on Artificial Intelligence (ECAI)*, pages 382–386, 2006.

[101] Fausto Giunchiglia and Mikalai Yatskevich. Element level semantic matching. In *Proceedings of the Meaning Coordination and Negotiation Workshop at the International Semantic Web Conference (ISWC)*, 2004.

[102] Fausto Giunchiglia, Mikalai Yatskevich, and Enrico Giunchiglia. Efficient semantic matching. In *Proceedings of the European Semantic Web Conference (ESWC)*, pages 272–289, 2005.

[103] Fausto Giunchiglia, Mikalai Yatskevich, and Pavel Shvaiko. Semantic matching: algorithms and implementation. *Journal on Data Semantics (JoDS)*, IX, 2006. to appear.

[104] Fausto Giunchiglia and Ilya Zaihrayeu. Making peer databases interact - a vision for an architecture supporting data coordination. In *Proceedings of the International Workshop on Cooperative Information Agents (CIA)*, pages 18–35, 2002.

[105] François Goasdoué, Véronique Lattes, and Marie-Christine Rousset. The use of CARIN language and algorithms for information integration: The PICSEL system. *International Journal of Cooperative Information Systems*, 9(4):383–401, 2000.

[106] Cheng-Hian Goh. *Representing and reasoning about semantic conflicts in heterogeneous information sources.* PhD thesis, MIT, 1997.

[107] Irving John Good. *The estimation of probabilities: an essay on modern Bayesian methods.* Classics Series. The MIT press, 1965.

[108] Nicola Guarino. Formal ontology in information systems. In *Proceedings of the International Conference on Formal Ontology in Information Systems (FOIS)*, pages 3–15, 1998.

[109] Nicola Guarino. The role of ontologies for the semantic web (and beyond). Technical report, Laboratory for Applied Ontology, Institute for Cognitive Sciences and Technology (ISTC-CNR), 2004.

[110] Nicola Guarino and Christopher A. Welty. Evaluating ontological decisions with OntoClean. *Communications of the ACM*, 45(2):61–65, 2002.

[111] Nicola Guarino and Christopher A. Welty. An overview of OntoClean. In Steffen Staab and Rudi Studer, editors, *Handbook on Ontologies*, International Handbooks on Information Systems, pages 151–172. Springer, 2004.

[112] V. Haarslev, R. Moller, and M. Wessel. RACER: Semantic middleware for industrial projects based on RDF/OWL, a W3C Standard. `http://www.sts.tu-harburg.de/~r.f.moeller/racer/`.

[113] Laura Haas, Mauricio Hernández, Howard Ho, Lucian Popa, and Mary Roth. Clio grows up: from research prototype to industrial tool. In *Proceedings of the International Conference on Management of Data (SIGMOD)*, pages 805–810, 2005.

[114] Alon Halevy, Naveen Ashish, Dina Bitton, Michael Carey, Denise Draper, Jeff Pollock, Arnon Rosenthal, and Vishal Sikka. Enterprise information integration: successes, challenges and controversies. In *Proceedings of the International Conference on Management of Data (SIGMOD)*, pages 778–787, 2005.

[115] Adil Hameed, Alun Preece, and Derek Sleeman. Ontology reconciliation. In Steffen Staab and Rudi Studer, editors, *Handbook of ontologies*, International handbooks on information systems, chapter 12, pages 231–250. Springer Verlag, Berlin (DE), 2004.

[116] Bin He and Kevin Chang. Automatic complex schema matching across web query interfaces: A correlation mining approach. *ACM Transactions on Database Systems*, 31(1):1–45, 2006.

[117] Bin He, Mitesh Patel, Zhen Zhang, and Kevin Chang. Accessing the deep web: a survey. *Communications of the ACM*, 2006.

[118] Ian Horrocks, Peter Patel-Schneider, Harold Boley, Said Tabet, Benjamin Grosof, and Mike Dean. SWRL: a semantic web rule language combining OWL and RuleML, 2004. http://www.w3.org/Submission/SWRL/.

[119] Eduard Hovy. Combining and standardizing large-scale, practical ontologies for machine translation and other uses. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC)*, pages 535–542, 1998.

[120] Richard Hull. Managing semantic heterogeneity in databases: a theoretical prospective. In *Proceedings of the Symposium on Principles of Database Systems (PODS)*, pages 51–61, 1997.

[121] Ryutaro Ichise, Hideaki Takeda, and Shinichi Honiden. Integrating multiple internet directories by instance-based learning. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 22–30, 2003.

[122] Zachary Ives, Alon Halevy, Peter Mork, and Igor Tatarinov. Piazza: mediation and integration infrastructure for semantic web data. *Jornal of Web Semantics*, 1(2):155–175, 2004.

[123] Yannis Kalfoglou and Marco Schorlemmer. Ontology mapping: the state of the art. *The Knowledge Engineering Review*, 18(1):1–31, 2003.

[124] Vipul Kashyap and Amit Sheth. Semantic and schematic similarities between database objects: a context-based approach. *The VLDB Journal*, 5(4):276–304, 1996.

[125] Vipul Kashyap and Amit Sheth. Semantic heterogeneity in global information systems: The role of metadata, context and ontologies. In Michael Papazoglou and Gunter Schlageter, editors, *Cooperative Information Systems*, pages 139–178. Academic Press, 1998.

[126] David Kensche, Christoph Quix, Mohamed Amine Chatti, and Matthias Jarke. Gerome: A generic role based metamodel for model management. In *Proceedings of the International Conference on Ontologies, DataBases, and Applications of Semantics (ODBASE)*, pages 1206–1224, 2005.

[127] Won Kim and Jungyun Seo. Classifying schematic and data heterogeneity in multidatabase systems. *Computer*, 24(12):12–18, 1991.

[128] Michel Klein. Combining and relating ontologies: an analysis of problems and solutions. In *Proceedings of the Workshop on Ontologies and Information Sharing at the International Joint Conference on Artificial Intelligence (IJCAI)*, 2001.

[129] Loredana Laera, Valentina Tamma, Jérôme Euzenat, Trevor Bench-Capon, and Terry Payne. Reaching agreement over ontology align-

ments. In *Proceedings of the International Semantic Web Conference (ISWC)*, 2006.

[130] James Larson, Shamkant Navathe, and Ramez Elmasri. A theory of attributed equivalence in databases with application to schema integration. *IEEE Transactions on Software Engineering*, 15(4):449–463, 1989.

[131] Daniel Le Berre. Sat4j: A satisfiability library for Java. http://www.sat4j.org/, 2004.

[132] Alain Léger, Johannes Heinecke, Lyndon Nixon, Pavel Shvaiko, Jean Charlet, Paola Hobson, and François Goasdoué. The semantic web from an industry perspective. In *Tutorial at Reasoning Web, Second International Summer School*, pages 232–268. Springer, 2006.

[133] Alain Léger, Lyndon Nixon, and Pavel Shvaiko. On identifying knowledge processing requirements. In *Proceedings of the International Semantic Web Conference (ISWC)*, pages 928–943, 2005.

[134] Alan Léger, Lyndon Nixon, Pavel Shvaiko, and Jean Charlet. Semantic web applications: Fields and business cases.the industry challenges the research. In *Proceedings of the International Conference on Industrial Applications of Semantic Web (IASW)*, pages 27–46, 2005.

[135] Maurizio Lenzerini. Data integration: A theoretical perspective. In *Proceedings of the Symposium on Principles of Database Systems (PODS)*, pages 233–246, 2002.

[136] Nicola Leone, Gianluigi Greco, Giovambattista Ianni, Vincenzino Lio, Giorgio Terracina, Thomas Eiter, Wolfgang Faber, Michael Fink, Georg Gottlob, Riccardo Rosati, Domenico Lembo, Maurizio

Lenzerini, Marco Ruzzi, Edyta Kalka, Bartosz Nowicki, and Witold Staniszkis. The INFOMIX system for advanced integration of incomplete and inconsistent data. In *Proceedings of the International Conference on Management of Data (SIGMOD)*, pages 915–917, 2005.

[137] Wen-Syan Li and Chris Clifton. Semantic integration in heterogeneous databases using neural networks. In *Proceedings of the Very Large Data Bases Conference (VLDB)*, pages 1–12, 1994.

[138] Wen-Syan Li and Chris Clifton. SEMINT: a tool for identifying attribute correspondences in heterogeneous databases using neural networks. *Data and Knowledge Engineering*, 33(1):49–84, 2000.

[139] Vanessa Lopez, Enrico Motta, and Victoria Uren. PowerAqua: Fishing the semantic web. In *Proceedings of the European Semantic Web Conference (ESWC)*, pages 393–410, 2006.

[140] Vanessa Lopez, Michele Pasin, and Enrico Motta. AquaLog: An ontology-portable question answering system for the semantic web. In *Proceedings of the European Semantic Web Conference (ESWC)*, pages 546–562, 2005.

[141] László Lovász and Michael Plummer. *Matching Theory*. North-Holland, Amsterdam (NL), 1986.

[142] Alexander Mädche, Boris Motik, Nuno Silva, and Raphael Volz. MAFRA – a mapping framework for distributed ontologies. In *Proceedings of the International Conference on Knowledge Engineering and Knowledge Management (EKAW)*, pages 235–250, 2002.

[143] Alexander Mädche and Steffen Staab. Measuring similarity between ontologies. In *Proceedings of the International Conference on Knowledge Engineering and Management (EKAW)*, pages 251–263, 2002.

[144] Jayant Madhavan, Philip Bernstein, An-Hai Doan, and Alon Halevy. Corpus-based schema matching. In *Proceedings of the International Conference on Data Engineering (ICDE)*, pages 57–68, 2005.

[145] Jayant Madhavan, Philip Bernstein, Pedro Domingos, and Alon Halevy. Representing and reasoning about mappings between domain models. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 122–133, 2002.

[146] Jayant Madhavan, Philip Bernstein, and Erhard Rahm. Generic schema matching using Cupid. In *Proceedings of the International Conference on Very Large Data Bases (VLDB)*, pages 48–58, 2001.

[147] B. Magnini, Luciano Serafini, and M. Speranza. Making explicit the semantics hidden in schema models. In *Proceedings of the Workshop on Human Language Technology for the Semantic Web and Web Services at the International Semantic Web Conference (ISWC)*, 2003.

[148] Bernardo Magnini, Manuela Speranza, and Christian Girardi. A semantic-based approach to interoperability of classification hierarchies: Evaluation of linguistic techniques. In *Proceedings of the International Conference on Computational Linguistics (COLING)*, 2004.

[149] Andrew McCallum and Kamal Nigam. A comparison of event models for naive bayes text classification. In *Proceedings of the Workshop on Learning for Text Categorization at the National Conference on Artificial Intelligence (AAAI)*, 1998.

[150] Deborah McGuinness, Richard Fikes, James Rice, and Steve Wilder. An environment for merging and testing large ontologies. In *Proceedings of the International Conference on the Principles of Knowledge Representation and Reasoning (KR)*, pages 483–493, 2000.

[151] Deborah McGuinness and Paulo Pinheiro da Silva. Infrastructure for web explanations. In *Proceedings of the International Semantic Web Conference (ISWC)*, pages 113–129, 2003.

[152] Deborah McGuinness and Paulo Pinheiro da Silva. Registry-based support for information integration. In *Proceedings of the Workshop on Information Integration on the Web at the International Joint Conference on Artificial Intelligence (IJCAI)*, 2003.

[153] Deborah McGuinness and Paulo Pinheiro da Silva. Explaining answers from the semantic web: The Inference Web approach. *Journal of Web Semantics*, 1(4):397–413, 2004.

[154] Deborah L. McGuinness, Pavel Shvaiko, Fausto Giunchiglia, and Paulo Pinheiro da Silva. Towards explaining semantic matching. In *Proceedings of the International Workshop on Description Logics (DL) at the International Conference on the Principles of Knowledge Representation and Reasoning (KR)*, 2004.

[155] Fiona McNeill. *Dynamic ontology refinement*. PhD thesis, University of Edinburgh (UK), 2006.

[156] Open information model, version 1.0. http://mdcinfo/oim/oim10.html, 1999.

[157] Brahim Medjahed and Athman Bouguettaya. A multilevel composability model for semantic web services. *IEEE Transactions on Knowledge and Data Engineering*, 17(7):954–968, 2005.

[158] Sergey Melnik, Philip Bernstein, Alon Halevy, and Erhard Rahm. Supporting executable mappings in model management. In *Proceedings of the International Conference on Management of Data (SIGMOD)*, pages 167–178, 2005.

[159] Sergey Melnik, Hector Garcia-Molina, and Erhard Rahm. Similarity flooding: a versatile graph matching algorithm. In *Proceedings of the International Conference on Data Engineering (ICDE)*, pages 117–128, 2002.

[160] Sergey Melnik, Erhard Rahm, and Philip Bernstein. Developing metadata-intensive applications with Rondo. *Journal of Web Semantics*, 1(1):47–74, 2003.

[161] Sergey Melnik, Erhard Rahm, and Philip Bernstein. Rondo: A programming platform for model management. In *Proceedings of the International Conference on Management of Data (SIGMOD)*, pages 193–204, 2003.

[162] Eduardo Mena, Vipul Kashyap, Amit Sheth, and Arantza Illarramendi. Observer: An approach for query processing in global information systems based on interoperability between pre-existing ontologies. In *Proceedings of the International Conference on Cooperative Information Systems (CoopIS)*, pages 14–25, 1996.

[163] George Miller. WordNet: A lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

[164] Renée Miller, Laura Haas, and Mauricio Hernández. Schema mapping as query discovery. In *Proceedings of the International Conference on Very Large Data Bases, VLDB*, pages 77–88, 2000.

[165] Renée Miller, Mauricio Hernández, Laura Haas, Lingling Yan, Howard Ho, Ronald Fagin, and Lucian Popa. The Clio project: managing heterogeneity. *SIGMOD Record*, 30(1):78–83, 2001.

[166] Tova Milo and Sagit Zohar. Using schema matching to simplify heterogeneous data translation. In *Proceedings of the Very Large Databases Conference (VLDB)*, pages 122–133, 1998.

[167] Prasenjit Mitra, Natalya Noy, and Anuj Jaiswal. Ontology mapping discovery with uncertainty. In *Proceedings of the International Semantic Web Conference (ISWC)*, pages 537–547, 2005.

[168] Prasenjit Mitra and Gio Wiederhold. Resolving terminological heterogeneity in ontologies. In *Proceedings of the Workshop on Ontologies and Semantic Interoperability at the European Conference on Artificial Intelligence (ECAI)*, 2002.

[169] Prasenjit Mitra, Gio Wiederhold, and Jan Jannink. Semi-automatic integration of knowledge sources. In *Proceedings of the International Conference On Information Fusion (FUSION)*, 1999.

[170] Prasenjit Mitra, Gio Wiederhold, and Martin Kersten. A graph-oriented model for articulation of ontology interdependencies. In *Proccedings of the International Conference on Extending Database Technology (EDBT)*, pages 86–100, 2000.

[171] Giovanni Modica, Avigdor Gal, and Hasan Jamil. The use of machine-generated ontologies in dynamic information seeking. In *Proceedings of the International Conference on Cooperative Information Systems (CoopIS)*, pages 433–448, 2001.

[172] Felix Naumann, Ching-Tien Ho, Xuqing Tian, Laura Haas, and Nimrod Megiddo. Attribute classification using feature analysis. In *Proceedings of the International Conference on Data Engineering (ICDE)*, page 271, 2002.

[173] Wolfgang Nejdl, Boris Wolf, Changtao Qu, Stefan Decker, Michael Sintek, Ambjörn Naeve, Mikael Nilsson, Matthias Palmér, and Tore Risch. Edutella: A P2P networking infrastructure based on RDF. In *Proceedings of the World Wide Web Conference (WWW)*, pages 604–615, 2002.

[174] Ian Niles and Adam Pease. Towards a standard upper ontology. In *Proceedings of the International Conference on Formal Ontology in Information Systems (FOIS)*, pages 2–9, 2001.

[175] Henrik Nottelmann and Umberto Straccia. sPLMap: A probabilistic approach to schema matching. In *Proceedings of the European Conference on Information Retrieval Research (ECIR)*, pages 81–95, 2005.

[176] Natalya Noy. Semantic integration: A survey of ontology-based approaches. *SIGMOD Record*, 33(4):65–70, 2004.

[177] Natalya Noy. Tools for mapping and merging ontologies. In Steffen Staab and Rudi Studer, editors, *Handbook of ontologies*, International handbooks on information systems, chapter 18, pages 365–384. Springer Verlag, Berlin (DE), 2004.

[178] Natalya Noy and Michel Klein. Ontology evolution: Not the same as schema evolution. *Knowledge and Information Systems*, 6(4):428–440, 2004.

[179] Natalya Noy and Marc Musen. SMART: Automated support for ontology merging and alignment. In *Proceedings of the Workshop on Knowledge Acquisition, Modeling, and Management*, 1999.

[180] Natalya Noy and Marc Musen. The PROMPT suite: interactive tools for ontology merging and mapping. *International Journal of Human-Computer Studies (IJHCS)*, 59(6):983–1024, 2003.

[181] Natalya Noy and Mark Musen. Anchor-PROMPT: Using non-local context for semantic matching. In *Proceedings of the Workshop on Ontology and Information Sharing at the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 63–70, 2001.

[182] Natalya Noy and Mark Musen. PromptDiff: A fixed-point algorithm for comparing ontology versions. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 744–750, 2002.

[183] Natalya Noy and Mark Musen. Ontology versioning in an ontology management framework. *IEEE Intelligent Systems*, 19(4):6–13, 2004.

[184] Swapna Oundhakar, Kunal Verma, Kaarthik Sivashanugam, Amit Sheth, and John Miller. Discovery of web services in a multi-ontology and federated registry environment. *International Journal of Web Services Research (JWSR)*, 2(3):1–32, 2005.

[185] Tamer Özsu and Patrick Valduriez. *Principles of Distributed Database Systems*. Prentice Hall, second edition, 1999.

[186] Luigi Palopoli, Luigi Pontieri, Giorgio Terracina, and Domenico Ursino. Intensional and extensional integration and abstraction of heterogeneous databases. *Data Knowledge Engineering*, 35(3):201–237, 2000.

[187] Luigi Palopoli, Domenico Saccá, Giorgio Terracina, and Domenico Ursino. Uniform techniques for deriving similarities of objects and subschemes in heterogeneous databases. *IEEE Transactions on Knowledge and Data Engineering*, 15(2):271–294, 2003.

[188] Luigi Palopoli, Domenico Saccà, and Domenico Ursino. An automatic techniques for detecting type conflicts in database schemes. In *Proceedings of the Conference on Information and Knowledge Management (CIKM)*, pages 306–313, 1998.

[189] Luigi Palopoli, Giorgio Terracina, and Domenico Ursino. DIKE: a system supporting the semi-automatic construction of cooperative information systems from heterogeneous databases. *Software–practice and experinece*, 33(9):847–884, 2003.

[190] Rong Pan, Zhongli Ding, Yang Yu, and Yun Peng. A bayesian network approach to ontology mapping. In *Proceedings of the International Semantic Web Conference (ISWC)*, pages 563–577, 2005.

[191] Massimo Paolucci, Takahiro Kawamura, Terry Payne, and Katia Sycara. Semantic matching of web services capabilities. In *Proceedings of the International Semantic Web Conference (ISWC)*, pages 333–347, 2002.

[192] Christine Parent and Stefano Spaccapietra. Issues and approaches of database integration. *Communications of the ACM*, 41(5):166–178, 1998.

[193] Christine Parent and Stefano Spaccapietra. Database integration: the key to data interoperability. In *Object-Oriented Data Modeling*. The MIT Press, Cambridge (MA US), 2000.

[194] Christine Parent, Stefano Spaccapietra, and Esteban Zimányi. *Conceptual Modeling for Traditional and Spatio-Temporal Applications : The MADS Approach*. Springer-Verlag, Heidelberg (DE), 2006.

[195] Paulo Pinheiro da Silva, Deborah McGuinness, and Richard Fikes. A proof markup language for semantic web services. *Information systems*, 31(4):381–395, 2006.

[196] Alun Preece, Kit-Ying Hui, Alex Gray, Philippe Marti, Trevor Bench-Capon, Dean Jones, and Zhan Cui. The KRAFT architecture for knowledge fusion and transformation. *Knowledge-Based Systems*, 13(2-3):113–120, 2000.

[197] Roy Rada, Hafedh Mili, Ellen Bicknell, and Maria Blettner. Development and application of a metric on semantic nets. *IEEE Transactions on Systems, Man and Cybernetics*, 19(1):17–30, 1989.

[198] Erhard Rahm and Philip Bernstein. A survey of approaches to automatic schema matching. *The VLDB Journal*, 10(4):334–350, 2001.

[199] Erhard Rahm, Hong-Hai Do, and Sabine Maßmann. Matching large XML schemas. *SIGMOD Record*, 33(4):26–31, 2004.

[200] Phillip Resnik. Semantic similarity in a taxonomy: an information-based measure and its application to problems of ambiguity in natural language. *Journal of Artificial Intelligence Research (JAIR)*, 11:95–130, 1999.

[201] George G. Robertson, Mary P. Czerwinski, and John E. Churchill. Visualization of mappings between schemas. In *Proceedings of the Conference on Human Factors in Computing Systems (CHI)*, pages 431–439, 2005.

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

[203] Marie-Christine Rousset, Philippe Adjiman, Philippe Chatalic, François Goasdoué, and Laurent Simon. Somewhere in the semantic

web. In *Proceedings of the Conference on Current Trends in Theory and Practice of Informatics (SOFSEM)*, pages 84–99, 2006.

[204] Marta Sabou, Vanessa Lopez, and Enrico Motta. Ontology selection for the real semantic web: How to cover the queens birthday dinner? In *Proceedings of the International Conference on Knowledge Engineering and Knowledge Management (EKAW)*, 2006.

[205] Mayssam Sayyadian, Yoonkyong Lee, An-Hai Doan, and Arnon Rosenthal. Tuning schema matching software using synthetic scenarios. In *Proceedings of the Very Large Data Bases Conference (VLDB)*, pages 994–1005, 2005.

[206] Randall Schuh. *Biological Systematics: Principles and Applications*. Cornell University Press, 1999.

[207] Ellen Schulten, Hans Akkermans, Guy Botquin, Martin Dorr, Nicola Guarino, Nelson Lopes, and Norman Sadeh. Call for participants: The e-commerce product classification challenge. *IEEE Intelligent Systems*, 16(4):86–c3, 2001.

[208] Luciano Serafini, Heiner Stuckenschmidt, and Holger Wache. A formal investigation of mapping language for terminological knowledge. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 576–581, 2005.

[209] Microsoft BizTalk Server. Biztalk mapper. http://msdn.microsoft.com/biztalk/, 2004.

[210] Dennis Shasha, Jason Tsong-Li Wang, and Rosalba Giugno. Algorithmics and applications of tree and graph searching. In *Proceedings of the Symposium on Principles of Database Systems (PODS)*, pages 39–52, 2002.

[211] Dennis Shasha and Kaizhong Zhang. Approximate tree pattern matching. In Alberto Apostolico and Zvi Galil, editors, *Pattern Matching Algorithms*, chapter 14, pages 341–371. Oxford University Press, 1997.

[212] Amit Sheth and James Larson. Federated database systems for managing distributed, heterogeneous, and autonomous databases. *ACM Computing Surveys*, 22(3):183–236, 1990.

[213] Pavel Shvaiko. A classification of schema-based matching approaches. In *Proceedings of the Meaning Coordination and Negotiation Workshop at the International Semantic Web Conference (ISWC)*, 2004.

[214] Pavel Shvaiko and Jérôme Euzenat. A survey of schema-based matching approaches. *Journal on Data Semantics (JoDS)*, IV:146–171, 2005.

[215] Pavel Shvaiko, Fausto Giunchiglia, Paulo Pinheiro da Silva, and Deborah McGuinness. Web explanations for semantic heterogeneity discovery. In *Proceedings of the European Semantic Web Conference (ESWC)*, pages 303–317, 2005.

[216] Pavel Shvaiko, Fausto Giunchiglia, Marco Schorlemmer, Fiona McNeill, Alan Bundy, Maurizio Marchese, Mikalai Yatskevich, Ilya Zaihrayeu, Bo Ho, Vanessa Lopez, Marta Sabou, Joaqín Abian, Ronny Siebes, and Spyros Kotoulas. Dynamic ontology matching: a survey. Deliverable 3.1, OpenKnowledge STReP, 2006.

[217] Mike Smith, Christopher Welty, and Deborah McGuinness (eds.). OWL web ontology language guide. Recommendation, W3C, February 10 2004.

[218] Anastasiya Sotnykova, Christèle Vangenot, Nadine Cullot, Nacra Bennacer, and Marie-Aude Aufaure. Semantic mappings in description logics for spatio-temporal database schema integration. *Journal on Data Semantics (JoDS)*, III:143–167, 2005.

[219] Stefano Spaccapietra and Christine Parent. Conflicts and correspondence assertions in interoperable databases. *SIGMOD Record*, 20(4):49–54, 1991.

[220] Stefano Spaccapietra and Christine Parent. View integration: A step forward in solving structural conflicts. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 6(2):258–274, 1994.

[221] Stefano Spaccapietra, Christine Parent, and Yann Dupont. Model independent assertions for integration of heterogeneous schemas. *The VLDB Journal*, 1(1):81–126, 1992.

[222] Steffen Staab and Rudi Studer. *Handbook of ontologies*. International handbooks on information systems. Springer Verlag, Berlin (DE), 2004.

[223] Gerd Stumme and Alexander Mädche. FCA-Merge: Bottom-up merging of ontologies. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 225–234, 2001.

[224] Kai Ming Ting and Ian Witten. Issues in stacked generalization. *Journal of Artificial Intelligence Research (JAIR)*, 10:271–289, 1999.

[225] Mike Uschold and Michael Gruninger. Ontologies and semantics for seamless connectivity. *SIGMOD Record*, 33(4):58–64, 2004.

[226] Petko Valtchev. *Construction automatique de taxonomies pour l'aide à la représentation de connaissances par objets*. Thèse d'informatique, Université Grenoble 1, 1999.

[227] Petko Valtchev and Jérôme Euzenat. Dissimilarity measure for collections of objects and values. In *Proceedings of the Symposium on Intelligent Data Analysis (IDA)*, pages 259–272, 1997.

[228] Rogier van Eijk, Frank de Boer, Wiebe van de Hoek, and John-Jules Meyer. On dynamically generated ontology translators in agent communication. *International Journal of Intelligent Systems (IJIS)*, 16:587–607, 2001.

[229] Marjolein van Gendt, Antoine Isaac, Lourens van der Meij, and Stefan Schlobach. Semantic web techniques for multiple views on heterogeneous collections: A case study. In *Proceedings of the European Conference on Digital Libraries (ECDL)*, pages 426–437, 2006.

[230] Cornelis Joost (Keith) van Rijsbergen. *Information retrieval*. Butterworths, 1975. http://www.dcs.gla.ac.uk/Keith/Preface.html.

[231] Pepijn Visser, Dean Jones, Trevor Bench-Capon, and Michael Shave. An analysis of ontological mismatches: heterogeneity versus interoperability. In *Proceedings of the AAAI Spring Symposium on Ontological Engineering*, 1997.

[232] Piek Vossen, editor. *EuroWordNet: A Multilingual Database with Lexical Semantic Networks*. Kluwer Academic Publishers, 1998.

[233] Holger Wache, Thomas Voegele, Ubbo Visser, Heiner Stuckenschmidt, Gerhard Schuster, Holger Neumann, and Sebastian Hübner. Ontology-based integration of information - a survey of existing approaches. In *Proceedings of the Workshop on Ontologies and Information Sharing at the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 108–117, 2001.

[234] Jun Wang and Les Gasser. Mutual on-line ontology alignment. In *Proceedings of the Workshop on Ontologies in Agent Systems (OAS) at the International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, 2002.

[235] Christopher A. Welty and Nicola Guarino. Supporting ontological analysis of taxonomic relationships. *Data and Knowledge Engineering*, 39(1):51–74, 2001.

[236] Ilya Zaihrayeu. *Towards Peer-to-Peer Information Management Systems*. PhD thesis, International Doctorate School in Information and Communication Technology, University of Trento, March 2006.

[237] Stefano Zanobini. *Semantic coordination: the model and an application to schema matching*. PhD thesis, International Doctorate School in Information and Communication Technology, University of Trento, March 2006.

[238] Anna Zhdanova and Pavel Shvaiko. Community-driven ontology matching. In *Proceedings of the European Semantic Web Conference (ESWC)*, pages 34–49, 2006.

[239] Antoine Zimmermann, Markus Krötzsch, Jérôme Euzenat, and Pascal Hitzler. Formalizing ontology alignment and its operations with category theory. In *Proceedings of the International Conference on formal ontologies for information systems (FOIS)*, 2006.

[240] Sagit Zohar. Schema-based data translation. Master's thesis, Tel-Aviv University (IL), 1997.