D2.2.3: State of the art on ontology alignment

Coordinator: Jérôme Euzenat (INRIA)
Thanh Le Bach (INRIA), Jesús Barrasa (UP. Madrid), Paolo Bouquet (U. Trento), Jan De Bo (VU Brussels), Rose Dieng (INRIA), Marc Ehrig (U. Karlsruhe), Manfred Hauswirth (EPF Lausanne), Mustafa Jarrar (VU Brussels), Ruben Lara (U. Innsbruck), Diana Maynard (Sheffield U.), Amedeo Napoli (CNRS/INRIA), Giorgos Stamou (NTU Athens), Heiner Stuckenschmidt (VUAmsterdam), Pavel Shvaiko (U. Trento), Sergio Tessaris (FU Bolzano), Sven Van Acker (VU Brussels), Ilya Zaihrayeu (U. Trento)

Abstract.
In this document we provide an overall view of the state of the art in ontology alignment. It is organised as a description of the need for ontology alignment, a presentation of the techniques currently in use for ontology alignment and a presentation of existing systems. The state of the art is not restricted to any discipline and consider as some form of ontology alignment the work made on schema matching within the database area for instance.
Keyword list: schema matching, ontology matching, ontology alignment, ontology mapping, schema mapping,

Document Identifier	KWEB/2004/D2.2.3/v1.2
Project	KWEB EU-IST-2004-507482
Version	v1.3
Date	August 3, 2004
State	final (revised)
Distribution	public

Knowledge Web Consortium

This document is part of a research project funded by the IST Programme of the Commission of the European Communities as project number IST-2004-507482.

University of Innsbruck (UIBK) - Coordinator
Institute of Computer Science
Technikerstrasse 13
A-6020 Innsbruck
Austria
Contact person: Dieter Fensel
E-mail address: dieter.fensel@uibk.ac.at

École Polytechnique Fédérale de Lausanne (EPFL)
Computer Science Department
Swiss Federal Institute of Technology
IN (Ecublens), CH-1015 Lausanne
Switzerland
Contact person: Boi Faltings
E-mail address: boi.faltings@epfl.ch

France Telecom (FT)
4 Rue du Clos Courtel
35512 Cesson Sévigné
France. PO Box 91226
Contact person : Alain Leger
E-mail address: alain.leger@rd.francetelecom.com

Freie Universität Berlin (FU Berlin)
Takustrasse 9
14195 Berlin
Germany
Contact person: Robert Tolksdorf
E-mail address: tolk@inf.fu-berlin.de

Free University of Bozen-Bolzano (FUB)
Piazza Domenicani 3
39100 Bolzano
Italy
Contact person: Enrico Franconi
E-mail address: franconi@inf.unibz.it

Institut National de Recherche en Informatique et en Automatique (INRIA)
ZIRST - 655 avenue de l'Europe -
Montbonnot Saint Martin
38334 Saint-Ismier
France
Contact person: Jérôme Euzenat
E-mail address: Jerome.Euzenat@inrialpes.fr

Centre for Research and Technology Hellas / Informatics and Telematics Institute (ITI-CERTH)
1st km Thermi - Panorama road
57001 Thermi-Thessaloniki
Greece. Po Box 361
Contact person: Michael G. Strintzis
E-mail address: strintzi@iti.gr

Learning Lab Lower Saxony (L3S)
Expo Plaza 1
30539 Hannover
Germany
Contact person: Wolfgang Nejdl
E-mail address: nejdl@learninglab.de

National University of Ireland Galway (NUIG)
National University of Ireland
Science and Technology Building
University Road
Galway
Ireland
Contact person: Christoph Bussler
E-mail address: chris.bussler@deri.ie

The Open University (OU)
Knowledge Media Institute
The Open University
Milton Keynes, MK7 6AA
United Kingdom
Contact person: Enrico Motta
E-mail address: e.motta@open.ac.uk

Universidad Politécnica de Madrid (UPM)
Campus de Montegancedo sn
28660 Boadilla del Monte
Spain
Contact person: Asunción Gómez Pérez
E-mail address: asun@fi.upm.es

University of Karlsruhe (UKARL)
Institut für Angewandte Informatik und Formale
Beschreibungsverfahren - AIFB
Universität Karlsruhe
D-76128 Karlsruhe
Germany
Contact person: Rudi Studer
E-mail address: studer@aifb.uni-karlsruhe.de

University of Liverpool (UniLiv)
Chadwick Building, Peach Street
L697ZF Liverpool
United Kingdom
Contact person: Michael Wooldridge
E-mail address: M.J.Wooldridge@csc.liv.ac.uk

University of Manchester (UoM)
Room 2.32. Kilburn Building, Department of Computer
Science, University of Manchester, Oxford Road
Manchester, M13 9PL
United Kingdom
Contact person: Carole Goble
E-mail address: carole@cs.man.ac.uk

University of Sheffield (USFD)
Regent Court, 211 Portobello street
S14DP Sheffield
United Kingdom
Contact person: Hamish Cunningham
E-mail address: hamish@dcs.shef.ac.uk

University of Trento (UniTn)
Via Sommarive 14
38050 Trento
Italy
Contact person: Fausto Giunchiglia
E-mail address: fausto@dit.unitn.it

Vrije Universiteit Amsterdam (VUA)
De Boelelaan 1081a
1081HV. Amsterdam
The Netherlands
Contact person: Frank van Harmelen
E-mail address: Frank.van.Harmelen@cs.vu.nl

Vrije Universiteit Brussel (VUB)
Pleinlaan 2, Building G10
1050 Brussels
Belgium
Contact person: Robert Meersman
E-mail address: robert.meersman@vub.ac.be

Work package participants

The following partners have taken an active part in the work leading to the elaboration of this document, even if they might not have directly contributed to writing parts of this document:

Centre for Research and Technology Hellas
École Polytechnique Fédérale de Lausanne
Free University of Bozen-Bolzano
Institut National de Recherche en Informatique et en Automatique
National University of Ireland Galway
Universidad Politécnica de Madrid
University of Innsbruck
University of Karlsruhe
University of Manchester
University of Sheffield
University of Trento
Vrije Universiteit Amsterdam
Vrije Universiteit Brussel

Changes

Version	Date	Author	Changes
0.9	12.03.2004	Jérôme Euzenat	creation
1.0	15.06.2004	Jérôme Euzenat	delivered to quality assessor
1.1	07.07.2004	Jérôme Euzenat	delivered to quality controler
1.2	02.08.2004	Jérôme Euzenat	final version

Executive Summary

Heterogeneity problems on the semantic web can be solved, for some of them, by aligning heterogeneous ontologies. This is illustrated through a number of use cases of ontology alignment.

Aligning ontologies consists of providing the corresponding entities in these ontologies. This process is precisely defined in deliverable D2.2.1. The current deliverable presents the many techniques currently used for implementing this process. These techniques are classified along the many features that can be found in ontologies (labels, structures, instances, semantics). They resort to many different disciplines such as statistics, machine learning or data analysis. The alignment itself is obtained by combining these techniques towards a particular goal (obtaining an alignment with particular features, optimising some criterion). Several combination techniques are also presented.

Finally, these techniques have been experimented in various systems for ontology alignment or schema matching. Several such systems are presented briefly in the last section and characterized by the above techniques they rely on.

The conclusion is that many techniques are available for achieving ontology alignment and many systems have been developed based on these techniques. However, few comparisons and few integration is actually provided by these implementations. This deliverable serves as a basis for considering further actions along these two lines. It provide a first inventory of what should be evaluated and suggests what evaluation criterion can be used.

Contents

List of definitions

Chapter 1

Introduction

Like the Web, the semantic Web will be distributed and heterogeneous. As such, the integration of resources found on the semantic Web is one of its main problems. To develop a solution of this problem, data will be expressed in the framework of ontologies. However, ontologies themselves can be heterogeneous and some work will have to be done to restore interoperability.

Even with emerging web and ontology standards, coordinating ontology development – whether manual or automatic – will prove to be a challenging task. In evolving domains, it is expected that ontologies will not remain static and various versions of ontologies will have to be tracked. Interdisciplinary ontologies may need to be created from existing domain-specific ontologies, domain-specific ontologies may need to be merged with more general ontologies, different versions of a single-domain ontology may need to be merged, and new information may need to be merged with existing ontologies. Furthermore, new ontologies may be built by merging information from heterogeneous databases or other information sources. Hence, these ontologies will have to be reconciliated.

Semantic interoperability can be grounded in ontology reconciliation: finding relationships between entities belonging to different ontologies. We call this process "ontology alignment". Alignment results can be used for various purposes such as displaying the correspondences, transforming one source into another, creating a set of bridge axioms or rules between the ontologies [Bouquet and Serafini, 2003], or generating query wrapper (query rewriting instead of transformation).

The ontology alignment problem can be described in one sentence: given two ontologies which describe each a set of discrete entities (which can be classes, properties, rules, predicates, etc.), find the relationships (e.g., equivalence or subsumption) holding between these entities. This is more precisely defined in the companion Framework deliverable (D2.2.1 and D2.2.2). The framework proposes an external definition of alignment precising its input, output and parameters.

The purpose of this deliverable is to present what is existing and ongoing in these various contexts so that research in ontology alignment can progress towards more complete, more integrated and especially more efficient means of solving heterogeneity problems through alignment. It thus complements the Framework deliverable by uncovering the internal parts of the ontology alignment process.

In the following, a number of use cases for ontology alignment are presented, justifying the high importance of this topic in the context of the semantic web (§2). Then the internals of known alignment methods are reviewed as classified under local methods (assessing the correspondence between two entities: §3) and global methods (establishing alignments from the results of local comparison: §4). We end by a presentation of existing systems and characterize them with regards to the techniques proposed in the former sections (§5).

Chapter 2

Use cases

In order to motivate the interest for ontology alignment in the context of the semantic web and semantic web services, we present here a number of use cases. Each case is presented through its context, the heterogeneity problems raised by the interoperation of knowledge resources in this context and illustrates the potential solution to these problems offered by alignment. They are not necessarily implemented use cases.

2.1 Agent communication

Agents are computer entities characterized by their autonomy and capacity of interaction. They are often divided in cognitive agents and reactive agents. Reactive agents implement a simple behavior and the strength of these agents is their capacity to let a global behavior emerge from the individual behavior of many such agents. Cognitive agents have a rather more elaborate behavior often characterized by the ability to pursue goals, to plan their achievement and to negociate with other agents in order to achieve their goals.

Reactive agents are often used in applications in which the solution is not known in advance and a large number of agents are used for covering a large search space. Cognitive agents are rather used when a number of rational rules leading to a solution are known. In both case, the common ground is that these agents are autonomous and can adapt to their environment fairly easily. Software agents are often used for programming applications with agents that are intermediate between purely reactive and and fully cognitive.

The current models of cognitive agents considers their behavior as a set of beliefs, desires and intentions symbolically expressed and use speech-act inspired languages for interacting with each others. These languages determine the “enveloppe” of the messages and enable agents to position them within a particular interaction contexts. But they do not specify the actual content of the message which can be expressed by various content languages. In order to help them, currrent standards for expressing these messages provides slots for declaring the content language and the ontology used.

As a consequence, when two autonomous and independently designed agents meet, they have the possibility to exchange messages but little chance to understand each others if they do not share the same content language and ontology. It is thus necessary to provide the opportunity for these agents to align their ontologies in order to either translate their messages or integrate bridge axioms in their own models.

One solution to this problem would be to have an ontology alignment protocol able to be interleaved with any other agent interaction protocol and which could be triggered upon receiving a message expressed in an alien ontology. Such a protocol would allow communication with tier alignment services in order to:

- browse alignment libraries;
- ask for the (partial) alignment of some ontology pair;
- ask for the translation of a message given an alignment;
- ask for a translation program, bridge axioms or view definition given an alignement;
- ask for the completion of a partial alignment.

In addition, it should allow an agent to ask for agreement on some returned alignment or translate a message expressed with regard to some private ontology in a public one.

In such a case, the dialog between two agents could become:

agent1 sends a message m expressed with a private ontology o;
agent2 asks for a translation of m with regard to a public ontology o';
agent1 sends m' expressed with regard to o';
agent2 sends to **align** a request to align o' with ontology o'' with regard to message m;
align replies by partial alignment a;
agent2 asks **align** to translate m' thanks to alignment a;
align replies by m";
agent2 asks **align** to complete alignment a with terms in messge n;
align replies with alignment a';
agent2 asks align a translation program made from alignment a';
align replies with program p;
agent2 applies p to n and send the result as an answer to **agent1**.

2.2 Emergent Semantics

In what follows, we summarize the status of a collaborative effort on the development of the notion of "emergent semantics", which has been initiated by the IFIP 2.6 Working Group on Data Semantics. This summary is based on [Aberer *et al.*, 2004b] and [Aberer *et al.*, 2004a].

This approach is motivated by the belief that global semantic interoperability emerges from large numbers of purely local, pair-wise interactions. "Semantic interoperability is viewed as an emergent phenomenon constructed incrementally, and its state at any given point in time depends on the frequency, the quality and the efficiency with which negotiations can be conducted to reach agreements on common interpretations within the context of a given task". This type of semantic interoperability is called "emergent semantics".

The key principles of this approach are:

- Agreements as a Semantic Handshake Protocol. Emergent semantics ("Dynamic ontologies") can be established on the bases of mutually accepted propositions between the interacting agents. The quality of "emerged semantics" depends on the strength of the agreed propositions, and their trustworthiness.

- Agreements emerge from negotiations. Information environments are assumed dynamic. Thus, interactions between agents are necessary to identify and resolve semantic conflicts, and to verify whether a consensus leads to the expected actions. Interactions are message exchanges or references to distributed information resources.

- Agreements emerge from local interactions. Emergent semantics are assumed to be established incrementally, based on local agreements. Global agreements are obtained through aggregations of local agreements.

This approach is currently active in the area of peer-to-peer data management and integration, where local schema mappings are introduced in order to enable semantic interoperability. Local schema mappings can be seen as the local communication mechanisms for establishing consensus on the interpretation of data.

While the Semantic Web approach uses ontologies for obtaining semantic interoperability, the ambition of the emergent semantics approach is to obtain such interoperability in a more scalable and decentralized fashion, without necessarily using ontologies.

2.3 Web service integration

We understand web service discovery as the process of finding web services that fulfil a given requester goal. Both the requester goal and the service capability i.e. requested functionality and provided functionality are defined declaratively and in a machine-processable way.

Both the goal and the capability will be described using one or more domain-specific ontologies. Here we can find two different problems:

1. The descriptions of the capability or the goal use several domain ontologies that have to be mediated, as conflicts can arise, either from the conceptual model or from the ontology language used.

2. The capability and the goal are expressed using ontologies that describe a common domain, but still the ontologies used for the capability are different from the ones used for the goal. In this case, the discovery process still has to find suitable services despite the use of different terminologies for the goal and the capability.

Using the ontologies defined in [1] for train connections, locations, purchase orders and date and time, we illustrate here concrete problems for 1) and 2)

1a Use of different ontology languages The train connections ontology (Listing 1 in note 1) imports an ontology for persons described in OWL[2] and two ontologies described in WSML (F-Logic) for locations, and date and time. In the first case, a simple import is not possible as the ontologies are described in different languages (WSML and OWL). Therefore, the heterogeneity of ontology languages has to be overcome.

1b Changes on the conceptual model The capability described of a web service selling train tickets make use of the train connections ontology and the purchase order ontology (Listing 3 in note 1). This capability wants to express that the items of the trade it establishes are train trips. However, the range of the “items” property of the “trade” concept in the purchase order ontology is the concept “product”. For that reason, when importing the ontologies the capability has to say that “train trip” is a subconcept of “product”.

[1]http://www.wsmo.org/2004/d3/d3.2/

[2]http://daml.umbc.edu/ontologies/ittalks/person

1c Using different ontologies for the same domain A capability of a service selling flight tickets can use an airport codes ontology[3] which defines “city” as a property of the “AirportCode” concept, and the same time use a different ontology for countries[4], in which “country” is a concept and not a property. These two ontologies have to be used in a consistent way, resolving possible conflicts.

2a The goal is described using the locations ontology (Listing 4 in note 1) which defines the concept “Country” as an extension of the concept “Country” in the CIA factbook[5]. The service capability is described using an airport code resource[6], and it uses the “Country” property of the “AirportCode” concept. If the goal says that a service providing a flight from the country “Austria” to the country “Spain” is required, and the capability offers flights from airports with the property “Country” with value “Austria” to airports with the property “Country” with value “Spain”, a match have to be established regardless of the different conceptual model.

The heterogeneity problems above must be solved in order to enable the reuse of (possibly conflicting) ontologies for goal and capability descriptions, and in order to enable the match of goals and capabilities providing compatible functionalities but described using heterogeneous ontologies.

2.4 Ontology-driven data integration: The fund finder application

Our use case is about migrating relational database content to the semantic web allowing the integration of information from multiple and heterogeneous sources. Typically the input to this kind of problem is a set of n databases, each of them containing the data to be migrated and an ontology to be populated with instances extracted from the databases.

The important idea behind our approach is that mappings between the database SQL schema elements (tables, columns and keys) and the corresponding concepts, relations and attributes of the ontology will be defined declaratively in a mapping document. This mapping document will be the input of a processor charged of carrying out the effective migration in an automatic way. The fact of defining these mappings declaratively will make our solution domain independent and reusable. Another important aspect in our approach is that it uses the database and ontology as they are, we do not create the ontology from the database schema, that’s why some complex mapping situations may arise. Basically, the level of complexity of the mappings to be defined will depend on the level of similarity of the ontology’s model and the database schema. Normally, one of them will be richer, more generic or specific, better structured, etc., than the other.

We have created R2O, an extensible and fully declarative language to describe mappings between relational database schemas and ontologies. R2O is intended to be expressive enough to describe the semantics of these mappings and is proposed as a DBMS independent high level language that can work with any database implementing the SQL standard. The R2O language allows the definition of explicit correspondences between components of two models. A basic approach to the ideas underlying R2O is showed by the following expression:

[3]http://www.daml.ri.cmu.edu/ont/AirportCodes.daml
[4]http://www.daml.org/2003/09/factbook/factbook-ont
[5]http://www.daml.org/2003/09/factbook/factbook-ont
[6]http://www.daml.ri.cmu.edu/ont/AirportCodes.daml

$$OntologyComponent_i = Transformation(DatabaseComponent_j, DatabaseComponent_k?)$$

Where $OntologyComponent_i$ is any concept, attribute or relation in the target ontology and $DatabaseComponent_j$ is any database table or column.

A mapping between a database schema and an ontology can then be defined as a set of basic mapping expressions or mapping elements between components in both models like the one showed before. Details on the language and the use case can be found at [Barrasa *et al.*, 2003].

This approach has been implemented and tested with the Fund Finder application[7] which has been developed in the context of the ESPERONTO project. In our particular case, the database we want to migrate (FISUB) contains incentives and funds provided by the Catalan and Spanish Governments and by the European Union, for companies or entrepreneurs located in the Spanish region of Catalonia. It contains more than 300 registers that are updated manually on a daily basis. The reason why we want to migrate these contents to the Semantic Web is to be able to aggregate to them information from other web resources related to funding in the European Union and to allow web users to ask intelligent queries about funding resources according to some parameters like their profile, to look for complementary ones, to check compatibilities and incompatibilities between types of funding, and so on.

2.5 Catalog matching

Many e-Commerce application are based on the publication of electronic catalogs which describe the goods on sale and allow customers to select the goods they need. However, a very important obstacle to the success of distributed e-Commerce applications is the problem of interoperating different catalogs. Indeed, many systems require participant parties to perform very costly operations on their local catalogs to enter into the system, and this is a tremendous barrier for newcomers. Some typical instances of this situation are:

e-Marketplaces electronic malls where different sellers provide their goods in a common environment. The problem is that each provider typically owns a local catalog, in which goods are organized according to criteria that suit its internal business processes. However, to take part in the marketplace, providers should translate their catalogs into a common catalog, which will be presented to users as a single access point to what is sold in the marketplace. Notice that, in principle, this translation is required for each marketplace in which a company is involved, which means that a potentially very high number of catalogs should be maintained at the same time by each company. This is considered one of the strong barriers against the success of eMarketplaces.

products and services reclassification there are some efforts in trying to harmonize catalogs through the adoption of standard classification structures. Well-known examples are the United Nations Standard Products and Services Code - UNSPSC[8] and eCL@ss[9]. The problem is that adopting one of these standards would require companies to translate their catalogs into the standard one, and this can be a very costly activity, which would impact the management of their internal business processes.

[7]http://www.esperonto.net/fundfinder
[8]http://www.unspsc.org/
[9]http://www.eclass-online.com/

aggregation of buyers' needs in many e-Procurement scenarios, it would be important for buyers to aggregate their product demand to get some advantage in terms of supply conditions or buying power. To support the aggregation process, the system should be able to identify groups of buyers interested in a similar category of product, and possibly to suggest buyers how to modify some requested features to increase the advantages of aggregation. This should be possible without presupposing that differnet organizations share the same classification system.

The scenarios above (and many others) would be much more appealing (or would simply become viable) if we could provide means for aligning catalogs through rich mappings which allow the system to compare demand and offer. Moreover, in some applications (e.g. aggregation of buyers' needs), this mapping process should be (semi or fully) automatic, and runtime.

Notice that catalogs are a significant challenge for alignment, as catalogs are not simple concept hierarchies (classifications), but classifications in which classes may have multiple attributes. For example, a shirt has a size, a color and a price. Therefore, mappings should align not only the concept of shirt with an equivalent concept on another catalog, but should take into account the alignement of attribute names and values.

2.6 Information retrieval from heterogeneous multimedia databases

Digital archiving of multimedia content including video, audio, still images and various types of multimedia documents has been recognized by content holding organizations as a mature choice for the preservation, preview and partial distribution of their assets. The advances in computer, data networks and web technologies along with the success of standardization efforts of MPEG and JPEG boosted the movement of the archives towards the conversion of their fragile and manually indexed material to digital, web accessible data. By the end of last century the question was not on whether digital multimedia archives are technically and economically viable, but rather on how they would be efficient and informative. In this framework, different scientific fields such as database management systems, multimedia information processing, artificial intelligence, etc., have observed a close cooperation with each other during the last few years. The attempt has been to develop intelligent and efficient information retrieval systems, enabling the user to access vast amounts of heterogeneous multimedia information, stored in different sites and archives. Lately, using the advanced technologies of multimedia standardization, large databases and semantic web, access to heterogeneous multimedia archives is done throughout the following steps: - construction of large multimedia databases providing the raw multimedia information (images, video, audio, text, etc.) and its annotation (syntactic and semantic description) given in standardized (MPEG-4, MPEG-7, MPEG-21 etc) or not standardized metadata, - construction of ontologies providing the semantics of the above metadata, - construction of mediators (advanced search engines) unifying the multimedia annotation and the user access to heterogeneous multimedia content. Ontology and data alignment will play an important role in the above framework. It has become clear among the research community dealing with content-based multimedia data retrieval and new emerging related standards, that the results to be obtained will be ineffective, unless major focus will be given to the semantic unification of the heterogeneous content annotations. Algorithms and tools that find the correspondence between the semantics of metadata stored in the database of each archive will be based on the alignment of the ontologies providing these semantics.

2.7 P2P information sharing

Peer-to-Peer (P2P) information sharing systems have recently received a lot of attention both from the academia and industry. P2P file sharing systems already have a variety of implementations and are widely used on the Web (for instance, Kazaa and Morpheus have more than 450 million of downloads (see http://www.download.com, March 2004). Some of these applications provide a simple schema in order to describe file contents (e.g., Kazaa, Napster), which is shared by all parties and can not be changed locally at some party. Other P2P file sharing applications do not provide any schema at all (e.g., Gnutella), and encapsulate semantic information into the file names.

P2P information sharing systems which use rather complex structures to describe data (e.g., database schemas, ontologies in distributed knowledge management) have been largely studied in the academia but, to our knowledge, did not go far beyond prototypical test beds. The main underlying idea of these approaches is that peers define pair-wise mappings between their schemas (or ontologies) and use these mappings in order to propagate requests and data. Most of the above mentioned works are based on the assumption that the mappings between peer schemas are already defined a priori, and they do not consider how the mappings are actually created.

A real life example for P2P databases could be the real estate agents example. Agents coordinate their databases in exchanging real estate information with the goal of pushing sales. Different agents may use different schemas to describe their data, so that they establish mappings between their schemas to enable interoperation. Since they travel to their customers (who may want to sell or, instead, to buy), they always carry relevant data with them. When one is on the site of a customer, who wants to sell a house, the agent updates his database and makes this data available for other agents. Or, when an agent talks with a potential buyer, and nothing from the agent's database satisfies the client, the agent may want to query other agents' databases to look for additional sale options.

In P2P settings assumptions that all parties agree on the same schema, or that all parties rely on one global schema (as in data integration) can not be made. Peers come and go, import multiple schemas into the system, and have a need to interoperate with other nodes at runtime. In this activity we see schema alignment as the main process to enable nodes' interoperation. Namely, when two peers "meet" on the network, they establish mappings between their schemas in a (semi) automatic alignment discovery process.

Automation of the schema alignment discovery process will create a great advance for the P2P information sharing systems on the Semantic Web. Peers will be able to bring to the system various schemas, "align" them to the schemas of other peers on the network with no (or minimal) user involvement, and exchange requests and data in a decentralized, collaborative manner. So far the heterogeneity problem has been the main obstacle for the P2P applications with "rich" heterogeneous schemas to flow into the Web infrastructure (as in the case of file sharing systems). But the advance of the Semantic Web technologies allows us to make an optimistic assumption that it will be the case in the nearest future.

2.8 GridVine: aligning schemas in overlay networks

P2P systems construct so-called *overlay networks* over the physical network. In principle, applications could use the services provided by the networking layer to locate their resources of interest. However, having a separate, application-specific overlay network has the advantage of supporting application-specific identifiers and semantic routing and offers the possibility to provide additional services for supporting network maintenance, authentication, trust, etc., all of which would be hard to integrate into and support at the networking layer. The introduction of overlay networks is probably the essential innovation of P2P systems.

In the GridVine use case we want to build a semantic overlay network which provides structured search and (semi-) automatic schema integration based on the semantic gossiping approach [Aberer *et al.*, 2003b] as described in Section 4.3.3. As a case study we will use the domain of image annotation. The setup we assume is as follows: People take pictures with digital cameras and want to share them with others via a P2P system. By default digital cameras store images under cryptic names not related to the image content and can add technical annotations, for example, the date and the time the photo was taken. In GridVine we want to allow the user to freely define application-specific schemas, for example, annotated photos with meaningful names, descriptions, or any other data they consider meaningful, without imposing a global schema. This assumption has been shown to be realistic since in P2P systems with millions of users and no authority that can enforce standards, it is impossible to impose global schemas. Additionally, a global schema may neglect useful user knowledge that could otherwise be provided.

Concretely, users will be able to define individual schemas in RDFS to describe their photo images. These schemas are indexed by the P2P system and thus can be found and retrieved by any participant via RDQL. If a user is interested to integrate his/her schema with the schema of another user he/she can provide a translation in OWL. Thus, a network of semantic translations among end-user schemas will emerge. Again these translations are indexed in the P2P system and can be retrieved by other participants which serves as the basis to support semantic gossiping and schema integration.

In resolving translation links and assessing their quality we will investigate iterative and recursive approaches. In iterative resolution the peer issuing the RDF query will try to find all translations links itself: The peer issues a query for the translation of a certain concept. Upon finding a translation, it translates the original query using the found translation (Query') and issues a search for the transformed query. Furthermore, the gossiping peer will issue a query for a translation of the translation (Predicate'). This continues until no more translations are available or the transformed query is considered as being too different from the original query following syntactic and semantic similarity values [Aberer *et al.*, 2003b].

In recursive resolution, the issuing peer tries to resolve the translation by delegating rather than doing it itself: First, it looks for translations of the concepts used in the query and translates the query upon finding a translation. The transformed query is issued and results for the query will be returned to the issuer of the query. The receiver of the transformed query will follow the exact same procedure, and so on recursively.

In the case study we will investigate the applicability of our semantic gossiping approach in a practical setting. We will implement GridVine on top ouf our P-Grid P2P system [Aberer *et al.*, 2003a] and perform experiments to assess the achievable quality of schema integration and to obtain performance and scalability figures of the approach.

2.9 Personnal information delivery

These days Internet radio is becoming reality. A number of audio feeds are now available on the web. Philips is already working on a stand alone media station supporting internet radio streams.

At the moment internet radio is not really different from conventional radio concerning the limited ability to influence the content of the radio program. The idea of smart internet radio is now to exploit the possibilities of web technology, in order to provide internet radio that is customized to match the users interests by combining different audio streams and selecting the most appropriate stream. The idea is that the based on model of the users interests an appropriate audio stream is selected and played.

Besides more technical problems like dealing with different audio formats and timing of audio sequences, there are some conceptual problems to be addressed. These problems originate from the need to compare user interests with metadata provided with the audio streams. Besides information about artists, songs or artists are normally assigned to different musical genres. In principle, information about genres can be used to select songs the user is likely to be interested in. The problem with this approach is that there is neither a universal agreement on a fixed set of genres nor is there an agreement on the correct classification of songs into genres. The idea is now to use semantic web technologies, in particular explicit representation of terminologies and their intended meanings. While hierarchies of genres can often be found on the web sites of audio content providers such as MusicMoz[10] or AllMusic[11] the classification of music genres that reflects the opinion of a user can often be obtained from the users file systems. The corresponding integration problem is two-fold:

- The categorizations of audio sources have to be matched against the users interests reflected in his or her personal categorization of music
- The categorizations of different information providers have to be compared in order to be able to use matches and user feedback on a single source to draw conclusions about other sources

The specific problem of an alignment of genre information lies in the fact that there is no agreement about the right categorization of artists or even songs into genres. This categorization can be different from person to person. On the other hand, linguistic approaches for comparing categories are doomed to fail due to the artificial nature of genre names that do not have a connection to the standard semantics of natural language.

2.10 Vertical Publishing in Life Science Domain

Innovative research institutes rely on the availability of complete and accurate information about new research and development, and it is the business of information providers such as Elsevier to provide the required information in a cost-effective way. It is very likely that the semantic web will make an important contribution to this effort, since it facilitates access to an unprecedented quantity of data. However, with the unremitting growth of scientific information, integrating access to all this information remains a significant problem, not least because of the heterogeneity of the information sources involved – sources which may use different syntactic standards (syntactic heterogeneity), organize information in very different ways (structural heterogeneity) and even use different terminologies to refer to the same information (semantic heterogeneity). The ability to address these different kinds of heterogeneity is the key to integrated access.

[10] http://musicmoz.org/Styles/
[11] http://www.allmusic.com/mus_Styles.html

Thesauri have already proven to be a core technology to effective information access as they provide controlled vocabularies for indexing information, and thereby help to overcome some of the problems of free-text search by relating and grouping relevant terms in a specific domain. Two examples of thesauri in the life sciences are MeSH[12], which is produced by the U.S. National Library of Medicine (NLM) and Elsevier's life science thesaurus EMTREE[13]. In the medical area a lot of work has been done on the definition and standardization of terminologies. The result of these efforts is a large number of medical thesauri such as MeSH. The complexity of the terminologies used in medicine and the strong need for quality control has also lead to the development of ontologies that feature complex concept definition. Some of these ontologies are available in OWL and can be seen as the first OWL applications that have a use in real life applications. The need for an integration of exiting thesauri and ontologies has been widely recognized in the medical area leading to a number of efforts for defining standardized terminologies. It is, however, also acknowledged by the literature, that the creation of a single universal terminology for the medical domain is neither possible nor beneficial, because different tasks and viewpoints require different, often incompatible conceptual choices. As a result a number of communities of practice have been evolved that commit to one of the proposed standards. This situation demands for a weak for of integration, also referred to as alignment in order to be able to exchange information between the different communities.

This implies that terminology alignment is important for new publishing models known as "vertical publishing". Different from the traditional model where information providers sell predefined source information (e.g., in terms of a medical journal) in vertical publishing, the information provider sells information about a certain topic independent from the source (e.g., by retrieving relevant articles from different journals). As relevant information may be provided by different communities, the respective terminologies have to be aligned and compared with the terminology of the customer in order to provide all relevant information.

[12]http://www.nlm.nih.gov/mesh/meshhome.html

[13]http://www.elsevier.com/homepage/sah/spd/site/

Chapter 3

Local methods

The main issue in aligning consist of finding to what entity or expression in one ontology corresponds another one in the other ontology. Here are presented the basic methods which enable to measure this correspondence at a local level, i.e., only comparing one element with another and not working at the global scale of ontologies.

Very often, this amounts to measuring a pair-wise similarity between entities (which can be as reduced as an equality predicate) and computing the best match between them, i.e., the one that minimizes the total disimilarity (or maximizes the similarity measure).

There are many different ways to compute such a dissimilarity with different methods designed in the context of data analysis, machine learning, language engineering, statistics or knowledge representation. Their condition of use depends of the objects to be compared, their context and sometimes the external semantics of these objects. Some of this context can be found in Figure 3.1 (From [Rahm and Bernstein, 2001a] enhanced in [Giunchiglia and Shvaiko, 2003; Shvaiko, 2004] and [Euzenat and Valtchev, 2003]) which decomposes the set of methods along two perspectives: the kind of techniques (descending) and the kind of manipulated objects (ascending).

After providing base definitions about ontologies and measures (§3.1), the outline of the chapter follows the lower classification of Figure 3.1: it is decomposed in terminological (§3.2), structural (§3.3), extensional (§3.4) and semantic methods (§3.5). Their combination is explored in next chapter and their use in actual systems presented in Chapter 5.

3.1 Ontologies and measures

In order to fix the vocabulary used in this deliverable, some definitions are given here concerning ontologies and similarities. They are only presented for the sake of completeness and can be skipped without harm by the knowledgeable reader.

3.1.1 Ontology language

We will not give a definition of ontology but rather consider that an ontology is expressed in an ontology language and alignment methods must work in function of these languages and their features. There are a large variety of languages for expressing ontologies (see [Staab and Studer, 2004]). It is not the purpose of this deliverable to present them all or to commit to one particular language. Fortunatelly, they most often share the same kind of entities (with different names

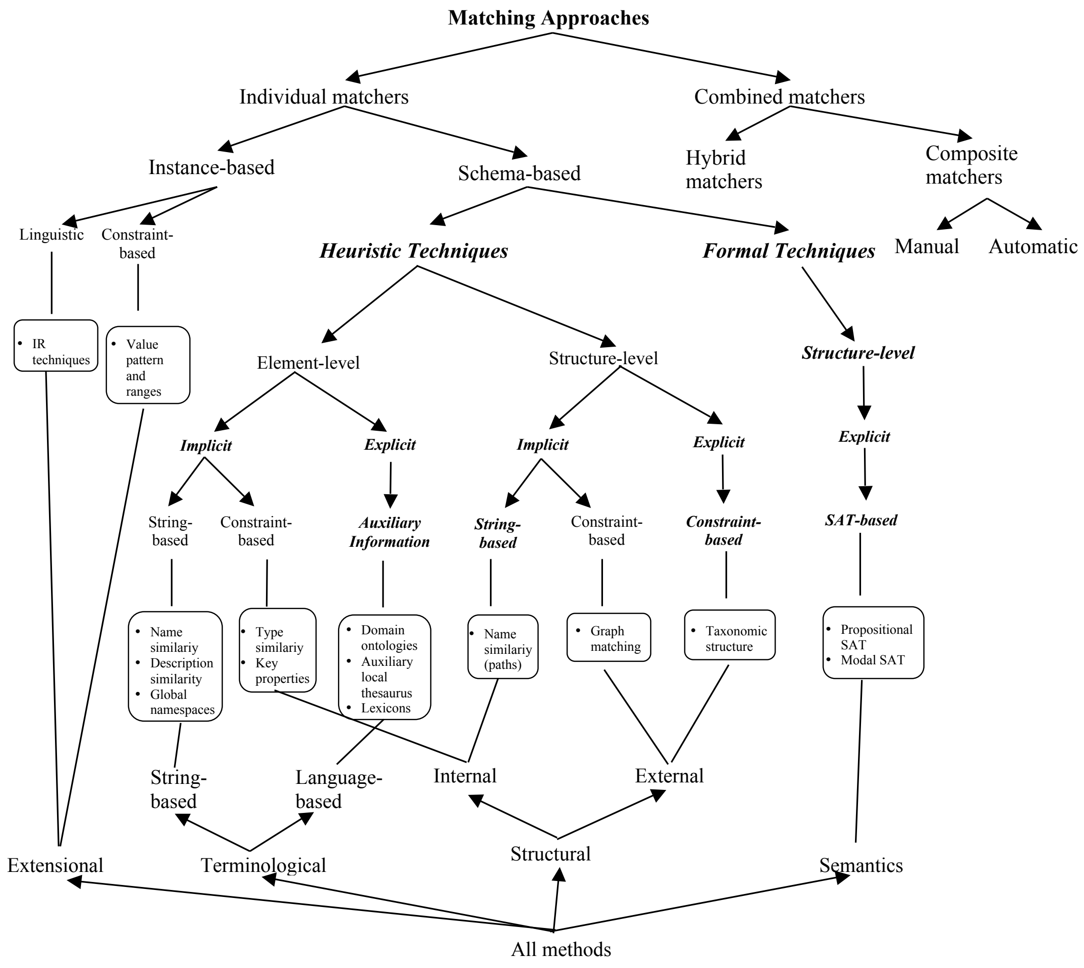


Figure 3.1: Classification of local methods.

but comparable interpretation). So, we very shortly describe here what entities are found in an ontology languages. This is the goal of the local methods to assess the correspondence of the entities in these languages. These entities are mainly:

classes or concepts are the main entities of an ontology. They are interpreted as a set of individuals in the domain.

objects or instances are interpreted as particular individual of a domain;

relations are the ideal notion of a relation independently to why it applies (e.g., the name relation in itself), they are interpreted as a subset of the products of the domain.

properties are the relations precisely applied to a class (the name of a man);

property instances are the relations applied to precise objects (the name of this individual)

datatypes are a particular part of the domain which specifies values (as opposed to individuals), values do not have identities;

datavalues are simple values.
property restrictions are the constraints applying to properties (they restrict the interpretation of the property for a class).

These entities are linked by various kinds of relationships that are also very common:

specialization (or subsumption) between two classes or two properties (interpreted as inclusion of the interpretations);
exclusion between two classes or two properties (interpreted as the exclusion of their interpretations, i.e., their intersection is empty);
instanciation (or typing) between objects and classes, property instances and properties, values and datatypes (which is interpreted as membership);
attribution between classes and properties, objects and property instances;
restriction expressing the restriction on a property in a class;
assignment of a property in an individual

Most of these features are found in modern ontology languages (e.g., OWL). Other kinds of constructs exist such as arbitrary formulas (or axioms). They are not discussed here.

3.1.2 Similarity and other measures

There are many ways to assess the similarity between two entities. The most common way amounts to defining a measure of this similarity. We present the characteristics which can be asked from these measures.

Definition 1 (Similarity). *A similarity $\sigma : O \times O \rightarrow \mathbb{R}$ is a function from a pair of entities to a real number expressing the similarity between two objects such that:*

$$\forall x, y \in O, \sigma(x, y) \geq 0 \qquad \textit{(positiveness)}$$
$$\forall x \in O, \forall y, z \in O, \sigma(x, x) \geq \sigma(y, z) \qquad \textit{(maximality)}$$
$$\forall x, y \in O, \sigma(x, y) = \sigma(y, x) \qquad \textit{(symmetry)}$$

The dissimilarity is a dual operation:

Definition 2 (Dissimilarity). *Given a set O of entities, a dissimilarity $\delta : O \times O \rightarrow \mathbb{R}$ is a function from a pair of entities to a real number such that:*

$$\forall x, y \in O, \delta(x, y) \geq 0 \qquad \textit{(positiveness)}$$
$$\forall x \in O, \delta(x, x) = 0 \qquad \textit{(minimality)}$$
$$\forall x, y \in O, \delta(x, y) = \delta(y, x) \qquad \textit{(symmetry)}$$

Some authors consider a "non-symmetric (dis)similarity", e.g. [Tverski, 1977], we will then use the term non-symmetric measure. There are more constraining notions of dissimilarity such as distance and ultrametrics.

Definition 3 (Distance). *A distance (or metrics) $\delta : O \times O \rightarrow \mathbb{R}$ is a dissimilarity function satisfying the definiteness and triangular inequality:*

$$\forall x, y \in O, \delta(x, y) = 0 \textit{ iff } x = y \qquad \textit{(definiteness)}$$
$$\forall x, y, z \in O, \delta(x, y) + \delta(y, z) \geq \delta(x, z) \qquad (triangular\ inequality)$$

Definition 4 (Ultrametrics). *Given a set O of entities, an ultrametrics is a metrics such that:*

$$\forall x, y, z \in O, \delta(x, y) \leq max(d(x, z), d(y, z)) \qquad \textit{(ultrametric inequality)}$$

Very often, the measures are normalized especially if measure of the similarity of different kind of entities must be compared. Reducing each value to the same scale (i.e., proportionnaly to the size of the considered space) is the common way to normalize.

Definition 5 (Normalized (dis)similarity). *A (dis)similarity is said to be* normalized *if it ranges over the unit interval of real numbers* $[0\ 1]$. *A normalized version of a (dis)similarity σ (resp. δ) will be noted $\overline{\sigma}$ (resp. $\overline{\delta}$).*

It is easy to see that to any normalized similarity $\overline{\sigma}$ corresponds a normalized dissimilarity $\overline{\delta} = 1 - \overline{\sigma}$ and vice-versa.

In the remainder, we might consider only normalized measures. We will assume that a dissimilarity function between two entities must return some real number between 0 and 1.

3.2 Terminological methods

Terminological methods compare strings. They can be applied to the name, the label or the comments concerning entities to find those which are similar. This can be used for comparing class names and/or URI.

Throughout this section, the set $\mathbb{S}$ will represent the set of strings, i.e., the sequences of letters over an alphabet $\mathbb{L}$ (so, $\mathbb{S} = \mathbb{L}*$). The empty string is noted ϵ, and $\forall s, t \in \mathbb{S}$, $\forall c \in \mathbb{L}$ $s + t$ is the concatenation of the strings s and t (which will not be further defined). $|s|$ will be the length of the string s (i.e., the numbers of characters it contains). $s[i]$ for $i \in [1\ |s|]$ will be the letter in position i of s.

A string s is the substring of another t, if there exists two strings s' and s'' such that $s'+s+s'' = t$ (this is noted $s \in t$). Two string are equal ($s = t$) if and only if $s \in t$ and $t \in s$. The number of occurence of s in t (noted $s\#t$) is the number of distinct pairs s', s'' such that $s' + s + s'' = t$.

In the field of terminology, the relation between terms and concepts tends to be distinctly multivocal [Maynard, 1999], i.e. terms can refer to more than one concept, and a single term can have many variants, all related to a single concept. Although this is strictly prohibited by ISO Standard 704 (Principles and Methods for Terminology, 1987), modern terminological theory accepts that this is neither practical nor even desirable and rejects the rather narrow prescriptive view of the past in favour of communicative theories requiring different forms in different situational needs [Sager, 1990]. Such problems hold equally for any ontology – domain-specific or not — and for any kind of instance, not just for specialised terms. URIs used as names in the semantic web do not require that the names be unique, although identical names must refer to the same entity. So two objects may be identical, but have different URIs

If this problem exists even within a single ontology, it is increased tenfold when two or more ontologies are merged. There is no way that the use of terms can be expected to be consistent across merged ontologies, and different parts of the merged ontology may have conflucting or ambiguous elements within them, not just for concepts and instances, but also for relations.

There are two main categories of methods for comparing terms depending on their consideration of character strings only (§ 3.2.1) or if they use some linguistic knowledge (§ 3.2.2).

3.2.1 String-based methods

String-based methods take advantage of the structure of the string (as a sequence of letter). String-based methods will typically find as similar classes `Match` and `match`, but not `alignment`.

There are many ways to compare strings depending of the way the string is seen (as an exact sequence of letters, an erroneous sequence of letters, a set of letters, a set of words...). The most frequent are presented below. [Cohen *et al.*, 2003] compares various string-matching techniques, from distance like functions to token-based distance functions.

Normalization

Before comparing strictly strings which have meaning in (occidental) natural language, there are a number of normalization procedures that help improving the results of subsequent comparison:

Case normalisation consists of converting each alphabetic character in the strings in their downcase counterpart;

Diacritics suppression consists in replacing characters with diacritic signs with their most frequent replacement (e.g., replacing *Montréal* with *Montreal*);

Blank normalisation consists of normalising all blank characters (blank, tabulation, carriage return, our sequence of theses) into a single blank character;

Link stripping consists of normalizing some links between words (like replacing apostrophes and blank underline into dashes;

Digit suppression consists of suppressing digits (to be used with care, there are chemical names containing digits);

Punctuation elimination is useful when only words are considered and not sentences;

Stopword elimination eliminates words that can be found in a list (usually like, “to”, “a"...). This is usually used for comparing long texts.

String equality

String equality returns 0 if the string are not the same and 1 if they are the same.

Definition 6 (String equality). *String equality is a similarity* $\sigma : \mathbb{S} \times \mathbb{S} \rightarrow [0,1]$ *such that* $\forall x, y \in \mathbb{S}$, $\sigma(x,x) = 1$ *and* $if x \neq y, \sigma(x,y) = 0$.

It can be performed after some syntactic normalisation of the string (e.g., downcasing, encoding conversion, accent normalisation).

This measure does not tell how strings are different. A more immediate way of comparing two strings is the Hamming distance which counts the number of positions in which the two strings differ (we give here the version normalised by the length of the largest one).

Definition 7 (Hamming distance). *The Hamming distance is a dissimilarity* $\delta : \mathbb{S} \times \mathbb{S} \rightarrow [0,1]$ *such that:*

$$\delta(s,t) = \frac{(\sum_{i=1}^{min(|s|,|t|)} s[i] \neq t[i]) + ||s| - |t||}{max(|s|,|t|)}$$

Substring test

A number of variations can be obtained from the string equality such as considering that strings are very similar when one is a substring of another:

Definition 8 (Substring test). *Substring test is a similarity $\sigma : \mathbb{S} \times \mathbb{S} \rightarrow [0, 1]$ such that $\forall x, y \in \mathbb{S}$, if there exist $p, s \in \mathbb{S}$ such that $x = p + y + s$ or $y = p + x + s$, then $\sigma(x, y) = 1$, otherwise $\sigma(x, y) = 0$.*

This is obviously a similarity. This measure can be refined in a substring similarity which measures the ratio of the common subpart between two strings.

Definition 9 (Substring similarity). *Substring similarity is a similarity $\sigma : \mathbb{S} \times \mathbb{S} \rightarrow [0, 1]$ such that $\forall x, y \in \mathbb{S}$, let t be the largest common substring of x and y, $\sigma(x, y) = \frac{2|t|}{|x|+|y|}$.*

It is easy to see that this measure is indeed a similarity. One could also consider a subsequence similarity as well. This definition can be used for building function based on the largest common prefix or largest common suffix.

The N-gram distance is also well used in comparing strings with some robustness:

Definition 10 (N-gram distance). *Let $ngram(s, n)$ be the set of substrings of s (augmented with $n - 1$ irrelevant characters at the beginning and the end) of length n, the n-gram distance is a dissimilarity $\delta : \mathbb{S} \times \mathbb{S} \rightarrow \mathbb{R}$ such that:*

$$\delta(s, t) = |ngram(s, n) \cap ngram(t, n)|$$

The normalized version of this function is:

$$\overline{\delta}(s, t) = \frac{|ngram(s, n) \cap ngram(t, n)|}{n * min(|s|, |t|)}$$

This function is quite efficient when characters are only missing.

Edit distance

Generally speaking, an edit distance between two objects, is the minimal cost of operations to apply to one of the object for obtaining the other. The edit distance on strings (as known as Levenshtein distance) is the minimum number of insertions, deletions, and substitutions of characters required to transform one string into the other. Each operation much be assigned a cost.

Definition 11 (Edit distance). *Given a set Op of string operations ($op : \mathbb{S} \rightarrow \mathbb{S}$), and a cost function $w : Op \rightarrow \mathbb{R}$, such that for any pair of strings there exist a sequence of operations which transforms the first one into the second one (and vice versa), the edit distance is a dissimilarity $\delta : \mathbb{S} \times \mathbb{S} \rightarrow [0, 1]$ such that $\delta(s, t)$, is the cost of the less costly sequence of operations which transform s in t.*

$$\delta(s, t) = min_{(op_i)_I; op_n(\ldots op_1(s))=t}(\sum_{i \in I} w_{op_i})$$

In string edit distance, the operations usually considered are insertion of a character $ins(c,i)$, replacement of a character by another $sub(c,c',i)$ and deletion of a character $del(i,c)$. It can be easily checked that these operations are such that $ins(c,i) = del(i,c)^{-1}$ and $sub(c,c',i) = sub(c',c,i)^{-1}$. Moreover, it can be proven that the edit distance is indeed a distance if $\forall op \in Op, w_{op} = w_{op^{-1}}$.

The Levenstein distance is the edit distance with all costs to 1. The Needleman-Wunch distance is the edit distance with a higher costs for ins and del. Monger-Elkan distance function which has particular cost parameters for operations, scaling to the interval [0 1].

A roughly similar metric, but not based on an edit distance model, the Jaro-Winkler metric which is based on the number and order of the common characters between two strings is also compared.

Definition 12 (Jaro similarity). *The Jaro similarity is a similarity* $\sigma : \mathbb{S} \times \mathbb{S} \rightarrow [0,1]$

$$\sigma(s,t) = \frac{1}{3}.(\frac{com(s,t)}{|s|} + \frac{com(t,s)}{|t|} + \frac{com(s,t) - transp(s,t)}{2 * com(s,t)})$$

in which

$$s[i] \in com(s,t) \textit{ iff } \exists j \in [i - (min(|s|,|t|)/4\; i + (min(|s|,|t|)]$$

and $transp(s,t)$ *are the element of* $com(s,t)$ *which are at different places in* s *and* t.

Token-based distances

Token-based distances are used for comparing pieces of texts rather than labels. They starts with a segmentation of the text into "token" (generally substrings of the initial strings) which are compared as (multi-)sets of such tokens instead of strings (these multi sets are also known as vectors in which each dimension corresponds to a term and the value corresponds to the number of term occurence in the string).

There exists several measures for this tasks which are based on comparing two sets of strings and are thus relevant of set comparison (see § 3.4). One can cite the Jaccard similarity, TF/IDF, cosine similarity, Jensen-Shannon distance and Fellegi and Sunter method extended to a token distance.

TF/IDF (Term frequency/Inverse document frequency) is usually not a measure of similarity: it assesses the relevance of a term to a document (and is used here to assess the relevance of a substring to a string by comparing the frequency of appearence of the string in the document with regard to its frequency in the whole corpus.

Definition 13 (Term frequency/Inverse document frequency). *Given a corpus* C *of strings (usually documents), we define the following measures:*

$$\begin{aligned} \forall t \in \mathbb{S}, \forall s \in C, tf(t,s) &= s\#t && (term\ frequency) \\ \forall t \in \mathbb{S}, doc(t) &= |\{s \in C; t \in s\}| \\ \forall t \in \mathbb{S}, idf(t) &= log(\frac{|C|}{|doc(t)|}) && (inverse\ document\ frequency) \end{aligned}$$

Building a similarity from TF/IDF, amounts to computing them for the terms of both documents and aggregating and comparing the results. This is a special similarity measure since it is not intrinsic but dependent on a corpus (like most of the other cited methods).

Path comparison

Path difference consists in comparing not only the labels of objects but the sequence of labels of entities to which those bearing the label are related. A simple (and only) example is the one which concatenates all the names of the superclasses of a particular class before comparing it. So the result is dependent on the individual string comparison aggregated in some ways. The Comma system uses it [Do *et al.*, 2002].

Definition 14 (Path distance). *Given two sequences of strings, $\langle s_i\rangle_{i=1}^n$ and $\langle s'_j\rangle_{j=1}^m$, their path distance is obtained by:*

$$\delta(\langle s_i\rangle_{i=1}^n, \langle s'_j\rangle_{j=1}^m) = \lambda.\delta'(s_1, s'_1) + (1-\lambda).\delta(\langle s_i\rangle_{i=1}^{n-1}, \langle s'_j\rangle_{i=1}^{m-1})$$

where

$$\delta(\langle\rangle, \langle s'_j\rangle_{j=1}^m) = \delta(\langle s_i\rangle_{i=1}^n, \langle\rangle) = 0$$

with δ' some of the other string or language based distances and $\lambda \in [0\ 1]$.

This measure is very dependant on the similarity between the last element of each paths. It takes into account the prefix, but it can only influence form some amount which decreases as their distance from the end of the sequence increases.

Another way to take these paths into account is simply to apply them a distance on sequences such as described in [Valtchev, 1999].

3.2.2 Language-based methods

Language-based methods rely on using Natural Language Processing (NLP) techniques to find associations between instances of concepts or classes. These methods may be either intrinsic (using the internal linguistic properties of the instances, such as morphological and syntactic properties) or extrinsic (requiring the use of external resources, e.g. lexicon-based and multilingual methods). Language-based methods essentially rely on the **expressive** and **productive** properties of natural language, which means that even technical terms can be expressed in many different ways without intrinsically altering their meaning [Maynard and Ananiadou, 1999]. This is usually referred to as **term variation**. Some of the most notable research in this area was carried out by the FASTR project [Jacquemin and Royaute, 1994; Jacquemin, 1996] which aimed to find alternative variants of terms for automatic indexing. The goal in this case was slightly different from our concerns with ontological alignment, but the methods are equally applicable since the idea is to identify whether two terms essentially refer to the same concept. In an ontology these terms may represent either instances of classes that we wish to match.

[Maynard and Ananiadou, 1999] distinguishes 3 main kinds of term variation: morphological, syntactic and semantic, although combinations of these are also possible (in particular, morphosyntactic variation is very common). We can add to this multilingual variation, i.e. where the term variant is expressed in a different language. These types can be subdivided further. Morphological variants can be divided into inflectional and derivational variants (or a combination of the two). Jacquemin and Royauté distinguish between 3 types of syntactic variants: coordination, permutation and insertion, and also define a separate category of morphosyntactic variants: a combination of (derivational) morphological and syntactic variants. However, all three main types of variants can be combined in various ways. Table 3.1 depicts the types of variants and gives some examples of possible variants of the term *enzyme activity*.

Terminological methods allow to find the relation between these entities.

Type	Subtype	Example
Morphological	Inflection	enzyme activities
	Derivation	enzymatic activity
	Inflectional-Derivational	enzymatic activities
Syntactic	Insertion	enzyme amidolytic activity
	Permutation	activity of enzyme
	Coordination	enzyme and bactericidal activity
Morphosyntactic	Derivation-Coordination	enzymatic and bactericidal activity
	Inflection-Permutation	activity of enzymes
Semantic		fermentation
Multilingual	French	activité d'enzyme

Table 3.1: Variants of the term *enzyme activity*.

Intrinsic methods

These methods perform the terminological matching with the help of morphological and syntactic analysis to perform term normalisation. They are frequently used in Information Retrieval to improve searching. For example, they will find as similar classes `Match` and `Matching`. They operate on the principle of finding linguistic variants of the same string, as described above.

Morphological variants are most commonly identified through *stemming* algorithms, which strip words to their base form by removing suffixes such as plural forms and affixes denoting declension or conjugation. For example, `match`, `matching` and `matches` would all be reduced to the single stem `match`. The Porter stemming algorithm (or "Porter stemmer") [Porter, 1980] is often used for removing the more common morphological and inflectional endings from words in English. It has also been implemented for other languages such as German, and is freely available in many different formats and programming languages[1].

However, morphological conflation of terms is not just about suffix stripping. Morphological variations can also be expressed through different graphic transformations which have nothing to do with suffixes – or indeed any kind of affixes. This is referred to as *allomorphy*. Derived allomorphs, which are essentially a phonological mutation of the basic form, may still be recognisable using stemming techniques, e.g. "loaf" and "loaves". In the case of partial or total suppletion, however, the two allomorphs may bear only a partial resemblance to each other, or no resemblance at all (respectively), e.g. "bring – brought" or "go – went". In this case, a greedy clustering algorithm based on suffix strings may be used (for partial suppletion) [Jacquemin, 1997] or reference to external lists may be required.

Additionally, suffixes may be specific to a particular domain (e.g. medicine and biomedicine) and are therefore not considered by traditional stemmers such as [Porter, 1980] and [Lovins, 1968] or mentioned in the traditional literature on morphology, so domain-specific and/or ad-hoc methods may be required.

Jacquemin's method also deals with syntactic variants through the use of metarules, which state possible transformations that can be applied to particular classes of words. Syntactic transformations can only be applied to multi-word terms (at least, where one of the two terms in question is multi-word). For example, the rule

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

```
Metarule Perm (X1 --> X2 X3) = X1 --> X3 X4 X5 X2
```

enables the term "effect of glucose concentration" to be matched with the term "concentration effect", where X_1 can consist of either X_2 X_3 (glucose concentration) or X_3 X_4 X_5 X_2 (effect of glucose concentration).

This can be turned easily in a distance based on a cost model in which, for instance, term equality is 0, simple modifications are .5 and modifications are multiplied.

Extrinsic methods

Extrinsic methods make use of external resources such as dictionaries and lexicons. Lexicon-based methods essentially match terms which are semantically related, using an external lexicon or dictionary. For example, they consider synonyms as equivalent and hyponyms as subsumed, finding as similar classes `Match` and `Alignment`. Typically, WordNet is used – either to simply find close relationships such as synonymy between the two terms, or to compute some kind of semantic distance between them in order to decide if a relationship should hold. For example, [Patel *et al.*, 2003] use the principle of locality to discover relationships in WordNet between instances of concepts; [Su and Gulla, 2003] uses a semantic distance measure to strengthen the mappings of instances whose concept names are closely related in WordNet; [Silva and Rocha, 2003] also uses a semantic distance measure, adapted from that proposed by [Resnik, 1995], to compute similarities in order to establish correspondences between the terms, which are then used to transform instances from the source ontology into instances from the target ontology. A number of these methods have been implemented in a Perl package[2].

Simple measures can be defined here (we only consider synonyms because they are the basis of wordnet) but other relationships can be used. Moreover, the hyponym/hyperonym hierarchy is, in this respect, similar to a class hierarchy and the measure defined in § **??** can be used here. The simplest use of synonyms is in the following:

Definition 15 (Synonymy). *Given two terms s and t and a synonym resource Σ, the synonymy is a similarity $\sigma : \mathbb{S} \times \mathbb{S} \rightarrow [0, 1]$ such that:*

$$\sigma(s,t) = \begin{cases} 1 & \textit{if } \exists c \in \Sigma; s \in c \wedge t \in c \\ 0 & \textit{otherwise} \end{cases}$$

This strict exploitation of synonymy does not allow to discriminate when two objects are not synonyms, how far they are and when they are synonyms, how close they are. But, the synonymy being a relation, all the measures on the graph of relations can be used on wordnet synonyms, such as:

- compute the symmetric difference of the sets of synonyms of two terms;
- compute the size of the shortest path between two terms in the synonym graph;

A more elaborate measure is the one proposed in [Resnik, 1995]. It takes into account that the terms can be part of several “synset” and uses a measure in the “is-a” hierarchy between synsets. Each synset (c) is associated a probability of occurence ($pi(c)$) of an instance of the concept associated to a particular synset. This probability is obtained from corpus study. It is obviously

[2]http://wn-similarity.sourceforge.net/

such that the more specific the concept, the lower its probability. The similarity between two terms is function of the common synset of both terms which maximises the information content (taken as the negation of the logarithm of the probability).

Definition 16 (Resnik semantic similarity). *Given two terms s and t and a partially ordered synonym resource $\langle \Sigma, \leq \rangle$ provided with a probability π, Resnik semantic similarity is a similarity $\sigma : \mathbb{S} \times \mathbb{S} \to [0,1]$ such that:*

$$\sigma(s,t) = max_{k;\exists c,c' \in \Sigma; s \in c \wedge t \in c' \wedge c \leq k \wedge c' \leq k}(-log(\pi(k)))$$

This similarity is not normalised.

Multilingual methods

Multilingual methods involve matching between terms in different languages, in order to create a multilingual ontology from two bilingual ones, or to align two multilingual ontologies. Typically they would make use of a multilingual dictionary such as EuroWordNet, though there are other possible methods. They use ideas and techniques from machine translation, clustering and monolingual ontology alignment.

There has been much work on semantic matching within a single language, but very little on cross-lingual semantic matching, i.e. measuring the semantic similiarity of words across languages. One of the main difficulties with this is that there can be many-to-many translations of words or terms, rather than a single direct correspondence. [Ngai *et al.*, 2002] proposes a method for multilingual ontology alignment based on the approach of [Fung and Lo, 1998] which uses word co-occurrence patterns to indicate semantic similarity. This method has the advantage of being able to use non-parallel bilingual corpora.

Ngai's approach is based on the assumption that even though each sense of a term may have different translations, i.e. there may be a one-to-many correspondence for translations of its different meanings, it is unlikely that its synonyms will have the exact same set of translations for each of their meanings. The approach considers the average similarity between terms in a synset (set of synonyns for that term) from one ontology and terns from all potential candidates for alignment from the other ontology. All candidate synsets are ranked according to a similarity measure, and the highest ranked set wins.

The Polylex project [3] aimed at creating a single multilingual lexicon for Dutch, English and German from individual monolingual lexicons (contained in the CELEX database[4]). Their approach relies on the fact that the 3 languages share many aspects of syntax, morphology, morphophonology, phonology and orthography. Their method makes use of orthogonal multiple inheritance, which allows a node in the hierarchy to inherit different kinds of information from different parent nodes. With this resource, each language's nodes can inherit a mix of information from within the language's own hierarchy and from the common hierarchy.

3.3 Structural (internal and external) methods

The structure of entities that can be found in ontology can be compared, instead of comparing their names or identifiers. This comparison can be subdivided in a comparison of the internal structure

[3]http://www.informatics.susx.ac.uk/research/nlp/polylex/polylex.html

[4]http://www.kun.nl/celex/index.html

of an entity (i.e., its attributes or, for speaking OWL, the properties which takes their values in a data type) or the comparison of the entity with other entities to which it is related.

3.3.1 Internal structure

Methods based on the internal structure of entities use criteria such as the range of their properties (attributes and relations), their cardinality, and the transitivity and/or symmetry of their properties to calculate the similarity between them. Internal structure based methods are sometimes referred to as constraint based approaches in literature, e.g., [Rahm and Bernstein, 2001b].

Entities with comparable internal structure or properties with similar domain and range in two ontologies can be numerous, that is why these kinds of methods are commonly used to create alignment clusters rather than to discover accurate correspondences between entities. They usually appear combined with other local methods like terminological, structural, or extensional ones and are in charge of reducing the number of align candidates. As stated by Li and Clifton in [Li and Clifton, 2000], methods comparing field specifications at the schema level do not intend to completely replace searching through a synonym lexicon, but help to determine attribute correspondences when no conclusion can be made simply by searching a synonym dictionary. They can be used with other approaches, as a 'first step' to eliminate most of the attributes that are clearly incompatible.

Compatibility evaluation

The Cupid algorithm for discovering mappings between schema elements [Madhavan *et al.*, 2001b] depends, among others things, on the compatibility between datatypes of attributes which is assessed thanks to a lookup table. Identical data types have the highest compatibility value. Compatible type a compatibility value which does not disqualify them.

Data-based domain comparison

Data-based domain comparison is first an extensional method for aligning. It can be used for inductively finding domain information which is often missing in database schemas in which type structures are relatively poor.

SEMantic INTegrator (SEMINT) is a tool based on neural networks described in [Li and Clifton, 2000] to assist in identifying attribute correspondences in heterogeneous databases. SEMINT supports access to a variety of database systems and utilizes both schema information and data contents to produce rules for matching corresponding attributes automatically. The schema information used by SEMINT includes data types, length, scale, precision, and the existence of keys, value, and range constraints, disallowing null values, etc.

The instance data is used to compute some statistics like maximum, minimum, mean, variance, coefficient of variance, existence of null values, existence of decimals, scale, precision, grouping, and number of segments.

Other approaches to determine attribute correspondences using instance data try to compare attribute values. Larson et al. [Larson *et al.*, 1989] and Sheth et al. [Sheth *et al.*, 1988] discussed how relationships and entity sets can be integrated primarily based on their domain relationships: EQUAL, CONTAINS, OVERLAP, CONTAINED-IN, and DISJOINT. The problem is that determining such relationships can be time consuming and tedious. Another limitation is the ability to handle faults: small amounts of incorrect data may lead the system to draw a wrong conclusion on domain relationships. Other approaches like [Li and Clifton, 1994a] proposes methods that utilize data patterns and distributions instead of data values and domains. The result is a better fault tolerance and less time-consumption since only a small portion of data values are needed by employing data sampling techniques. In general, applying internal structure methods to instances allow a more precise characterization of the actual contents of schema elements and thus, more accurately determine corresponding data types based, for example, on the discovered value ranges and character patterns.

Relative volume

Comparing the internal structure of objects amounts to compare their properties and to compose the obtained comparison index. The composition operation is considered in section 4.1. It can be used for composing the values of internal properties alone or to aggregate with the result of other similarities such as those resulting from the external structure.

Depending on the entities to be considered, the property values can be different: values in classes are domains while values in individuals are values. Moreover, these values can be structured in sets or sequences. It is thus important to consider this in the comparison.

[Valtchev, 1999] proposes a framework in which the types or domains of properties must be compared on the basis of their interpretations: sets of values. Type comparison is based on their respective size, in which the size of a type is the cardinal of the set of values it defines. The distance between two domains is then given by the difference between their size and that of their common generalization. This measure is usually normalized by the size largest possible distance attached to a particular datatype.

Definition 17 (Relative size distance). *Given two domain expressions e and e' over a datatype τ, the relative size distance $\delta : 2^\tau \times 2^\tau \longrightarrow [0, 1]$, is such that:*

$$\delta(e, e') = \frac{|gen_\tau(e \vee e')| - 1/2 * (|gen_\tau(e)| + |gen_\tau(e')|)}{|\tau|}$$

in which $gen(.)$ provides the generalization of a type expression.

There are three advantages to this measure: the most obvious one is that it is normalized. The second one is that it is totally general. The third one is that can easily be mapped to the usual measures that are often used.

Usually, the common generalization depends on the type: it is a set for enumerated types, an interval for ordered types (it can also be a set of intervals). In case of dense types, the size of a domain is the usual measure of its size (Euclidean distance). The case of infinite types has to be taken adequately (by evaluating the largest possible domain in a computer or by normalizing with regard to the actual corpus). Normalizing over the actual largest distance in the corpus, if possible, is often a good idea. Indeed, it is not fair to normalise the age of people with that of planets or their size even if they use the same unit.

Another advantage of this framework is that is encompasses value comparisons which are compared as singletons.

Comparing multiplicities and properties

Another approach that compares attribute specifications using design information (the schema information) has been proposed in [Navathe and Buneman, 1986], the characteristics of attributes discussed are uniqueness, cardinality, domain, static semantic integrity constraints, dynamic semantic integrity constraints, security constraints, allowable operations, and scale.

Evaluating the compatibility of types on multiplicities is relatively easy: multiplicities are first interpreted as reducing the integer interval $[0\ \ +\infty[$ and two multiplicities are compatible if the intersection of the corresponding intervals is non empty. Evaluating the similarity between the multiplicites of two properties can be achieved by the other methods presented here considering that they are interpreted as sets of integers.

In [Ehrig and Sure, 2004], Ehrig and Sure proposed the definition of a set of rules for determining similarity between ontology entities and point the fact that some features from OWL related to internal structure could be used, but are discarded by now, as they do not have any wide distribution yet. These features are property characteristics as symmetry and restrictions of values, among others.

Similarity between collections

It is often necessary to compare sets or lists of objects (e.g., the set of children of someone or the sequence of meals in a menu). In this case, general techniques can be used for assessing the similarity or distance between these sets depending on the similarity applying to the type of their elements. Concerning sets, these methods will be presented in section 3.4 in the context of extension comparison. Concerning sequences, they can be adapted from some of the measures that have been presented in section 3.2.1 which have considered strings as sequences of characters and paths as sequences of strings. In addition, some of the methods of § 3.4 can also be applied to sequences.

These observations apply both to internal and external structure.

3.3.2 External structure

The similarity comparison between two entities from two ontologies can be based on the position of entities within their hierarchies. If two entities from two ontologies are similar, their neighbours might also be somehow similar. This remark can be used in several different ways. Criteria for deciding that the two entities are similar include:

C1 Their direct super-entities (or all of their super-entities) are already similar [Dieng and Hug, 1998a].

C2 Their sibling-entities (or all of their sibling-entities) are already similar.

C3 Their direct sub-entities (or all of their sub-entities) are already similar [Dieng and Hug, 1998a].

C4 All (or most) of their descendant-entities (entities in the subtree rooted at the entity in question) are already similar.

C5 All (or most) of their leaf-entities (entities, which have no sub-entity, in the subtree rooted at the entity in question) are already similar [Madhavan *et al.*, 2001a].

C6 All (or most) of entities in the paths from the root to the entities in question are already similar [Bach *et al.*, 2004].

Of course, an approach can combine several of the above criteria [Mädche and Staab, 2002; Bach *et al.*, 2004].

External structure comparison faces problems when the viewpoint of two ontologies is highly different (see Deliverable D2.1.1). For example, with the same class "Human", in the first ontology, it can be specialized into two sub-classes "Man" and "Woman" but in the second ontology, it can be divided into "Adult" and "Young_Person". In this case, the application of this method is not a good solution.

The methods for aggregating the external structure features of entities are very similar to those to be used in case of internal structure. However, one can find some particularities when dealing with partially ordered domains and, in particular, with hierarchies. Methods specifically related to hierarchical domains are considered here.

Mereologic structure

In a mereologic hierarchy, the relations between entities are whole-part relations. The sub-entity is a "part" of the super-entity, and vice versa, the super-entity can be composed of some different sub-entities. For example, a super-class "Desktop_Computer" can have some whole-part relations with a sub-class "Motherboard", with a sub-class "Graphics_Card", with a sub-class "CPU"... The application of criterion [C5] for computing the similarity between entities from different ontologies with mereologic structure does not seem to be convenient here. The other criteria may still be applied.

Taxonomic structure

In a taxonomic hierarchy, the relations between entities are specialisation relations. The sub-entity is a "specialisation" of the super-entity, and vice versa, the super-entity is a "generalisation" of the sub-entity. A super-entity can have relations with one or more sub-entities and similarly, a sub-entity can have relations with one or more super-entities. For example, a super-class "Motor" can have some specialisation classes such as "Motocycle", "Passenger Vehicle". As an other example, class "Researcher" is a generalisation class of two sub-classes "Research Fellow" and "Senior Researcher".

Contrary to mereologic structure, the similarity computation between entities from different ontologies with taxonomic structure can apply criterion [C5]. The application of above criteria [C1-6] can tell us that the class "Researcher" may be similar to the class "Research Staff Member" if the latter has also two classes "Research Fellow" and "Senior Researcher" as its specialisation.

There have been several measures proposed for comparing classes based on the taxonomic structure. The *structural topological dissimilarity* δ^s on a domain [Valtchev and Euzenat, 1997] follows the graph distance, i.e. the shortest path distance in a graph (taken here as the transitive reduction of the hierarchy).

Definition 18 (structural topological dissimilarity on hierarchies). *The structural topological dissimilarity $\delta : O \times O \to \mathbb{R}$ is a dissimilarity over a hierarchy $H = \langle O, \leq \rangle$, such that:*

$$\forall e, e' \in O, \delta(e, e') = \min_{c \in O}[\delta(e, c) + \delta(e', c)] \tag{3.1}$$

where $\delta(e, c)$ is the number of intermediate edges between an element e and another element c.

This corresponds to the unit tree distance of [Barthélemy and Guénoche, 1992] (i.e., with weight 1 on each edge). The corresponding normalized function is:

$$\overline{\delta}(e, e') = \frac{\delta(e, e')}{\max_{o, o' \in O} \delta(o, o')} \tag{3.2}$$

The result given by such a measure is not always semantically relevant since a long path in a class hierarchy can often be summarized as a short one.

The upward cotopy distance had been described in [Mädche and Zacharias, 2002] as follows.

Definition 19 (upward cotopic distance). *The upward cotopic distance $\delta : O \times O \to \mathbb{R}$ is a dissimilarity over a hierarchy $H = \langle O, \leq \rangle$, such that:*

$$\delta(c, c') = \frac{|UC(c,H) \cap UC(c',H)|}{|UC(c,H) \cup UC(c',H)|} \tag{3.3}$$

where $UC(c,H) = \{c' \in H; c \leq c'\}$ is the set of superclasses of c.

Of course, these measures cannot be applied as such in the context of ontology alignment since the ontologies are not supposed to share the same taxonomy H (but this can be used in conjunction with a common resource such as wordnet). For that purpose, it is necessary to develop these kinds of measure over a pair of ontologies. In [Valtchev, 1999; Euzenat and Valtchev, 2004], this amounts to use a (local) matching between the elements to be compared (for instance, the hierarchies).

Relations

The similarity computation between entities can be also based on their relations. If class A relates to class B by relation R in one ontology, and if class A' relates to class B' by relation R' in the other ontology, and if we know that B and B' are similar, R and R' are similar, we can infer that A and A' may be similar too. By the same way, if A is similar to A', R is similar to R', B may be similar with B'; or R may be similar with R' if we know before that A and A' are similar, B and B' are similar: the similarity among relations in [Mädche and Staab, 2002] is computed according to this principle. For example, classes “Company” and “University” will be considered similar because they have a similar relation “hasEmployee” with class “Employee” and class “Professor” which are themselves similar.

This can be extended to a set of classes and a set of relations. It means that if we have a set of relations $R_1 \dots R_n$ in the first ontology which are similar with an other set of relations $R'_1 \dots R'_n$ in the second ontology, it is possible that two classes, which are the domains of relations in those two sets, are similar too.

One of the problems for this approach is to define how two relations are similar. This approach is based on the similarity of relations to infer the similarity of their domain classes or their range classes. Relations between classes in an ontology can be considered as entities in that ontology, they can be organized in a relation hierarchy, and like classes, the similarity computation between relations is also a big problem.

Remark: Both above problems in which we compare the similarity of entities in mereologic hierarchy or taxonomic hierarchy can be considered as a sub-case of the last problem where the relations between entities (classes) are only whole-part relations or is_a/specialisation relations.

3.4 Extensional (based on instances)

Extension-based methods compares the extension of classes, i.e., their set of instances rather than their interpretation. There are two very different conditions in which such techniques can be used: when the classes share the same instances (§ 3.4.1) and when they do not (§ 3.4.2).

3.4.1 Common extension comparison

The easiest way to compare classes A and B when they share classes is to test their intersection and to consider that these classes are very similar when $A \cap B = A = B$, more general when $A \cap B = B$ or $A \cap B = A$. However, the dissimilarity can only be 1 when none of these cases apply, for instance if the classes have some instance in common but not all. A way to refine this is to use of the symmetric difference between the two extension.

Definition 20 (Symmetric difference). *The symmetric difference (sometimes also called Hamming distance) between two sets is a disimilarity function* $\delta : 2^E \times 2^E \rightarrow \mathbb{R}$ *such that* $\forall x, y \subseteq E$, $\delta(x,y) = \frac{|x \cup y - x \cap y|}{|x \cup y|}$.

This version of the symmetric difference is normalized.

It is also possible to compute a similarity based on the probabilistic interpretation of the set of instances. This is the case of the Jaccard similarity.

Definition 21 (Jaccard similarity). *Given two sets A and B, let $P(X)$ the probability of a random instance to be in set X, the Jaccard similarity is defined by:*

$$\sigma(A,B) = \frac{P(A \cap B)}{P(A \cup B)}$$

This measure is normalized and reaches 0 when $A \cap B = \emptyset$ and 1 when $A = B$.

3.4.2 Similarity-based extension comparison

Similarity-based techniques do not ask the classes to share the same set of instances (however, they can still be applied in that case). In particular, the above methods always return 0 when the two classes do not share any instances, disregarding the distance between the elements of the sets. In some case, it is preferable to assess the distance between these classes. In order to compare the set of instances they use a (dis)similarity between the instances which can be computed with any of the methods presented here.

In data analysis, the linkage aggregation methods allows to assess the distance between two sets whose objects are only similar.

Definition 22 (Single linkage). *The single linkage measure between two sets is a disimilarity function* $\Delta : 2^E \times 2^E \rightarrow \mathbb{R}$ *such that* $\forall x, y \subseteq E$, $\Delta(x,y) = \min_{(e,e') \in x \times y} \delta(e,e')$.

Definition 23 (Full linkage). *The complete linkage measure between two sets is a disimilarity function* $\Delta : 2^E \times 2^E \rightarrow \mathbb{R}$ *such that* $\forall x, y \subseteq E$, $\Delta(x,y) = \max_{(e,e') \in x \times y} \delta(e,e')$.

Definition 24 (Average linkage). *The average linkage measure between two sets is a disimilarity function* $\Delta : 2^E \times 2^E \rightarrow \mathbb{R}$ *such that* $\forall x, y \subseteq E$, $\Delta(x,y) = \frac{\sum_{(e,e') \in x \times y} \delta(e,e')}{|x| * |y|}$.

Each of these methods have its own benefits. Another methods from the same familly is the Hausdorff distance measuring the maximal distance of a set to the nearest point in the other set:

Definition 25 (Haussdorf distance). *The Haussdorf distance between two sets is a disimilarity function* $\Delta : 2^E \times 2^E \rightarrow \mathbb{R}$ *such that* $\forall x, y \subseteq E$,

$$\Delta(x,y) = max(\max_{e \in x} \min_{e' \in y} \delta(e,e'), \max_{e' \in y} \min_{e \in x} \delta(e,e'))$$

3.4.3 Matching-based comparison

The problem with the former distances, but average, is that their value is function of the distance between one couple of members of the set. The average linkage on the opposite has its value function of the distance between all the possible comparison.

Matching-based comparisons [Valtchev, 1999] consider that the element to be compared are those which are corresponds to each others, i.e., the most similar one.

To that extent, the distance between two sets is considered as a value to be minimized and its computation as an optimization problem: the one of finding the elements of both sets which corresponds to each others. This corresponds to solving the square assignment problem.

Definition 26 (Match-based similarity). *The match-based similarity between two sets is a similarity function $MSim : 2^E \times 2^E \rightarrow \mathbb{R}$ such that $\forall x, y \subseteq E$,*

$$MSim(x,y) = \frac{\sum_{\langle n,n' \rangle \in Pairing(x,y)} \sigma(n,n')}{max(|x|,|y|)}$$

in which $Pairing(x,y)$ is a mapping of elements of x to elements of y which maximises the group $MSim$ similarity.

This match-based similarity already require an alignment of entities to be computed. It also depends on the kind of mapping that is required. Indeed, the result will be different if the mapping is required to be injective or not.

The match-based comparison can also be used when comparing sequences. See [Valtchev, 1999] for a complete discussion on that topic.

3.5 Semantic methods (based on models)

The key characteristics of semantic methods is that they have model-theoretic semantics which is used to justify their results. Hence they are deductive methods. Examples are propositional satisfiability (SAT) and modal SAT techniques or description logic based techniques.

As from [Giunchiglia and Shvaiko, 2004; Giunchiglia *et al.*, 2004; Bouquet *et al.*, 2003] the approach of applying propositional satisfiability (SAT) techniques to alignment is to translate the matching problem, namely the two tree-like structures (e.g., concept hierarchies) and mapping queries into a propositional formula and then to check it for its validity. By mapping query we mean here the pair of nodes and a possible relation between them. Notice that SAT deciders are correct and complete decision procedures for propositional satisfiability, and therefore will exhaustively check for all possible mappings.

Modal SAT can be used, as proposed in [Shvaiko, 2004], for extending the methods related to propositional SAT to binary predicates. Its basis is to delimit propositional SAT from the case of trees which allows handling only unary predicates (e.g., classes) by admitting binary predicates (e.g., slots, etc.). The key idea is to enhance propositional logics with modal logic (or a kind of description logics) operators. Therefore, the matching problem is translated into a modal logic formula which is further checked for its validity using sound and complete satisfiability search procedures.

Description logics techniques, i.e. subsumption test, can be used to establish the relations between classes in a purely semantic manner. In fact, first merging two ontologies (after renaming) and then testing each pair of concepts and role for subsumption is enough for aligning terms with the same interpretation (or with a subset of the interpretations of the others).

Of course, pure semantic methods do not perform very well alone, they often need a preprocessing phase providing “anchors”, i.e., entities which are declared to be equivalent (based on their name or human input for instance).

These methods being semantically exact do only provide a similarity of 1 for objects considered equivalent. However, they allow for more variety in the expression of the correspondence between entities such as establishing that one entity satisfies all the models of another or that two entities cannot share any instance.

Chapter 4

Global methods

Once the local methods for determining the similarity or (dis)similarity are available, there remain to compute the alignment. This involve some kind of more global treatments, including:

- aggregating the results of these base methods in order to compute the similarity between compound entities (§ 4.1);
- developing a strategy for computing these similarities in spite of cycles and non linearity in the constraints governing similarities (§ 4.2);
- organising the combination of various similarity/alignment algorithms (§ 4.4).
- involving the user in the loop (§ 4.5);
- finally extracting the alignments from the resulting (dis)similarity: indeed, different alignments with different characteristics can be extracted from the same (dis)similarity (§ 4.6).

Moreover, we will consider more global techniques of learning how to align ontologies from example (§ 4.3).

All these steps are considered here under the name of global methods. They combine local methods in order to define an original algorithm for providing an alignment.

4.1 Compound similarity

Compound similarity is concerned with the aggregation of local (and compound) similarities. As a matter of fact, some objects are understood as compound and their (dis)similarity depends on that holding between their components (the similarity between two classes may depend on the similarity of their names, their super-classes and their properties).

4.1.1 Classical distances and weighted sums

In case the difference between some properties must be aggregated, one of the more common falilly of distances are the Minkowski distances

Definition 27 (Minkowski distance). *Let O a set of objects which can be analized in n dimensions, the Minkowski distance between two such objects is:*

$$\forall x, x' \in O, \delta(x, x') = (\sum_{i=1}^{n} \delta(x_i, x'_i)^p)^(1/p)$$

in which $\delta(x_i, x'_i)$ is the dissimilarity of the pair of objects along the i^{th} dimension.

Instances of the Minkowski distances are the Euclidean distance (when $p = 2$), the City-block (a.k.a. Manhattan) distance (when $p = 1$) and the Chebichev distance (when $p = +\infty$).

These distances can be weighted in order to give more importance to some parameters. They can be normalized by dividing their results by the maximum possible distance (which is not always possible).

It has the main drawback of not being linear if $p \neq 1$, see [Valtchev, 1999] for a discussion of the consequences.

The simple linear aggregation can be further refined by adding weights to this sum. Weighted linear aggregation does consider that some of the values to be aggregated do not have the same importance (for instance, similarity in properties is more important than similarity in comments). The aggregation function will thus use a set of weights $w_1, \ldots w_n$ corresponding to a category of entities or properties. The aggregation function is then:

Definition 28 (Weighted sum). *Let O a set of objects which can be analized in n dimensions, the weighted sum (or weighted average) between two such objects is:*

$$\forall x, x' \in O, \delta(x, x') = (\sum_{i=1}^{n} w_i * \delta(x_i, x'_i)$$

in which $\delta(x_i, x'_i)$ is the dissimilarity of the pair of objects along the i^{th} dimension and w_i is the weight of dimension i.

In fact, the weights can be different depending on the categories of the object aggregated as well as that of the similarity computed [Euzenat and Valtchev, 2004]. Then, the function can use a set of weights w_C^P depending of the category of object C and the kind of value computed P.

This kind of measure can be normalized, if all values are normalized, by having:

$$\sum_{i=1}^{n} w_i = 1$$

.

4.1.2 Triangular norms

Triangular norms are used as conjunction operators in uncertain calculi.

Definition 29 (Triangular norm). *A triangular norm T is a function from $D \times D \rightarrow D$ (with D a set ordered by $\leq$ and provided with an upper bound $\top$) satifying:*

$$\begin{aligned} T(x, \top) &= x & (boundary\ condition) \\ x \leq y \implies T(x, z) &\leq T(y, z) & (monotonicity) \\ T(x, y) &= T(y, x) & (commutativity) \\ T(x, T(y, z)) &= T(T(x, y), z) & (associativity) \end{aligned}$$

There are typical examples of the triangular norms: $min(x, y)$, $x.y$ and $max(x + y - 1, 0)$. All are normalized if the measures provided to them are normalized; min is the only idempotent norm ($\forall x, min(x,x) = x$) and the values are ordered by $min(x,y) \geq x.y \geq max(x+y-1,0)$. Moreover, any triangular norm can be expressed as a combination of the these three functions [Hajek, 1998].

The triangular norms can be extended to n-ary measures. On instance of such a generalization using both weights and n arguments is the weighted product.

Definition 30 (Weighted product). *Let O a set of objects which can be analized in n dimensions, the weighted product between two such objects is:*

$$\forall x, x' \in O, \delta(x,x') = \prod_{i=1}^{n} \delta(x_i, x'_i)^{\lambda_i}$$

in which $\delta(x_i, x'_i)$ is the dissimilarity of the pair of objects along the i^{th} dimension and λ_i is the weight of dimension i.

These operators have the drawback that if only one of the dimension has a measure of 0, then the result is 0.

4.1.3 Weighted averages (and fuzzy aggregates)

The weighted average is very often used as a fuzzy aggregate [Gal *et al.*, 2004]. We will see why below.

Definition 31 (Fuzzy aggregate operator). *A fuzzy aggregate operator f is a function from $D^n \rightarrow D$ (with D a set ordered by $\leq$ and provided with an upper bound $\top$) satifying:*

$$\begin{array}{rr} f(x, \dots x) = x & (idempotency) \\ \forall x_i, y_i \text{ such that } x_i \leq y_i, f(x_1, \dots x_n) \leq f(y_1, \dots y_n) & (increasingmonotonicity) \\ f \text{ is a continuous function} & (continuity) \end{array}$$

A typical example of a fuzzy aggregate operator is the weighted average.

Definition 32 (Weighted average). *Given a set $w_1, \dots w_n$ of weigths,*

$$\frac{\sum_{i=1}^{n} w_i x_i}{\sum_{i=1}^{n} w_i}$$

is a weighted average function.

The simple average function is such a function with all weigths equals. Again, if the values are normalized, the weighted average is normalized.

These kinds of function are very useful if ones want to use a learning algorithm for learning the weights of the measure.

The use of neural networks (NNs) in order to realize the key concepts of a fuzzy logic system enriches the system with the ability of learning and improves the subsymbolic to symbolic mapping. Neural network realization of basic operations of fuzzy logic, such as fuzzy complement, fuzzy intersection and fuzzy union, has been proposed ([Pao, 1989])[Hsu et al., 1992]. The activation function of neurons is then set to be one of the three basic operations mentioned above in order to provide fuzzy logic inference. Another approach is to use the ordered weighted averaging neuron [Yager, 1992] for representing fuzzy aggregating operations. A feedforward network can also be used to represent the membership function of a fuzzy set.

Fuzzy logic inference systems can be used to represent complex relations of subsymbolic to symbolic mapping by defining possibility distributions on the antecedents and the consequents of if-then rules. The nonadaptive and heuristic behavior of fuzzy logic systems can be improved with the aid of NNs. For this aim, a connectionist approach of fuzzy inference has been proposed in [Keller and Hunt, 1992]. The network is referred to as *fuzzy inference network* and implements a rule of the form.

$$x_1 \in A_1 \wedge x_2 \in A_2 \wedge \ldots A_n \in a_n \implies y \in B$$

For choosing the parameters of the fuzzy inference that are associated with the parameters of the network, a training algorithm has been proposed.

4.2 Global similarity computation

The computation of compound similarity is still local because it only provides similarity considering the neighbourhood of a node. However, similarity may involve the ontologies as a whole and the final similarity values may ultimately depend on all the ontologies. Moreover, the distance defined by local methods can be defined in a circular way (for instance if the distance between two classes depends on the distances between their instances which themselves depends on the distance between their classes or if there are circles in the ontology). In case of circular dependencies, similarity computation is not anymore possible in a local fashion. It is more like an optimization problem.

For that purposes, strategies must be defined in order to compute this global similarity. The first one is defined as a process of propagating the similarity within a graph while the second one translate the similarity definitions in a set of equations which is solved by classical techniques.

4.2.1 Similarity flooding

Similarity flooding [Melnik *et al.*, 2002] is a generic graph matching algorithm which uses fix-point computation to determine corresponding nodes in the graphs.

The two ontologies are first translated into directed labelled graphs grounding on the OIM specification [MDC, 1999]. The principle of the algorithm is that the similarity between two nodes must depend on the similarity between their adjacent nodes (whatever are the relations that must be taken into account). To implement this, the algorithm creates another graph G whose nodes are pairs of nodes of the initial graphs and there is an edge between (o_1, o'_1) and (o_2, o'_2) labeled by p whenever there are edges (o_1, p, o_2) in the first graph and (o'_1, p, o'_2) in the second one. So, the alignment does only take into account edges with the same label.

Then, the algorithm computes initial similarity values between nodes (based on their labels for instance) and then iterates steps of re-computing the similarities between nodes in function of the similarity between their adjacent nodes at the previous step. It stops when no similarity changes more than a particular threshold ϵ or after a predetermined number of steps.

The chosen aggregation function is a weighted linear aggregation in which the weight of an edge is the inverse of the number of other edges with the same label reaching the same couple of entities.

$$\sigma^{i+1}(x,x') = \frac{\sum_{((x,x'),p,(y,y'))\in G}\sigma^i(y,y')}{|\{(y,y')|((x,x'),p,(y,y'))\in G\}|} + \frac{\sum_{((y,y'),p,(x,x'))\in G}\sigma^i(y,y')}{|\{(y,y')|((y,y'),p,(x,x'))\in G\}|}$$

The values are further normalised with regard to the maximal similarity value obtained (i.e., that value is assigned the value 1. and the other values are reduced proportionally).

4.2.2 Similarity equation fixpoint

In many situations, e.g., with symmetric or inverse properties, it is impossible to establish an ordering of entities in order to compute the similarities in a step-wise manner. [Euzenat and Valtchev, 2004] provides a method for dealing with circularities and dependencies between similarity definition which is described hereafter. In this case, the similarity values can only be expressed as a set of equations where each variable corresponds to the similarity between a pair of nodes. There is an equal number of equations, each of them associated to a variable. The structure of each equation follows the definition of the respective similarity function for the underlying node category.

Some facts are worth mentioning. First, there is no need for a different expression of the similarity functions in case there are no effective circular dependences between similarity values. In fact, the computation mechanism presented below establishes the correct similarity values even if there is an appropriate ordering of the variables (the ordering is implicitly followed by the step-wise mechanism). Moreover, in case some similarity values (or some similarity or (dis)similarity assertions) are available beforehand, the corresponding equation can be replaced by the assertion/value.

If each of the similarity expression is a linear aggregation of other similarity variables, this system would be solvable directly because all variables are of degree one.

However, in the case of OWL-Lite, and of many other languages, the system is not linear since there could be many candidate pairs for the best match. The similarity may depend on matching the multiple edges with the similar labels outgoing from the nodes under consideration. In this approach, the similarity is computed by an $MSim$ function that first finds an alignment between the set of considered entities and then computes the aggregated similarity in function of this matching.

Given two classes c and c', the resulting class similarity function reads as follows:

$$\begin{aligned}\sigma_C(c,c') &= \pi_L^C\sigma_L(\lambda(c),\lambda(c')) \\ &+ \pi_O^C MSim_O(\mathcal{I}(c),\mathcal{I}'(c')) \\ &+ \pi_S^C MSim_C(\mathcal{S}(c),\mathcal{S}'(c')) \\ &+ \pi_P^C MSim_P(\mathcal{A}(c),\mathcal{A}'(c'))\end{aligned}$$

The function is normalised since weights are, i.e., $\pi_L^C + \pi_S^C + \pi_O^C + \pi_P^C = 1$, whereas each factor that ranges over collections of nodes/feature values is averaged by the size of the larger collection.

Nevertheless, the resolution of the resulting system can still be carried out as an iterative process that simulates the computation of the greatest fixed point of a vector function, as shown by Bisson [Bisson, 1992]. The trick consists in defining an approximation of the $MSim$-measures, solving the system, replacing the approximations by the newly computed solutions and iterating.

The first values for these $MSim$-measures are the maximum similarity found for a pair, without considering the dependent part of the equations. The subsequent values are those of the complete similarity formula filled by the solutions of the system. Note that the local matching may change from one step to another depending of the current similarity values.

However, the system is converging because the similarities can only grow (because the non dependant part of the equation will remain and all dependencies are positive) and, in case of similarity values are bounded (e.g., to 1) the similarity is bounded. The iterations will stop when no gain above a particular ϵ value is provided by the last iteration. If the algorithm converges, we cannot guarantee that it does not stop in a local optimum (that is finding another matching in the $MSim$-measures would not increase the similarity values). This could be improved by randomly changing these matchings when the algorithm stops.

This methods has some similarity with the previous one: both methods work iteratively on a set of equations extracted from a graphical form of the ontologies. Both methods ultimately depends on the computed proximities between non-described language elements, i.e., data type names, values, URIRefs, property type names, etc. Indeed, these proximities are propagated throughout the graph structure by the similarity dependancies.

However, Similarity flooding is dependent on the edge labels, while the latter method takes similarity between properties into account. Nonetheless it also considers local mappings between alternative matching edges instead of averaging over all the potential match. Moreover, the Similarity flooding method is stated so generally that its convergence is not proved.

4.3 Learning methods

Like in many other fields, learning methods developed in machine learning reveals useful in ontology alignment. In order to achieve a global similarity that can be used for aligning, techniques created for machine learning can be used instead of the one presented above. They are mainly used in two particular areas:

- supervised learning in which the ontology alignment algorithm learns how to work through the presentation of many good alignment (positive examples) and bad alignments (negative examples). Of course, the alignment can be provided by the algorithm itself. This approach is not really used so far, the reason being that it is difficult to know which techniques works well for which ontology features, so an ontology alignment algorithm learnt with several ontology pairs, might not necessarily work well for a new ontology pair.
- learning from data in which a population of instances is communicated to the algorithm together with theirs relations and the classes they belong to. From this data, the algorithm can learn the relations between classes (specialisation) and the alignment of properties.

Both techniques usually take advantage of well-known methods in machine learning: formal-concept analysis [Stumme and Mädche, 2001], Bayes learning [Berlin and Motro, 2002] or neural networks [Li and Clifton, 1994b].

4.3.1 Learning from probabilistic distribution

The Glue system [Doan *et al.*, 2004; Doan, 2002] is based on learning classifiers for classes from instances in order to evaluate the joint probability distributions of instances.

The principle is that two classes are more likely to be the same if their instances are the same. So it goal is to evaluate the probability for a random instance to be a member of both classes ($P(c, c')$). More precisely, for each couple of classes c and c', the system requires the computation of $P(c, c')$, $P(c, \overline{c'})$, $P(\overline{c}, c')$ and $P(\overline{c}, \overline{c'})$ (in which $\overline{c}$ is the complement of c, i.e., reads "not in c"). This is useful because some similarities, e.g., the Jaccard similarity (see § 3.4) can be rewritten as:

$$\frac{P(c, c')}{P(c, c') + P(c, \overline{c'}) + P(\overline{c}, c')}$$

Joint probability distributions could be estimated if both ontologies shared the same set of instances. However, when this is not the case, the algorithm learns a classifier for each class in each ontology and uses the result for attributing classes to the instances of the other ontology. It can then estimate the joint probability distributions.

In Glue, there are several learners, which are trained by data instances of ontologies. They use different criteria to evaluate the reason to belong to a class. After learning phase, different characteristic instance patterns and matching rules for single elements of the target schema are discovered. The predictions of individual matchers are combined by a meta-learner, and from that, assignment of classes to instances will be deduced.

From the probability distribution the algorithm can use a probabilistic metrics for assessing similarities between classes.

4.3.2 Learning (fuzzy) aggregation through neural networks

Another approach is the generalization of the Sugeno-Takagi inference model [1988]. *Neural-network driven fuzzy reasoning*, proposed by Takagi and Hayashi [Hayashi *et al.*, 1992], constructs the antecedent part of a fuzzy system using a back-propagation network. A lot of interesting ideas, useful in symbolic to subsymbolic mapping, can be found in this approach and especially in the steps of selection of input-output variables and clustering of the training data.

The issue of identifying the fuzzy rules and tuning the membership functions of fuzzy sets using neural networks and training algorithms has been widely studied. Horikawa et al. [Horikawa *et al.*, 1992] proposed *fuzzy modeling networks* in order to realize the fuzzy reasoning process through association of its parameters with the weights of a backpropagation learning neural network. According to this method, the basic parameters of the fuzzy model can be automatically identified by modifying the connection weights of the network. Another approach to this problem related to fuzzy control has been proposed by Lin and Lee [Lin and Lu, 1995] (chapter 19), who introduced the *fuzzy adaptive learning control network* (FALCON) to study hybrid structure-parameter learning strategies. Structure learning algorithms are used to find appropriate fuzzy rules and parameter learning algorithms are used to fine tune the membership functions and other parameters of the fuzzy inference system. Actually, Lin and Lee proposed a number of different architectures and learning procedures. The first model, the FALCON-H, is a multi-layer feedforward network which represents in a connectionist structure the basic elements and functions of a fuzzy logic controller. It is supported by a hybrid learning algorithm that consists of two separate stages, combining unsupervised learning and supervised gradient-descent learning (backpropagation). Structure learning, which can lead to the extraction of rules, is based on the adaptive fuzzy partitioning of the input and output spaces and the combination (association) of these partitions. In order to provide a more flexible fuzzy partitioning, Lin and Lee proposed the FALCON-ART model, applying adaptive resonance theory (ART) learning. The above models require precise training data to indicate the desired output through a supervised learning process. Unfortunately, such type of data may be very difficult or even impossible to obtain (especially for the feature-to-symbol-mapping). The FALCON-R model, developed to remedy the above problem, is a reinforcement learning neurofuzzy system that performs automatic construction based on a right-wrong (reward-penalty) binary signal. The applied learning algorithm of FALCON-R is complicated, since it is designed to use deterministic reinforcement feedback, stochastic reinforcement feedback and reinforcement feedback with long time delay. Berenji and Khedkar [Berenji and Khedkar, 1992] have proposed another neurofuzzy control system using reinforcement learning, the *generalized approximate reasoning-based intelligent controller* (GARIC). All the above models have been well tested for fuzzy control applications. Their advantage is that they have the ability to support automatically the extraction of fuzzy inference rules. Multi-layer perceptrons have also been used for multistage fuzzy inference based on the propagation of linguistic truth values [Uehara and Fujise, 1992].

4.3.3 Semantic Gossiping

Semantic gossiping [Aberer *et al.*, 2003b] is an approach to establish global semantic agreements in any emergent way and is based on (1) local mappings among the schemas of the participating parties and their propagation and (2) analysis of the quality of these mappings. To simplify presentation we will assume a peer-to-peer (P2P) system in the following.

We assume that groups of peers have already agreed on common semantics, i.e., a common schema. We denote these groups as *semantic neighborhoods*. If two peers located in two disjoint neighborhoods meet, e.g., during query processing and forwarding, they can exchange their schemas and provide translations between them. We assume that the translations are provided by the users or through any feasible approach for alignment. During the life-time of the system, each peer has the possibility to learn about existing translations and add new ones. This means that incrementally a directed graph of translations will be built among the peer schemas along with the normal operation of the system.

This translation graph has two interesting properties: (1) based on the already existing translations and the ability to learn about translations, queries can be propagated to peers for which no direct translation link exists by means of transitivity and (2) the graph will have cycles. We call (1) *semantic gossiping*. (2) gives us the possibility to assess the degree of *semantic agreement* along a cycle, i.e., to measure the quality of the translations and the degree of semantic agreement in a community.

We expect peers to perform several task: (1) upon receiving a query, a peer has to decide where to forward the query to, based on a set of criteria overviewed below; (2) upon receiving results or feedback along translation cycles, it has to analyze the quality of the results at the schema and at the data level and adjust its criteria accordingly; and (3) update its view of the overall semantic agreement by modifying its query forwarding criteria or by adjusting the translation themselves.

The criteria to assess the quality of translations—which in turn is a measure of the degree of semantic agreement—can be categorized as *context-independent* and *context-dependent*. Context-independent criteria are syntactic in nature and relate only to the transformed query and to the required translation. We use the notion of *syntactic similarity* to analyze the extent to which a query is preserved after transformation. Context-dependent criteria relate to the degree of agreement that can be achieved among different peers upon specific translations. Such degrees of agreement may be computed using feedback mechanisms. We use two such feedback mechanisms, namely cycles appearing in the translation graph and results returned by different peers. This means that a peer will locally obtain both returned queries and data through multiple feedback cycles. In case a disagreement is detected (e.g., a wrong attribute mapping at the schema level or a concept mismatch at the content level), the peer has to suspect that at least some of the translations involved in the cycle were incorrect, including the translation it has used itself to propagate the query. Even if an agreement is detected, it is not clear whether this is not accidentally the result of compensating mapping errors along a cycle. Thus, analyses are required that assess which are the most probable sources of errors along cycles, to what extent the own translation can be trusted and therefore of how to use these translations in future routing decisions. At a global level, we can view the problem as follows: The translations between domains of semantic homogeneity (same schemas) form a directed graph. Within that directed graph we find cycles. Each cycle allows to return a query to its originator which in turn can make the analysis described above.

Each of these criteria is applied to the transformed queries and evaluated and results in a *feature vector* (see [Aberer *et al.*, 2003b] for details). The decision whether or not to forward a query using a translation link, i.e., whether a translation is assumed to be correct or not, is then based on evaluating these feature vectors.

Based on this approach the network converges to a state where a query is only forwarded to the peers most-likely understanding it, where the correct translations are increasingly reinforced by adapting the per-hop forwarding behaviors of the peers and where incorrect translations are rectified. Implicitly, this is a state where a global agreement on the semantics of the different schemas has been reached. Experimental results [Aberer *et al.*, 2003b] indicate that semantic agreement can be reached in a network of partially erroneous translations.

4.4 Method composition

Similarity values provided by local methods have to be aggregated. However, alignment and similarity assessment methods are also aggregated in order to compose a particular algorithm. For instance, one can first compute similarities between class names, then compute similarity between properties depending on how their names and classes to which they are attached are similar and then run a fix-point algorithm for computing interdependent similarities.

We can distinguish three ways to compose these methods:

built-in composition corresponds to most of the algorithms presented in next chapter: the chaining of methods is part of the algorithm and is applied to any data set which is given to the system.

opportunistic composition would correspond to a system which chooses the next method to run in function of the input data. We are not aware of any system working in that manner.

user-driven composition is used in environments in which the user has many different methods that she can apply following her will.

There are not a lot to be said on these three ways to compose the available methods. One system that proposes some starting point is Rondo [Melnik *et al.*, 2003]. Its goal is to propose a languages for composing the schemas and morphism. It thus defines Match and Merge operations (among many operations on their schemas ? Union, Difference ? and morphisms ? Compose, Invert). However, these two operators are placeholders for various matching and merging strategy which can be implemented as a combination of these other operators and SQL queries.

Most of the individual methods can be composed through a general purposes programming language.

4.5 User input

The support of effective interaction of the user with the system components is one concern of ontology alignment. User input can take place in many areas of alignment:

- for assessing initial similarity between some terms;
- for invoking and composing alignment methods (see above);
- for accepting or refusing similarity or alignment provided by the various methods.

4.5.1 Relevance feedback

In particular, the user feedback for each specific mapping extracted by the system could provide the alignment system with the ability of improving itself by changing the local alignment system parameters of the aggregation of the local alignments. However, asking users to specify this information is in general difficult since people conceptions change through their interaction with the alignment system. In any case, the main aspects of the user interaction could be summarised into the following questions:

- Given an alignment provided by the system, what is the structure and the way that the system can represent the user judgment for this specific alignment?
- How can the system take advantage of the above user feedback in terms of improving itself with the aid of this information?

Thus, the steps in implementing the user feedback are a) to gain an understanding of the nature of the this feedback and b) to specify an alignment system design and structure that supports and enhance it.

4.6 Alignment extraction

The ultimate alignment goal is a satisfactory set of correspondences between ontologies. A (dis)similarity measure between the entities of both ontologies provides a first set of correspondences. Those which will be part of the resulting alignment remains to be extracted with the help of the computed similarity.

This can be achieved during any of the global methods above if sufficiently constrained (for instance, to retain only one correspondence per entity). This can also be achieved afterwards by a specialised extracting method.

An alignment can be obtained by displaying the entity pairs with their similarity scores and/or ranks and leaving the choice of the appropriate pairs up to the user of the alignment tool. This user input can be taken as the definitive answer in helper environments, as the definition of an anchor for helping the system or as relevance feedback in learning algorithms.

One could go a step further and attempt at defining algorithms that automate alignment extraction from similarity scores. Various strategies may be applied to the task depending on the properties of the target alignment. As a matter of fact, one can ask the alignment to be complete (total) for one of the ontologies, i.e., all the entities of that ontology must be successfully mapped on the other side. Completeness is purposeful whenever thoroughly transcribing knowledge from one ontology to another is the goal. One can also require the mapping to be injective and hence reversible.

4.6.1 Thresholds

If neither ontology needs to be completely covered by the alignment, a threshold-based filtering would allows us to retain only the most similar entity pairs. Without the injectivity constraint, the pairs scoring above the threshold represent a sensible alignment.

The easier way to proceed consists in selecting correspondences over a particular threshold. Several methods can be found in the litterature [Ehrig and Sure, 2004]:

Hard threshold retains all the correspondence above threshold n;

Delta method consists in using as a threshold the highest similarity value to which a particular constant value d is substracted;

Proportional method consists in using as a threshold the a percentage of the highest similarity value;

Percentage retains the $n\%$ correspondences above the others.

4.6.2 Optimizing the result

In contrast, if an injective mapping is required then some choices need to be made in order to maximize the "quality" of the alignment that is typically measured on the total similarity of the aligned entity pairs. Consequently, the alignment algorithm must optimize the global criteria rather than maximizing the local similarity at each entity pair.

To sum up, the alignment computation may be seen as a less constrained version of the basic set similarity functions $MSim$ (see § 3.4.3). Indeed, its target features are the same: (i) maximal total similarity, (ii) exclusivity and (iii) maximal cardinality (in entity pairs). However, (ii) and (iii) are not mandatory, they depend on injectivity and completeness requirements, respectively.

A greedy alignment algorithm could construct the correspondences step-wise, at each step selecting the most similar pair and deleting its members from the table. The algorithm will then stop whenever no pair remains whose similarity is above the threshold.

The greedy strategy is not optimal: finding the global optimum would require the computing of a square assignment (polynomial assignment algorithms are suggested in [Papadimitriou and Steiglitz, 1998]). However, the ground on which a high similarity is forgotten to the advantage of lower similarities can be questioned and thus the greedy algorithm could be preferred in some situations.

Chapter 5

System presentations

The various methods presented above in isolation have been put together in order to implement ontology alignment or schema matching systems. There are a number of available systems that can be seen as addressing ontology alignment. We present some of them below through their principles and availability. Some of the following systems are developed by the projects partners and thus will be usable in order to benchmark them in the future.

There were some comparisons of these systems, in particular in [Do *et al.*, 2002; Rahm and Bernstein, 2001a; Kalfoglou and Schorlemmer, 2003b; Parent and Spaccapietra, 2000]. Our purpose here is not really to compare them, but rather to show their variety. Table 5.1 summarizes the kind of techniques implemented in each of these systems.

5.1 Prompt and Anchor-Prompt (Stanford SMI)

The Anchor-PROMPT [Noy and Musen, 2001] (an extension of PROMPT, also formerly known as SMART) is an ontology merging and alignment tool with a sophisticated prompt mechanism for possible matching terms. The anchor-PROMPT alignment algorithm takes as input two ontologies and a set of anchors-pairs of related terms, which are identified with the help of string-based techniques, or defined by a user, or another matcher computing linguistic (dis)similarity between frame names (labels at nodes), for example [McGuinness *et al.*, 2000]. Then it refines them based on the ontology structures and users feedback.

It constructs a directed labeled graph representing the ontology from the hierarchy of concepts (called classes in the algorithm) and the hierarchy of relations (called slots in the algorithm), where nodes in the graph are concepts and arcs are relations denoting relationships between concepts (the labels are the names of the relations). An initial list of anchors-pairs of related concepts defined by the users or automatically identified by lexical matching is the input for the algorithm. Anchor-PROMPT analyzes then the paths in the sub-graph limited by the anchors and it determines which concepts frequently appear in similar positions on similar paths. Based on these frequencies, the algorithm decides if these concepts are semantically similar concepts.

The PROMPT and Anchor-PROMPT systems have also contributed to the design of other algorithms such as PROMPTDiff which finds differences between two ontologies and provides the editing operation for transforming one ontology into another.

Page	System	T	TS	TL	I	S	ST	SC	E	M	U
53	**Multikat**	x		x	x	x	x				
49	**FCA-Merge**						x		x		
50	**IF-map**						x		x		
46	**APrompt**	x			x	x	x				x
48	**Cupid**	x	x	x	x	x	x				
62	**QOM**	x	x		x	x	x	x	x		
57	**OLA**	x	x		x	x	x	x	x		
48	**Rondo**		x			x		x			x
50	**T-tree**					x			x		
51	**S-match**	x	x	x			x			x	
53	**Buster**				x		x			x	
49	**Glue**								x		
48	**Chimerae**	x	x	x						x	x
50	**Artemis**	x		x		x					
52	**Coma**	x	x			x					x
56	**Asco**	x	x	x			x				
48	**MoA**			x							x
57	**Dogma**	x	x	x							
59	**ArtGen**			x							
60	**Bibster**	x	x		x		x				
63	**KILT**									x	

Table 5.1: Various contributions to alignment at a glance. The columns corresponds to categories of Chapter 3: Terminological (string- or language-based); Internal structure; Structure (terminological and cyclic); Extensional; semantics (Model-based) and User.

5.2 Chimerae (Stanford KSL)

Chimaera is an environment for merging and testing (diagnosing) large ontologies [McGuinness *et al.*, 2000]. Matching in the system is performed as one of the major subtasks of a merge operator. Chimaera searches for merging candidates as pairs of matching terms, involving term names, term definitions, possible acronym and expanded forms, names that appear as suffixes of other names. It also has techniques to identify terms that should be related by subsumption, disjointness, etc.

5.3 Rondo (Stanford U./U. Leipzig)

Rondo [Melnik *et al.*, 2003] is an environment for model (e.g., database schema) engineering which provides many unit primitives for manipulating models (extract, restrict, delete) and way to compose them. Among the unit primitives is the implementation of Similarity flooding (see § 4.2.1). It converts schemas (SQL DDL, XML) into directed labeled graphs whose nodes are candidate aligned pairs and arcs are shared properties. Arcs are weighted by their relevance to the nodes.

5.4 MoA (ETRI)

MOA[1] is an environment for merging ontologies developped by Electronics and Telecomunication Research Institute (ETRI) in South Korea. It is a library of methods and a shell for using them. It can work on OWL (but does not tell which flavor) and contains methods for importing, aligning, modifying and merging ontologies. Unfortunately, the methods are not known beside that they are based on (dis)similarity. The system uses Jena and Wordnet.

5.5 Cupid (Microsoft research)

The Cupid system [Madhavan *et al.*, 2001a] implements a generic schema matching algorithm combining linguistic and structural schema matching techniques, and computes normalized similarity coefficients with the assistance of a precompiled thesaurus. Input schemas are encoded as graphs. Nodes represent schema elements and are traversed in a combined bottom-up and top-down manner. Matching algorithm consists of three phases and operates only with tree-structures to which no-tree cases are reduced. The first phase (linguistic matching) computes linguistic similarity coefficients between schema element names (labels) based on morphological normalization, categorization, string-based techniques and a thesaurus look-up. The second phase (structural matching) computes structural similarity coefficients which measure the similarity between contexts in which individual schema elements occur in the schemas under consideration. The main idea behind the structural matching algorithm is to rely more on leaf level matches instead of the immediate descendents or intermediate substructures when computing similarity between non-leaf elements. The third phase (mapping generation) computes weighted similarity coefficients and generates final mappings by choosing pairs of schema elements with weighted similarity coefficients which are higher than a threshold. In comparison with the other hybrid matchers e.g.,

[1]http://mknows.etri.re.kr/moa

Dike [Palopoli *et al.*, 2000] and Artemis (see 5.9), referring to [Madhavan *et al.*, 2001a], Cupid performs better in the sense of mapping quality.

5.6 Glue (U. of Washington)

Glue [Doan, 2002] is an evolved version of LSD [Doan *et al.*, 2001] whose goal is to semi-automatically find schema mappings for data integration. Like its ancestor LSD, Glue use machine learning techniques to find mappings [Doan *et al.*, 2004]. It first applies statistical analysis to the available data (joint probability distribution computation). Then generates a similarity matrix, based on the probability distributions, for the data considered and use "constraint relaxation" in order to obtain an alignment from the similarity (see § 4.3.1). The algorithm works in three steps:

learning distributions the first phase is described above(see § 4.3.1), it learns the joint probability distributions of classes of each ontologies;

similarity estimation the system estimates the similarity between two classes in function of their joint probability distributions.

relaxation produces an alignment from the similarity matrix by using heuristic rules for choosing the more likely correspondences.

5.7 FCA-merge (U. Karlsruhe)

FCA-merge [Stumme and Mädche, 2001] uses formal concept analysis techniques to merge two ontologies sharing the same set of instances. The overall process of merging two ontologies consists of three steps:

1. instance extraction,
2. concept lattice computation, and
3. interactive generation of the final merged ontology.

The algorithms theoretically merges two ontologies sharing the same set of instances. However, the authors provide, as first step, methods for extracting the instances from documents. The extraction of instances from text documents circumvents the problem that in most applications there are no individuals which are simultaneously instances of the source ontologies, and which could be used as a basis for identifying similar concepts.

The computation of the lattice starts with two ontologies and instances belonging to both ontologies. From these, it computes two formal contexts, i.e., boolean tables indicating which instance belongs to which concept of the of the ontology. It then merges both contexts (by renaming the concepts and adding both contexts). Using classical formal concept analysis (i.e., the closure of an instances$\times$properties Galois connection [Ganter and Wille, 1999]) on contexts made of instances$\times$concepts, the method generates a pruned concept lattice. The lattice is pruned of all the concepts which are not more general than a concept of one of the ontologies.

The last step consists in helping a user to further simplify the lattice and generate the taxonomy of an ontology. The produced result is explored and transformed to a merged ontology by the ontology engineer. The final step of deriving the merged ontology from the concept lattice requires human interaction.

The result is rather a merge than an alignment. However, the concepts that are merged can be considered as exactly aligned and those which are not can be considered in subsumption relation with their ancestors or sibblings.

5.8 IF-Map

Another system inspired by formal concept analysis is IF-Map [Kalfoglou and Schorlemmer, 2003a]. It is an automatic method for ontology mapping based on the Barwise-Seligman theory of information flow [Barwise and Seligman, 1997]. The basic principle of IF-map is to align two local ontologies by looking at how these are mapped from a common reference ontology. It is assumed that such reference ontology is not populated with instances, while local ontologies usually are. IF-Map generates possible mappings between an unpopulated reference ontology and a populated local ontology by taking into account how local communities classify instances with respect to their local ontologies.

5.9 Artemis (U. Milano/U.Modena and Reggio Emilia)

Artemis (Analysis of Requirements: Tool Environment for Multiple Information Systems) [Castano *et al.*, 2000] was designed as a module of MOMIS mediator system [Bergamaschi *et al.*, 1999; 1998] for creating global views. Artemis does not cover all the issues of matching due to the origin function of schema integration. The matching algorithm performs affinity-based analysis and hierarchical clustering of source schemas elements. Affinity-based analysis is carried out through computation of the name, structural and global affinity coefficients by exploiting a common thesaurus. The common thesaurus presents a set of terminological and extensional relationships which depicts intra- and inter-schema knowledge about classes and attributes of the input schemas, which is built with the help of WordNet [Miller, 1995] and ODB-Tools [Beneventano *et al.*, 1998]. A hierarchical clustering technique exploiting global affinity coefficients categorizes classes into groups at different levels of affinity. For each cluster it creates a set of global attribute global class. Logical correspondence between the attributes of a global class and source attributes is determined through a mapping table.

5.10 T-tree (INRIA Rhône-Alpes)

Troeps [Mariño *et al.*, 1990] was a knowledge representation system enabling several class taxonomies (called viewpoints) over the same set of objects and bridges between these classes expressed equivalence or subsumption. T-tree [Euzenat, 1994] is an environment for generating taxonomies and classes from objects (instances). It can, in particular, infer dependencies between classes (bridges) of different ontologies sharing the same set of instances based only on the "extension" of classes.

An algorithm has been developed which is able to infer bridges. The bridge inference algorithm, given a set of source viewpoints and a destination viewpoint (built by T-Tree or by any other mean), returns all the bridges (in a minimal fashion) which are satisfied by the available data. That is the set of bridges for which the objects in every source class are indeed in the destination class. The algorithm compares the extension (set of instances) of the presumed destination to the intersection of these of the presumed source classes. If there is no inclusion of the latter in the former, the algorithm is re-iterated on all the sets of source classes which contain at least one class which is a sub-class of the tested source classes. If the intersection of the extension of the presumed source classes is included in that of the presumed destination class, then a bridge can be established from the latter (and also from any set of sub-classes of the source classes) to the former (and also any super-class of the destination class). But other bridges can exists on the sub-classes of the destination. The algorithm is thus re-iterated on them. It stops when the bridge is trivial, i.e. when the source is empty.

The algorithm is extension-correct (only valid bridges are inferred), extension-complete (all valid bridges are inferred) and extension-minimal (only more general bridges are inferred). The proof is carried out in the classification scheme framework and the "extension-" prefix just tells that what is considered is only the extension of the classes (the algorithm tests set inclusion on classes). Thus these results are not semantically grounded. For instance, is that a coincidence that all directors have formerly been at the same university? Maybe, maybe not. Hence the user has to decide the validation of inferred bridges. This has to be contrasted with a stronger kind of bridge inference based on the structural constraints on classes. But indeed, any possible bridge compliant with the current set of objects and the semantics must be a restriction of one of the bridges provided by the algorithm.

Bridge inference is nothing else than the search for correlation between two sets of variables. This correlation is particular from a data analysis point of view since it does not need to be valid on the whole set of individuals (the algorithm looks for subsets under which the correlation is valid) and it is based on strict set equality (not similarity). However, even if the bridge inference algorithm has been described with set inclusion, it can be helped by other measurements which will narrow or broaden the search. More generally, the inclusion and emptiness tests can be replaced out by tests based on the similarity of two sets of objects (as it is usual in data analysis). In fact, many parameters can be taken into account when inferring bridges; for that purpose, the algorithm is function of the meaning of the operators $\subseteq$, $\cap$ and $= \emptyset$-test. A second version of the algorithm (with the same properties) were made available and used structural comparison: $\subseteq$ is subtyping, $\cap$ is type intersection and $= \emptyset$-test is a sub-typing test.

5.11 S-MATCH (U. Trento)

S-Match is a schema/ontology matching system that implements the semantic matching approach [Giunchiglia *et al.*, 2004; Bouquet *et al.*, 2003]. It takes two graph-like structures (e.g., database schemas or ontologies) as input and returns semantic relations between the nodes of the graphs, that correspond semantically to each other, as output. Possible semantic relations are: equivalence ($=$), more general ($\sqsupseteq$), less general ($\sqsubseteq$), mismatch ($\perp$) and overlapping ($\sqcap$).

The current version of S-Match is a rationalized re-implementation of the CTXmatch system [Bouquet *et al.*, 2003] with a few added functionalities. S-Match is schema based, and, as such, it does not exploit the information encoded in data instances. S-Match is a hybrid system performing composition of element level techniques. At present, S-Match allows it to handle only tree-like structures (e.g., taxonomies or concept hierarchies).

S-Match was designed and developed as a platform for semantic matching, namely a highly modular system with the core of computing semantic relations where single components can be plugged, unplugged or suitably customized. The logical architecture of the system is depicted in Figure 5.1.

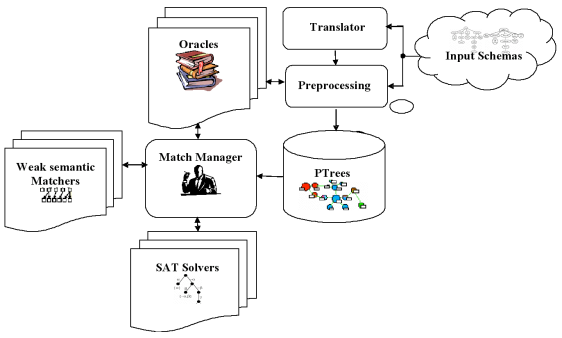


Figure 5.1: Architecture of the S-match platform

The *input schemas* (trees) are codified in a standard internal XML format. This internal format can be loaded from a file that is manually edited, or can be produced by an input format dependent translator. The module taking input schemas/ontologies does the preprocessing. In particular, it computes in a top-down manner for every label in a tree the meaning captured by the given label in a schema or ontology using the techniques described in [Magnini *et al.*, 2003]. The preprocessing module has access to the set of oracles which provide the necessary a priori lexical and domain knowledge. In the current version WordNet [Miller, 1995] is the only oracle. The output of the module is an enriched tree. These enriched trees are stored in an internal database (PTrees) where they can be browsed, edited and manipulated.

The Matching Manager coordinates matching process using three extensible libraries. The first library is contained of, what is called in [Giunchiglia *et al.*, 2004], weak semantics element level matchers. They perform string manipulations (e.g., prefix, n-grams analysis, edit distance, soundex, data types, and so on) and try to guess the semantic relation implicitly encoded in similar words. The current version of S-Match contains 13 weak semantics element level matchers. The second library is contained of strong semantics element level matchers, namely oracles. Currently, WordNet is the only oracle. The third library is contained of structure level strong semantics matchers, namely SAT solvers (among the others, the SAT deciders that we are currently testing is JSAT [Berre, 2001] and Open4J by Daniel Le Berre).

S-Match is implemented in Java 1.4 and the total amount of code (without optimizations!) is around 60K.

5.12 Coma (U. Leipzig)

The COMA system [Do and Rahm, 2002] is a generic schema matching tool, which implements composite generic matchers. COMA 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. Matching library is extensible, and as from [Do and Rahm, 2002] it contains 6 individual matchers, 5 hybrid matchers, and one “reuse-oriented” matcher. Most of them implement string-based techniques as a background idea; others share techniques with Cupid (see § 5.5) but reuse-oriented is a completely novel matcher, which tries to reuse previously obtained results for entire new schemas or for its fragments. Schemas are internally encoded as rooted directed acyclic graphs, where elements are the paths. This aims at capturing contexts in which the elements occur. One of the distinct features of the COMA tool is the capability to perform iterations in matching process. It presumes interaction with a user which approves obtained matches and mismatches to gradually refine and improve the accuracy of match.

Based on the comparative evaluations conducted in [Do *et al.*, 2002], COMA dominates Autoplex&Automatch [Berlin and Motro, 2001; 2002] LSD [Doan *et al.*, 2001], Glue [Doan *et al.*, 2003], SF [Melnik *et al.*, 2002] and SemInt [Li and Clifton, 1994b] matching tools.

5.13 Buster (U. Bremen)

The Bremen University Semantic Translator for Enhanced Retrieval (BUSTER) [Visser *et al.*, 2002] is an information broker middleware that was built to access heterogeneous and distributed information sources and to assess their conceptual, spatial, and temporal relevance with respect to a specific information request. BUSTER can also be used to integrate heterogeneous information through the resolution of structural, syntactical, and semantic heterogeneities. To be more precise, the BUSTER system provides two subsystems, one for information filtering and one for information integration.

The BUSTER search module supports the specification of queries of the type concept @ location in time [Vögele *et al.*, 2003]. In addition to the conceptual semantics, the system evaluates the spatial as well as the temporal relevance of an information source. In order to be able to reason about conceptual, spatial, and temporal relevance, BUSTER utilises metadata that provide formal descriptions of the respective context of an information source.

In principle, the main difference with respect to other system for query processing and information integration lies in the fact that the user does commit to a basic vocabulary that is used to define concepts in all the source ontologies. The basic vocabulary ensures that different source ontologies are comparable to each other. By formulating the query in terms of this basic vocabulary we the query can be interpreted with respect to all source ontologies in the system. In particular, each concept base on the shared vocabulary and can be constructed with the help of some construction operators like $\sqcap, \sqcup$ well-known from description logics. Because each concepts also from different source ontologies can be flatten to terms which only consists of elements of the shared vocabulary combined with some construction operators, they can easily compared with respect to equality ($\equiv$), subsumption ($\sqsubseteq$), overlap ($C \sqcap D$ is consistent), and inconsistence ($C \sqcap D$ is inconsistent). In other words, BUSTER can automatically determine these concepts in a source ontology that are most similar to the concept we asked for.

5.14 MULTIKAT (INRIA Sophia Antipolis)

MULTIKAT [Dieng and Hug, 1998a; 1998b] is a tool enabling comparison and merging of two ontologies, represented in Sowa's conceptual graph formalism [Sowa, 1984]. In this formalism, an ontology is represented through a support (i.e. a hierarchy of concept types, a hierarchy of relation types, a set of markers for identifying the instances and a conformity relation enabling to determine which types are compatible with a given marker).

The building of an integrated ontology from two ontologies relies on the following steps:

1. Comparison and merging of the two concept type hierarchies: this step enables to solve name conflicts and in case of need, to add new concept types and to adapt concept type definitions.

2. Comparison and merging of the two relation hierarchies: this step enables to solve name conflicts, and in case of need, to add new relation types, to adapt relation type definitions and to adapt relation type signatures.
3. Comparison and merging of the two sets of markers: this phase helps to solve name conflicts and to adapt the conformity relation.

MULTIKAT relies on a cooperative approach: the knowledge engineer can use MULTIKAT editor to tune the parameters and weights used in the mapping and merging algorithms.

5.14.1 Mapping of two types in both hierarchies

The mapping algorithm aims at determining, in two concept (resp. relation) type hierarchies, which types are identical. It relies on two phases:

Phase 1: terminology-based mapping During this first phase, MULTIKAT algorithm tries to identify which types of both hierarchies are similar, according to their main names and their synonyms. The knowledge engineer can combine several criteria and assign them different weights so as to privilege some criteria:

- t_1 and t_2 have the same main name,
- the number of common synonyms of t_1 and t_2 is greater than a given threshold,
- the main name of one type belongs to the list of synonyms of the other type.

This similarity function $Sim_1 : H_1 \times H_2 \rightarrow \mathbb{R}$ computes the similarity measure $Sim_1(t_1, t_2)$ between t_1, a type of H_1 and t_2, a type of H_2, according to this first identification phase, and its results are stored in a similarity matrix. After this phase, two types t_1 and t_2 are 1-similar iff Sim1 (t1, t2) is greater than a threshold Tsimilar.

Phase 2: context-based mapping In this second phase, the mapping algorithm now considers the contexts of the types to be compared. The context of a type consists of its relatives (i.e. its direct supertypes and its direct subtypes) in the type hierarchy. In this second phase, the algorithm tries to identify which types of both hierarchies are the same, according to their contexts. Three mapping cases are distinguished:

- The number of 1-similar direct supertypes (resp. direct subtypes) of t_1 and t_2 are greater than a threshold $Tpred$ (resp. $Tsucc$)
- All the direct supertypes (resp. direct subtypes) of t_1 and t_2 are 1-similar.
- The set of relatives of t_1 (resp. t_2) is included in the set of relatives of t_2 (resp. t_1) w.r.t. 1-similarity.

The knowledge engineer can associate different weights to these three cases. Another similarity function $Sim_2(t_1, t_2)$ is computed. If t_1 is the type numbered i in Hier1 and t_2 the type numbered j in Hier2, then, in the final similarity matrix $SimMatr$:

$$SimMatr_{ij} = Sim_1(t_1, t_2) + Sim_2(t_1, t_2)$$

The couples of identical types are computed from this similarity matrix. After the second phase, the types t_1 and t_2 are considered as identical iff $SimMatr_{ij}$ is the maximum value in the ith line and the jth column in the matrix, and this value is greater than a threshold Tsame.

Two comparison strategies can be applied:

One-to-one algorithm For each cycle of comparison of two previous identification phases, the algorithm compares each type of H_1 to each type of H_2.

Hierarchy-match algorithm This algorithm takes into account the hierarchical structure in its comparison strategy. It relies on a depth-first search in both hierarchies and it proceeds as follows: once two identical types have been found, then a search for further mappings in their sub-hierarchies is performed. In this algorithm, the thresholds $T_{similar}$ and T_{same} have the same values. In both previous phases, after each evaluation of a couple of types (t_1, t_2), the corresponding value $SimMatr_{ij}$ is compared to T_{same}. As soon as $SimMatr_{ij} > T_{same}$, then the pair (t_1, t_2) is included in the set IdenticalConceptTypes.

5.14.2 Merging of concept type hierarchies

The knowledge engineer can initialize the set IdenticalConceptTypes by indicating which types of both hierarchies are already known as identical. The mapping algorithm is applied (either with a one-to-one match strategy or with a hierarchy match strategy). Then, before the merging, the partial ordering relation of identical types is checked. The couples, responsible for violation of the merging precondition are eliminated from IdenticalConceptTypes. Then the integrated hierarchy T_{ccom} is built by representing each couple of identical types by a single type in T_{ccom} and by adding the types appearing in only one hierarchy. If a type is present only in one ontology and cannot be mapped to any type of the second ontology, it will be kept in the integrated hierarchy, with a prefix in its name indicating from which expert it comes from. If a type is present in both hierarchies, the experts can choose its final main name stored in T_{ccom}. In all cases, the associated synonyms are also stored in T_{ccom}.

The algorithm tries to detect terminology conflicts, topology conflicts and conflicts specific to conceptual graph formalism.

5.14.3 Comparison and merging of the relation type hierarchies

The mapping algorithm for relation type hierarchies is similar to the 2-phase-based algorithm previously presented. Once obtained the set IdenticalRelationTypes of pairs of identical relation types, the precondition for merging of the two hierarchies must also be checked. When two relation types are considered as identical, a verification of their signature compatibility must be performed. The signatures in the integrated relation type hierarchy Trcom must be adapted according to the integrated concept type hierarchy Tcom.

If a relation type is present only in one ontology, its signature is preserved in the integrated ontology. The signature of the integrated relation type obtained from two identical relations types relies on the supremum of the concept types appearing in their signatures.

5.14.4 Comparison and merging of the marker sets

The terminology-based mapping algorithm is used for the set of markers. When two markers are identical, their conformity relation must be compatible, otherwise they are eliminated from the set IdenticalMarkers.

5.14.5 Implementation

MULTIKAT was implemented in C/C++ and JAVA, above the conceptual graph platform, COGITO (developed by the LIRMM) and was applied in traffic accident analysis.

5.15 ASCO (INRIA Sophia-Antipolis)

ASCO prototype relies on an algorithm that identifies the pairs of corresponding elements in two different ontologies [Bach *et al.*, 2004]. These pairs may be pairs of concepts (classes) in the two ontologies or pairs of relations, or even pairs of a concept in one ontology and a relation in the other ontology.

ASCO tries to use as much as possible available information contained in the ontology for the process of matching two ontologies. This information consists of identifiers (names), labels, comments of concepts, identifiers, labels, comments, domain, range of relations, structure of the taxonomy of concepts or of relations, data instances of ontology, annotations, axioms, rules. So far, in its matching process, ASCO already takes into account some of above information such as identifiers, labels, comments of concepts, identifiers, labels, comments, domain, range of relations, structure of the taxonomy of concepts or of relations.

The matching process of ASCO is composed of several phases. The linguistic phase applies linguistic processing techniques, and uses string comparison metrics, and lexical databases such as WordNet to compute the similarity of two concepts or two relations. In the linguistic processing step, ASCO normalizes firstly terms, expressions thanks to punctuation, upper case, special symbols, digits to have a set of tokens. These tokens are then compared using string comparison metrics such as Jaro-Winkler, Levenstein or Monger-Elkan. Based on token similarities, the similarity of sets of tokens is computed. To increase the accuracy and to avoid the problems of term conflicts, a lexical database such as WordNet is integrated. To compute the similarity between long texts (for example, between the comments or descriptions of classes or of relations), ASCO uses Term frequency/Inverse document frequency metrics after applying a linguistic processing step to eliminate all of the stopwords in long texts.

The computed linguistic similarities are input for the structural phase. In this phase, ASCO tries to exploit the structure of ontology taxonomy for modifying or asserting the similarity of two concepts or relations. The similarities of classes or of relations are iteratively propagated to their neighbors in the tree of ontology which is built from the hierarchy of classes and the hierarchy of relations. When the propagation terminates (the class similarities and the relation similarities do not change after an iteration or a certain number of iterations is reached), if the similarities between classes or relations exceed a threshold, they are considered as similar. ASCO runs now on the two above phases.

ASCO algorithm was implemented in Java. It is built on Corese (Conceptual Resource Search Engine), the semantic search engine developed by ACACIA team [Corby *et al.*, 2000; Corby and Faron, 2002; Corby *et al.*, 2004]. Corese loads ontologies from RDF(S) files into memory, these ontologies are then supplied to ASCO. ASCO was tested with two real-world ontologies: O'COMMA, which has 472 concepts and 77 relations [Gandon, 2002]; and O'Aprobatiom, which has 460 concepts and 92 relations.

5.16 OLA (INRIA Rhône-Alpes & UoMontréal)

OLA [Euzenat and Valtchev, 2003; Euzenat and Valtchev, 2004] is a class of algorithm for ontology alignments which targets the following characteristics:

- covering all the possible characteristics of ontologies (i.e., terminological, structural and extensional);
- taking care of collection structures (lists, sets) and accounting for them during matching;
- expliciting all recursive relationships and finding the best matching through iteration.

OLA is currently implemented for ontologies described in OWL-Lite [Euzenat and Valtchev, 2003]. It uses the Alignment API and implementation that we recently developed [Euzenat, 2004].

The algorithm first compiles the OWL 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). This similarity is a weighted linear aggregation of the similarity measures between all the entities a couple of entities is in relation. This accounts for all the relationships between entities. However, these features (like subclasses) are sets of entities, the similarity between these sets of entities, thus depends on a local matching between these entities. A matching of both sets is considered which is: (i) of maximal total similarity, (ii) exclusive, and (iii) of maximal size [Valtchev, 1999].

Similarity between labels can be produced by any kind of particular terminological method (e.g., string distance, linguistic evaluation). Similarity between data values and data types can be provided by specilised external similarity measures (e.g., Euclidean distance, symmetric difference distance) [Valtchev, 1999].

The definition of this similarity provides a set of equations whose variables are the similarity values between entities of the ontologies. This set of equation cannot be solved directly due to local matching. As a matter of fact, depending on the currently computed similarity, the matching as defined above can be different. We thus developed an iterative algorithm which compute a first approximation of the similarity (without the local matching), then compute the local matching and reiterate. We proved that this algorithm is converging towards a solution, mainly because the similarity is always improving over the iterations. It can be that this solution is not the global optimum so the algorithm should be launched several times.

From this solution, it is possible to extract an alignment between the two ontologies (by retaining the correspondence whose similarity is over a certain threshold, or by optimising the selections of couples).

5.17 Dogma's Ontology Integration Methodology

Dogma's ontology integration methodology adopts for ontology integration the same methodological steps that were singled out in database schema integration [Batini *et al.*, 1986]:

$$\begin{array}{ccccccc} \text{Step 1} & & \text{Step2} & & \text{Step3} & & \text{Step4} \\ \text{Preintegration} & \longrightarrow & \text{Alignment} & \longleftrightarrow & \text{Conforming} & \longrightarrow & \text{Merging and Restructuring} \end{array}$$

The double sided arrow between step 2 and 3 denotes a loop. An alignment rule is suggested in step 2 and step 3 checks if it would cause inconsistencies when approved and stored in the alignment rule repository.

Preintegration

Preintegration consists of an analysis of the ontologies to decide the general integration policy: choosing the ontologies to be integrated, choosing the strategy of integration, deciding the order of integration, and possibly assigning preferences to entire ontologies or portions thereof. All these decisions have to be made by humans. We adopt the *binary ladder strategy* to integrate ontologies [Batini *et al.*, 1986].

Comparison of Ontologies: Ontology Alignment

An *alignment* between two Dogma inspired ontologies is defined as a commitment (i.e. an interpretation) of the source ontology's lexon base in terms of the target ontology's lexon base. A commitment consists of a set of *commitment rules*, here (for ontology integration purposes) also called *mapping rules*. We discuss these mapping rules in more detail in the sections that follow.

The fundamental activity in this step consists of detecting and resolving several kinds of heterogeneities like semantically equivalent concepts which are denoted by means of different terms in both ontologies and identifying inter ontology relationships between concepts of different ontologies.

Identifying equivalent concepts: Because ontology mismatches often occur through ambiguity of terms, miss-spelled terms, usage of different terminology to denote the same concept etc., we will always consider concepts in the comparison phase instead of the term labels that represent them. The degree of similarity between two concepts c_1 and c_2 is measured by means of a *similarity score*, formally stated as: $\mathbf{sc(c_1, c_2)} : \mathbf{C} \times \mathbf{C} \rightarrow [\mathbf{0}, \mathbf{1}]$. The way this similarity score is computed depends on how the concepts are defined. This is discussed next.

In case the concepts are WordNet synsets we can make use of the freely available software package, called *WordNet::Similarity*[2][Pedersen *et al.*, 2004], to measure the semantic similarity and relatedness between a pair of concepts.

If the concepts cannot be identified with existing WordNet synsets we compute the similarity score between two concepts $c_1 \equiv t_1^1, \ldots, t_n^1$ and $c_2 \equiv t_1^2, \ldots, t_m^2$ by computing the similarity score between all possible pairs of terms where the two terms in a pair come from different concepts. A similarity score between natural language terms is the weighted sum of similarity scores based on natural language and string processing techniques, like: Porter Stemmer, Metaphone, Levenshtein, longest common prefix/suffix, longest common substring, the theory of distributionalism. All these techniques are based on syntactic differences between terms and do not take any semantic value of them into consideration. Therefore we have to be very critical with the interpretation of these results. If the similarity score is above a given threshold then the concepts are considered to be equivalent. The treshold can be modified by the expert performing the alignment.

Identifying inter ontology relationships: In order to identify inter ontology relationships between concepts of different ontologies we distinguish the following set of relationships with predefined semantics: *SubClassOf, Generalize, PartOf, InstanceOf.* We have developed a methodology that allows to automate the task of finding inter ontology relationships between concepts [De Bo

[2]http://sourceforge.net/projects/wn-similarity

et al., 2004a; 2004b].. This methodology uses a formal upper level ontology, called SUMO (Suggested Upper Merged Ontology), which has been proposed as the initial version of an eventual Standard Upper Ontology (SUO)[Niles and Pease, 2001]. In a nutshell the methodology works as follows: Each concept used in the ontology is aligned with an appropriate SUMO concept. Since SUMO is a formal upper level ontology we can make use of its axioms to derive relations that hold between SUMO concepts to the ontology level.

Conforming of Ontologies: Instant Validation

Each time an alignment rule is proposed by the system or the user the rule is instantly checked whether it conflicts with other rules that have already been added to the alignment rule repository. In order to resolve conflicts in an automatic manner algorithms have been developed that detect conflicting situations caused by misalignments, look for cycles, etc.

Ontology Merging and Restructuring

During this activity the (conformed) ontologies are superimposed, thus obtaining a global ontology. The merge process is essentially based upon the mapping rules established in the comparison phase 5.17 and results in an algebra for merging. We distinguish following merge operators: *Merge, Specialize, Generalize, PartOf, InstanceOf*. All operators might require restructuring of the merged ontology.

5.18 ArtGen (Stanford U.)

In [Mitra and Wiederhold, 2002] the authors propose a semi-automated algorithm for resolving terminological heterogeneity among the ontologies and establishing the articulation rules necessary for meaningful interoperation. This algorithm forms the basis of the *articulation generator* for the ONION (ONtology compositION) system. The automated articulation generator (ArtGen) of ONION suggests articulation rules to the user performing the matching process. A human expert can either accept, modify or alter the suggestions. The expert can also indicate new matches that the articulation generator might have missed.

The authors distinguish two types of articulation rules:

linguistic matching rules Concept names are represented as a string of words. The linguistic matcher compaires all possible pairs of words from any two concepts of both ontologies and assigns a similarity score to each pair. The matcher uses a word similarity table generated by a word relator (Thesaurus-Based word relator or Corpus-Based word relator) to look up the similarity between all possible pairs of words. The similarity score between two concepts is the average of the similarity scores (different from zero) of all possible pairs of words in their names. The linguistic matching rule does not indicate the exact semantic relationship between the two concepts, for example, whether they have a class-subclass relationship, or are equivalent etc.

inference rules An inference engine in Datalog is capable of making logical derivations based on the inference rules available in the engine.

The ontologies that were used for the experiments were represented in RDF and contained 30 respective 50 nodes, which are very small ontologies. The authors demonstrate how the articulation rules are generated by the ONION system. The tool was evaluated by computing precision and recall measures for the corpus and thesaurus based word relators. Accuracy was measured by comparing the results of the automated articulation generator with those expected by the expert. If the expert deleted a match of the articulation generator then precision is lowered. In case the expert added a match that was not found by the articulation generator then recall is lowered.

The thesaurus-based method resulted in very poor results, thought the corpus-based method produced better results. However scalability was extremely low and the quality of the results were very dependent on the quality of the corpus available. When everything was pre-computed, the corpus-based method scaled very well.

5.19 Alimo (ITI-CERTH)

The development and maintenance of large multimedia databases has attracted much attention nowadays from companies and organizations that held multimedia content (archives, broadcast companies, radio and TV channels etc). The goal is to bypass the ineffective and time-consuming process of manual searching and retrieval of multimedia content and use computers to make the content easy to be found and accessible to other parties. Thus, two critical points are identified in making the above goal a reality; effective representation as well as effective retrieval and exploration of multimedia content. For accomplishing the above goal researchers have started to use ontologies in the field of multimedia in order to construct machine-understandable, descriptive versions of the multimedia content based on multimedia ontologies. Four different levels of information are represented in multimedia ontologies: signal information, featural information, symbolic information and semantic information.

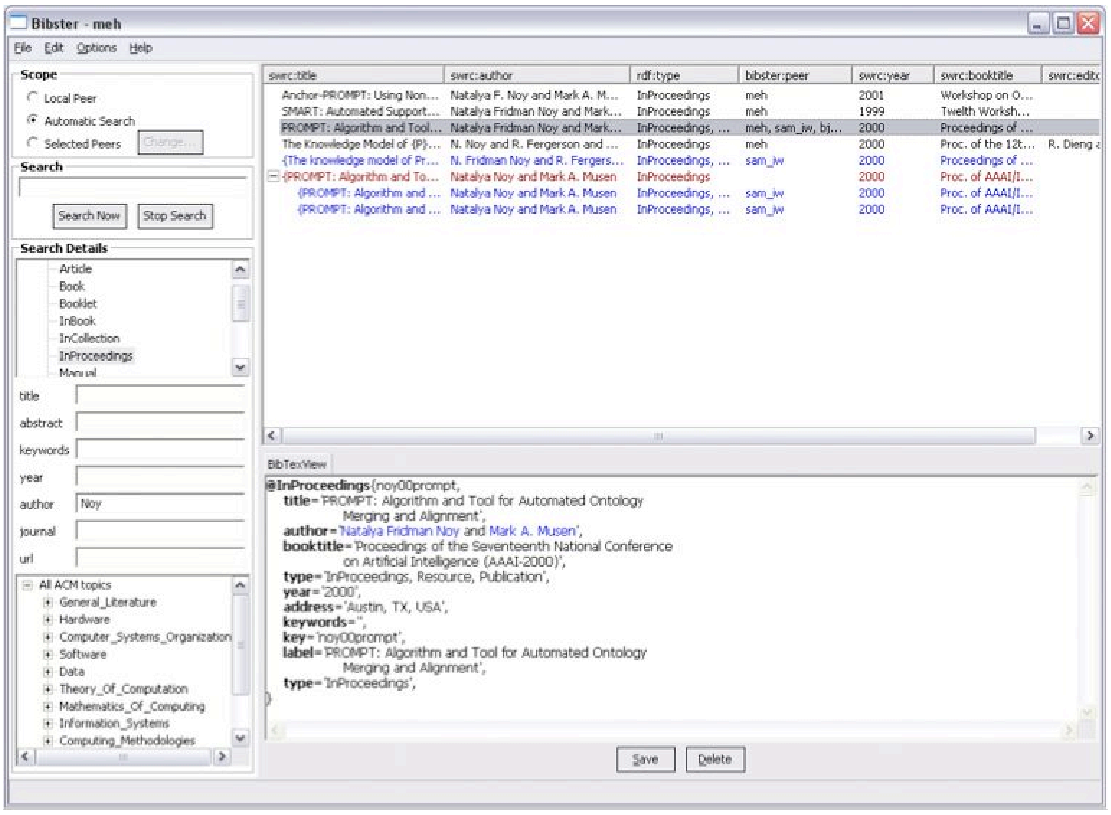
With the aid of multimedia ontologies the vision of querying and retrieving multimedia content from distributed databases has started to become more feasible. But in order for someone to be able to use all the levels of information, from the semantic to the raw audiovisual one, a proper alignment framework should be provided. For this reason ITI-CERTH is constructing ALIMO (Alignment of Multimedia Ontologies), an ontology alignment system that pay special care to each one of the subparts of a multimedia ontology and the attributes with the special meaning and structure. Semantic descriptions will be aligned using methods hybrid alignment systems (terminological, structural etc). The signal description parts will be compared by using visual matching algorithms from the field of digital image and video processing. The feature description by examining the XML schema of the MPEG-7 visual part and at last the symbolic description by referring to the definitions of the concepts that those labels are instances of, and also by examining the datatypes of the attributes assigned to those instances.

5.20 Bibster (U. Karlruhe)

Bibster[3] [Broekstra *et al.*, 2004] addresses a typical problem in the daily life of a computer scientist, where one regularly has to search for publications or their correct bibliographic metadata. The scenario that we support here is that researchers in a community share bibliographic metadata in a Peer-to-Peer fashion.

[3]http://bibster.semanticweb.org

Bibster is a Peer-to-Peer system based on the SWAP architecture[4], which allows to easily integrate, share and search bibliographic metadata using semantic technologies for the representation of the bibliographic instances and the peers' expertise to allow effectively route queries. Semantic similarity measures identifying duplicates allow to visualize and to integrate the heterogeneous search results from the peers. Bibliographic entries are extracted from BibTex into an ontology. The query results themselves represent small ontologies, containing duplicates.



swrc:title	swrc:author	rdf:type	bibster:peer	swrc:year	swrc:booktitle	swrc:edit
Anchor-PROMPT: Using Non...	Natalya F. Noy and Mark A. M...	InProceedings	meh	2001	Workshop on O...	
SMART: Automated Support...	Natalya Fridman Noy and Mark...	InProceedings	meh	1999	Twelth Worksh...	
PROMPT: Algorithm and Tool...	Natalya Fridman Noy and Mark...	InProceedings, ...	meh, sam_jw, bj...	2000	Proceedings of ...	
The Knowledge Model of {P}...	N. Noy and R. Fergerson and ...	InProceedings	meh	2000	Proc. of the 12t...	R. Dieng
{The knowledge model of Pr...	N. Fridman Noy and R. Fergers...	InProceedings, ...	sam_jw	2000	Proceedings of ...	
{PROMPT: Algorithm and To...	Natalya Noy and Mark A. Musen	InProceedings		2000	Proc. of AAAI/I...	
{PROMPT: Algorithm and ...	Natalya Noy and Mark A. Musen	InProceedings, ...	sam_jw	2000	Proc. of AAAI/I...	
{PROMPT: Algorithm and ...	Natalya Noy and Mark A. Musen	InProceedings, ...	sam_jw	2000	Proc. of AAAI/I...	

```
@InProceedings{noy00prompt,
   title='PROMPT: Algorithm and Tool for Automated Ontology
            Merging and Alignment',
   author='Natalya Fridman Noy and Mark A. Musen',
   booktitle='Proceedings of the Seventeenth National Conference
            on Artificial Intelligence (AAAI-2000)',
   type='InProceedings, Resource, Publication',
   year='2000',
   address='Austin, TX, USA',
   keywords='',
   key='noy00prompt',
   label='PROMPT: Algorithm and Tool for Automated Ontology
            Merging and Alignment',
   type='InProceedings',
}
```

Figure 5.2: Screenshot of the Bibster Application

Finding duplicates is closely related to finding corresponding mappings. In both cases it is necessary to recognize identical objects despite their different identifiers.

In the given scenario duplicates are bibliographic entries which refer to the same publication or person in the real world, but are modelled as different resources. The similarity function is based on different features of the respective instances. For persons one can refer to the name. For publications to title, authors, editors, journal, address, type of publication, etc. The function returns a value between 0 and 1 by applying specific heuristics to every feature: Strings are compared using the Levenshtein distance [Levenshtein, 1966], the authors of publications are compared by comparing the two sets. Some domain specific features require special heuristics: if the type of one publication is "Misc", this only means that no further information about the type was available. If another publication is e.g. type "Article" the similarity is set to 0.5 rather than 0. Besides individual functions our approach focuses on applying an aggregation function to achieve an overall

[4]http://swap.semanticweb.org

similarity. Through transitive closure we receive a set of "identical" entities. Instead of presenting all instances of the query result, duplicates are visualized as one, merged, resource. These merged resources consist of a union of properties of the individuals identified as duplicates.

After several rounds of testing Bibster is now openly available [5], with the component based on alignment working in the background of the system.

5.21 QOM (U. Karlsruhe)

QOM considers both the quality of mapping results as well as the run-time complexity. The hypothesis is that mapping algorithms may be streamlined such that the loss of quality (compared to a standard baseline) is marginal, but the improvement of efficiency is so tremendous that it allows for the ad-hoc mapping of large-size, light-weight ontologies. To substantiate the hypothesis, a number of practical experiments were performed.

The outcome is QOM — Quick Ontology Mapping [Ehrig and Staab, 2004]. It is defined by the steps of a process model as shown in Figure 5.3. Mapping one ontology onto another means that for each entity (concept C, relation R, or instance I) in ontology O_1, we try to find a corresponding entity, which has the same intended meaning, in ontology O_2.

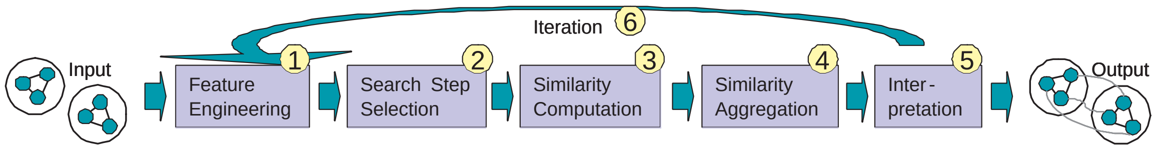


Figure 5.3: QOM Mapping Process

1. Firstly, QOM uses RDF triples as features.

2. Second, instead of comparing all entities of the first ontology with all entities of the second ontology, QOM uses heuristics to lower the number of candidate mappings, which is a major problem for run-time complexity. In this dynamic programming approach we only choose promising candidate mappings.

3. The actual similarity computation is done by using a wide range of similarity functions [Ehrig and Staab, 2004]. An entity is described by the kind of appearance that is found to hold for this entity for characteristics like: identifiers such as URIs, RDF/S primitives such as subclass and instance relations, or domain specific features e.g. a *hashcode-of-file* in a file sharing domain. These features of ontological entities are compared using *String Similarity* and *SimSet* for set comparisons. For efficiency reasons the similarity computation was disburdened by removing extremely costly feature-measure combinations such as the comparison of all subclasses of two concepts.

4. These individual measures are all input to the similarity aggregation. Instead of applying linear aggregation functions, QOM applies a sigmoid function, which emphasizes high individual similarities and de-emphasizes low individual similarities.

[5]http://bibster.semanticweb.org

5. From the similarity values we derive the actual mappings. A threshold to discard spurious evidence of similarity is applied. Further mappings are assigned based on a greedy strategy that starts with the largest similarity values first.

6. Through several iteration rounds the quality of the results rises considerably. Eventually, the output returned is a mapping table representing the relation map_{O_1,O_2}.

The evaluation was very promising. Depending on the scenario QOM reaches high quality mapping levels very quickly. QOM is on a par with other good state-of-the-art algorithms concerning the quality of proposed mappings, while outperforming them with respect to efficiency — in terms of run-time complexity ($O(n)$ instead of $O(n^2)$) and in terms of the experiments we have performed (by a factor of 10 to 100).

5.22 KILT (INRIA Lorraine)

A short description of KILT, a maintenance tool for comparing knowledge base versions within the KASIMIR system (see [d'Aquin *et al.*, 2003]).

The KASIMIR system is a knowledge-based system aimed at helping the decision process when searching for an adequate treatment for patients ill with cancer. During an update (or a revision) of a KASIMIR knowledge base, the need for automatically comparing the old base KBold (before the update) and the new base KBnew (after the update) has appeared and is rather important for controlling the evolution of a knowledge base. A module comparing versions has to indicate what has been actually updated, and to check whether the modifications are in accordance with the intents of the knowledge engineer. This is the role of the module called KILT, that has been implemented and integrated into the PROTEGE knowledge editor. KILT enables to make a partitioning of the problems (i.e. a problem is described by a concept denoting a set of patients, and is possibly associated with a solution or a treatment), represented in KBold and/or KBnew in four parts:

1. The problems that appear in the two bases, with the same solutions;
2. The problems that appear in the two bases, with different solutions;
3. The obsolete problems, appearing in KBold but not in KBnew;
4. The new problems, appearing in KBnew but not in KBold.

The above partitioning is based on the use of the KASIMIR reasoner. For example, the new problems in category (4) can be found in the following way. Each problem PBnew of KBnew is classified in the hierarchy of KBold, which enables to check whether there is a problem PBold of KBold equivalent to PBnew, i.e. PBold subsumes and is subsumed by PBnew. If this is not the case, then PBnew is a new problem. The three other categories of problems (1), (2), and (3), can be detected and checked in a similar way. This shows that the implementation of KILT is rather simple, once the connection with the KASIMIR reasoner is done.

KILT is integrated in PROTEGE in the following way. During a session, KBold corresponds to the state of the knowledge base at the beginning of the session, and KBnew to its current state. Therefore, the KILT module enables to visualize the edition modifications, i.e. addition or removal of a problem, and association of another solution to an already known problem, at any time of the session. KILT makes comparisons at a semantic level: two concepts match when they have equivalent definitions, based on their attribute values and on the subsumption relation between classes. One main drawback is that it is assumed that the attributes –and their names– do not change from one knowledge base version to another.

Chapter 6

Conclusions

As as been demonstrated by the number of provided of use cases (§2) and developed systems (§5, alignments are useful in the semantic web. There have been a number of techniques on which they could be based, both for the local grouping of objects (§3) and the global computation (§4 of the alignment.

However, there is no common understanding of what to align, how to provide the results and what is important. But, it seems necessary in order to stimulate and integrate research to:

- be able to compare the approches;
- be able to combine them.

For that purpose, developing a common framework and benchmarking experimentation is a first answer to these two requirements. These are the topics of other deliverables (D2.2.1 and D2.2.2 respectively) on which the work is already ongoing.

Bibliography

[Aberer *et al.*, 2003a] Karl Aberer, Philippe Cudré-Mauroux, A. Datta, Z. Despotovic, Manfred Hauswirth, M. Punceva, and R. Schmidt. P-Grid: A self-organizing structured p2p system. *ACM SIGMOD Record*, 32(3), 2003.

[Aberer *et al.*, 2003b] Karl Aberer, Philippe Cudré-Mauroux, and Manfred Hauswirth. Start making sense: The chatty web approach for global semantic agreements. *Journal of Web Semantics*, 1(1), December 2003.

[Aberer *et al.*, 2004a] Karl Aberer, Tiziana Catarci, Philippe Cudré-Mauroux, T. Dillon, S. Grimm, Mohand-Saïd Hacid, A. Illarramendi, Mustafa Jarrar, V. Kashyap, M. Mecella, E. Mena, E. Neuhold, A. Ouksel, T. Risse, M. Scannapieco, F. Saltor, L. De Santis, Stefano Spaccapietra, Steffen Staab, Rudi Studer, and O. De Troyer. Emergent semantics systems. In Mokrane Bouzeghoub, Carole Goble, V. Kashyap, and Stefano Spaccapietra, editors, *proceeding of International Conference on Semantics of a Networked World*, LNCS, pages 14 – 44. Springer Verlag, 2004.

[Aberer *et al.*, 2004b] Karl Aberer, Philippe Cudré-Mauroux, M. Ouksel, Tiziana Catarci, Mohand-Saïd Hacid, A. Illarramendi, V. Kashyap, M. Mecella, E. Mena, E. Neuhold, O. De Troyer, T. Risse, M. Scannapieco, F. Saltor, L. de Santis, Stefano Spaccapietra, Steffen Staab, and Rudi Studer. Emergent semantics principles and issues. In *Proceedings of the 9th International Conference on Database Systems for Advanced Applications (DASFAA 2004)*, Jeju Island, Korea, 2004.

[Bach *et al.*, 2004] T. L. Bach, Rose Dieng-Kuntz, and Fabien Gandon. On ontology matching problems (for building a corporate semantic web in a multi-communities organization). In *Proc. of ICEIS 2004, Porto (PT)*, 2004.

[Barrasa *et al.*, 2003] Jesus Barrasa, Oscar Corcho, and Asunsión Gómez-Pérez. Fundfinder – a case study of database-to-ontology mapping. In *Proc. ISWC Semantic integration workshop, Sanibel Island (FL US)*, 2003.

[Barthélemy and Guénoche, 1992] Jean-Pierre Barthélemy and Alain Guénoche. *Trees and Proximity Representations*. John Wiley & Sons, Chichester, West Sussex, 1992.

[Barwise and Seligman, 1997] Jon Barwise and Jerry Seligman. *Information flow: the logic of distributed systems*, volume 44 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, Cambridge (UK), 1997.

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

[Beneventano *et al.*, 1998] D. Beneventano, S. Bergamaschi, S. Lodi, and C. Sartori. Consistency checking in complex object database schemata with integrity constraints. *IEEE Transactions on Knowledge and Data Engineering*, 10:576–598, 1998.

[Berenji and Khedkar, 1992] H. R. Berenji and P. Khedkar. Learning and tuning fuzzy logic controllers through reinforcements. *IEEE Trans. on Neural Networks*, 3(5):724–740, 1992.

[Bergamaschi *et al.*, 1998] S. Bergamaschi, D. Beneventano, S. Castano, and M. 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.

[Bergamaschi *et al.*, 1999] S. Bergamaschi, S. Castano, and M. Vincini. Semantic integration of semistructured and structured data sources. *SIGMOD Record*, 28(1):54–59, 1999.

[Berlin and Motro, 2001] J. Berlin and A. Motro. Autoplex: Automated discovery of content for virtual databases. In *Proceeding of CoopIS*, pages 108–122, 2001.

[Berlin and Motro, 2002] J. Berlin and A. Motro. Database schema matching using machine learning with feature selection. In *Proc. conference on advanced information system engineering (CAiSE)*, 2002.

[Berre, 2001] Daniel Le Berre. Jsat: The java satisfiability library, 2001.

[Bisson, 1992] Gilles Bisson. Learning in FOL with similarity measure. In *Proc. 10th American Association for Artificial Intelligence conference, San-Jose (CA US)*, pages 82–87, 1992.

[Bouquet and Serafini, 2003] Paolo Bouquet and Luciano Serafini. On the difference between bridge rules and lifting axioms. Technical Report DIT-03-004, University of Trento (IT), January 2003.

[Bouquet *et al.*, 2003] Paolo Bouquet, Luciano Serafini, and Stefano Zanobini. Semantic coordination: A new approach and an application. In Dieter Fensel, Katia Sycara, and John Mylopoulos, editors, *Proc 2nd ISWC*, volume 2870 of *Lecture Notes in Computer Science*, pages 130–145, Sanibel Island (FL, USA), October 2003. Springer Verlag.

[Broekstra *et al.*, 2004] Jeen Broekstra, Marc Ehrig, Peter Haase, Frank van Harmelen, Maarten Menken, Peter Mika, Bjoern Schnizler, and Ronny Siebes. Bibster - a semantics-based bibliographic peer-to-peer system. In *Proceedings of the SemPGrid 04 Workshop*, New York, May 2004.

[Castano *et al.*, 2000] S. Castano, V. De Antonellis, and S. De Capitani di Vimercati. Global viewing of heterogeneous data sources. *IEEE Transactions on Knowledge and Data Engineering*, 2000.

[Cohen *et al.*, 2003] William Cohen, Pradeep Ravikumar, and Stephen Fienberg. A comparison of string metrics for matching names and records. In *Proc. KDD-2003 Workshop on Data Cleaning and Object Consolidation*, 2003.

[Corby and Faron, 2002] Olivier Corby and Catherine Faron. Corese: A corporate semantic web engine. In *Proc. WWW International Workshop on Real World RDF and Semantic Web Applications*, Hawai (HA US), May 2002.

[Corby *et al.*, 2000] Olivier Corby, Rose Dieng, and Cédric Hébert. A conceptual graph model for w3c resource description framework. In *Proc. 8th International Conference on Conceptual Structures Logical, Linguistic, and Computational Issues*, Darmstadt (DE), August 2000. Springer-Verlag.

[Corby *et al.*, 2004] Olivier Corby, Rose Dieng-Kuntz, and Catherine Faron-Zucker. Querying the semantic web with the corese search engine. In *Proc. 15th ECAI/PAIS*, Valencia (ES), August 2004. IOS Press.

[d'Aquin *et al.*, 2003] Mathieu d'Aquin, C. Bouthier, S. Brachais, Jean Lieber, and Amedeo Napoli. Knowledge editing and maintenance tools for a semantic portal in oncology. Rapport de recherche A03-R-162, LORIA, 2003.

[De Bo *et al.*, 2004a] J. De Bo, P. Spyns, and R. Meersman. Assisting ontology integration with existing thesauri. In *Submitted to ODBASE04*, 2004.

[De Bo *et al.*, 2004b] J. De Bo, P. Spyns, and R. Meersman. Towards a methodology for semi-automatic ontology aligning and merging. Technical Report 02, Vrije Universiteit Brussel, STARLab, 2004.

[Dieng and Hug, 1998a] Rose Dieng and Stefan Hug. Comparison of "personal ontologies" represented through conceptual graphs. In *Proc. 13th ECAI, Brighton (UK)*, pages 341–345, 1998.

[Dieng and Hug, 1998b] Rose Dieng and Stefan Hug. Multikat, a tool for comparing knowledge from multiple experts. In *Conceptual Structures: Theory, Tools and Applications, Proc. of the 6th Int. Conference on Conceptual Structures (ICCS'98)*, Montpellier (FR), August 10-12 1998. Springer-Verlag, LNAI 1453.

[Do and Rahm, 2002] Hong-Hai Do and Erhard Rahm. Coma – a system for flexible combination of schema matching approaches. In *Proc. VLDB*, pages 610–621, 2002.

[Do *et al.*, 2002] Hong-Hai Do, Sergey Melnik, and Erhard Rahm. Comparison of schema matching evaluations. In *Proc. GI-Workshop "Web and Databases", Erfurt (DE)*, 2002. http://dol.uni-leipzig.de/pub/2002-28.

[Doan *et al.*, 2001] An-Hai Doan, P. Domingos, and A. Halevy. Reconciling schemas of disparate data sources: A machine-learning approach. In *Proceeding of SIGMOD*, 2001.

[Doan *et al.*, 2003] An-Hai Doan, J. Madhavan, R. Dhamankar, P. Domingos, and A. Halevy. Learning to map ontologies on the semantic web. *VLDB journal*, 2003.

[Doan *et al.*, 2004] 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.

[Doan, 2002] An-Hai Doan. *Learning to map between structured representations of data*. PhD thesis, University of Washington, Seattle (WA US), 2002. http://www.kdnuggets.com/news/2004/n01/23i.html.

[Ehrig and Staab, 2004] Marc Ehrig and Steffen Staab. QOM - quick ontology mapping. In *Proc. 3rd ISWC, Hiroshima (JP)*, November 2004. to appear.

[Ehrig and Sure, 2004] Marc Ehrig and York Sure. Ontology mapping – an integrated approach. In Christoph Bussler, John Davis, Dieter Fensel, and Rudi Studer, editors, *Proc. 1st ESWS, Hersounisous (GR)*, volume 3053 of *Lecture Notes in Computer Science*, pages 76–91. Springer Verlag, MAY 2004.

[Euzenat and Valtchev, 2003] Jérôme Euzenat and Petko Valtchev. An integrative proximity measure for ontology alignment. In *Proc. ISWC-2003 workshop on semantic information integration, Sanibel Island (FL US)*, pages 33–38, 2003.

[Euzenat and Valtchev, 2004] Jérôme Euzenat and Petko Valtchev. Similarity-based ontology alignment in OWL-lite. In *Proc. 15th ECAI*, Valencia (ES), 2004.

[Euzenat, 1994] Jérôme Euzenat. Brief overview of T-tree: the Tropes taxonomy building tool. In *Proc. 4th ASIS SIG/CR workshop on classification research, Columbus (OH US)*, pages 69–87, 1994. ftp://ftp.inrialpes.fr/pub/sherpa/publications/euzenat93c.ps.gz.

[Euzenat, 2004] Jérôme Euzenat. An api for ontology alignment. In *Proc. 3rd international semantic web conference, Hiroshima (JP)*, 2004.

[Fung and Lo, 1998] P. Fung and Y.Y. Lo. An ir approach for translating new words from unparallel, comparable texts. In *Proc. of 36th Annual ACL Conference*, pages 414–420, Montreal, Canada, 1998.

[Gal *et al.*, 2004] Avigdor Gal, Ateret Anaby-Tavor, Alberto Trombetta, and Danilo Montesi. A framework for modeling and evaluating automatic semantic reconciliation. *VLDB Journal*, 2004. to appear.

[Gandon, 2002] Fabien Gandon. *Distributed Artificial Intelligence and Knowledge Management: ontologies and multi-agent systems for a corporate semantic web*. Scientific philosopher doctorate thesis in informatics, INRIA and University of Nice - Sophia Antipolis, November 2002.

[Ganter and Wille, 1999] Bernhard Ganter and Rudolf Wille. *Formal concept analysis: mathematical foundations*. Springer Verlag, Berlin (DE), 1999.

[Giunchiglia and Shvaiko, 2003] Fausto Giunchiglia and Pavel Shvaiko. Semantic matching. In *Proc. IJCAI 2003 Workshop on ontologies and distributed systems, Acapulco (MX)*, pages 139–146, 2003.

[Giunchiglia and Shvaiko, 2004] Fausto Giunchiglia and Pavel Shvaiko. Semantic matching. *The Knowledge Engineering Review*, 18(3):265–280, 2004.

[Giunchiglia *et al.*, 2004] Fausto Giunchiglia, Pavel Shvaiko, and Michael Yatskevich. S-Match: an algorithm and an implementation of semantic matching. In *Proceedings of ESWS 2004, Heraklion (GR)*, pages 61–75, 2004.

[Hajek, 1998] P. Hajek. *The metamathematics of fuzzy logic*. Kluwer, 1998.

[Hayashi *et al.*, 1992] Y. Hayashi, E. Czogala, and J. J. Bukley. Fuzzy neural controller. In *Proc. IEEE Int. Conf. Fuzzy Syst*, pages 197–202, San Diago, 1992.

[Horikawa *et al.*, 1992] S. Horikawa, T. Furuhashi, and Y. Uchikawa. On fuzzy modeling using fuzzy neural networks with the back-propagation algorithm. *IEEE Trans. Neural Networks*, pages 801–806, 1992.

[Jacquemin and Royaute, 1994] Christian Jacquemin and Jean Royaute. Retrieving terms and their variants in a lexicalised unification-based framework. In *Proc. of 13th Annual International ACM-SIGIR Conference in Research and Development in Information Retrieval*, pages 132–141, Dublin, 1994.

[Jacquemin, 1996] Christian Jacquemin. What is the tree that we see through the window: A linguistic approach to windowing and term variation. *Information Processing and Management*, 32(4):445–458, 1996.

[Jacquemin, 1997] Christian Jacquemin. Recognition and acquisition: Two inter-related activities in corpus-based term extraction. *Terminology*, 4(2):245 ff., 1997.

[Kalfoglou and Schorlemmer, 2003a] Yannis Kalfoglou and Marco Schorlemmer. If-map: an ontology mapping method based on information flow theory. *Journal of data semantics*, 1:98–127, 2003.

[Kalfoglou and Schorlemmer, 2003b] Yannis Kalfoglou and Marco Schorlemmer. Ontology mapping: the state of the art. *The Knowledge Engineering Review*, 18(1):1–31, 2003.

[Keller and Hunt, 1992] J. M. Keller and D. J. Hunt. Evidence aggregation networks for fuzzy logic interface. *IEEE Trans. Neural Networks*, pages 751–769, 1992.

[Larson *et al.*, 1989] J. A. Larson, S. B. Navathe, and R. Elmasri. A theory of attributed equivalence in databases with application to schema integration. *IEEE Trans. Softw. Eng.*, 15(4):449–463, 1989.

[Levenshtein, 1966] I. V. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Cybernetics and Control Theory*, 1966.

[Li and Clifton, 1994a] Wen-Syan Li and Chris Clifton. Semantic integration in heterogeneous databases using neural networks. In *Proceedings of the 20th International Conference on Very Large Data Bases*, pages 1–12. Morgan Kaufmann Publishers Inc., 1994.

[Li and Clifton, 1994b] W.S. Li and C. Clifton. Semantic integration in heterogeneous databases using neural networks. In *Proc. 20th VLDB, Santiago (CH)*, pages 1–12, 1994.

[Li and Clifton, 2000] Wen-Syan Li and Chris Clifton. Semint: a tool for identifying attribute correspondences in heterogeneous databases using neural networks. *Data Knowl. Eng.*, 33(1):49–84, 2000.

[Lin and Lu, 1995] C. T. Lin and Y. C. Lu. A neural fuzzy system with linguistic teaching signals. *IEEE Trans. Fuzzy Syst*, 3:169–189, 1995.

[Lovins, 1968] J. B. Lovins. Development of a stemming algorithm. *Mechanical Translation and Computational Linguistics*, 11:22–31, 1968.

[Mädche and Staab, 2002] Alexander Mädche and Steffen Staab. Measuring similarity between ontologies. In *Proc. Of the 13th Int. Conference on Knowledge Engineering and Management (EKAW-2002)*, Siguenza, Spain, October 2002. Springer-Verlag.

[Mädche and Zacharias, 2002] Alexander Mädche and Valentin Zacharias. Clustering ontology-based metadata in the semantic web. In *Proc. 13th ECML and 6th PKDD, Helsinki (FI)*, 2002.

[Madhavan *et al.*, 2001a] Jayant Madhavan, Philip Bernstein, and Erhard Rahm. Generic schema matching using Cupid. In *Proc. 27th VLDB, Roma (IT)*, pages 48–58, 2001. http://research.microsoft.com/ philbe/CupidVLDB01.pdf.

[Madhavan *et al.*, 2001b] Jayant Madhavan, Philip A. Bernstein, and Erhard Rahm. Generic schema matching with cupid. In *Proceedings of the 27th International Conference on Very Large Data Bases*, pages 49–58. Morgan Kaufmann Publishers Inc., 2001.

[Magnini *et al.*, 2003] B. Magnini, Luciano Serafini, and M. Speranza. Making explicit the semantics hidden in schema models. In *Proceedings of ISWC workshop on Human Language Technology for the Semantic Web and Web Services*, 2003.

[Mariño *et al.*, 1990] Olga Mariño, cois Rechenmann, Fran and Patrice Uvietta. Multiple perspectives and classification mechanism in object-oriented representation. In *9th European Conference on Artificial Intelligence*, pages 425–430, Stockholm, Suède, August 1990.

[Maynard and Ananiadou, 1999] D.G. Maynard and S. Ananiadou. Term extraction using a similarity-based approach. In *Recent Advances in Computational Terminology*. John Benjamins, 1999.

[Maynard, 1999] Diana Maynard. *Term Recognition Using Combined Knowledge Sources*. PhD thesis, Department of Computing and Mathematics, Manchester Metropolitan University, UK, 1999.

[McGuinness *et al.*, 2000] D.L. McGuinness, R. Fikes, J. Rice, and S. Wilder. An environment for merging and testing large ontologies. In *Proceeding of KR*, pages 483–493, 2000.

[MDC, 1999] Open information model, version 1.0, 1999. http://mdcinfo/oim/oim10.html.

[Melnik *et al.*, 2002] Sergey Melnik, Hector Garcia-Molina, and Erhard Rahm. Similarity flooding: a versatile graph matching algorithm. In *Proc. 18th International Conference on Data Engineering (ICDE), San Jose (CA US)*, 2002.

[Melnik *et al.*, 2003] Sergey Melnik, Erhard Rahm, and Philip Bernstein. Rondo: A programming platform for model management. In *Proc. ACM SIGMOD, San Diego (CA US)*, 2003.

[Miller, 1995] A.G. Miller. Wordnet: A lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

[Mitra and Wiederhold, 2002] P. Mitra and Gio Wiederhold. Resolving terminological heterogeneity in ontologies. In *Workshop on Ontologies and Semantic Interoperability at the 15th European Conference on Artificial Intelligence (ECAI)*, 2002.

[Navathe and Buneman, 1986] S. Navathe and P. Buneman. Integrating user views in database design. *Computer*, 19(1):50–62, January 1986.

[Ngai *et al.*, 2002] Grace Ngai, Marine Carpuat, and Pascale Fung. Identifying concepts across languages: A first step towards a corpus-based approach to automatic ontology alignment. In *Proc. of COLING 2002*, Taipei, Taiwan, 2002.

[Niles and Pease, 2001] I. Niles and A. Pease. Towards a standard upper ontology. In Chris Welty and Barry Smith, editors, *In Proceedings of the 2nd International Conference on Formal Ontology in Information Systems (FOIS-2001)*, 2001.

[Noy and Musen, 2001] Natasha Noy and Mark Musen. Anchor-PROMPT: Using non-local context for semantic matching. In *Proc. IJCAI 2001 workshop on ontology and information sharing, Seattle (WA US)*, pages 63–70, 2001. http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-47/.

[Palopoli *et al.*, 2000] L. Palopoli, G. Terracina, and D. Ursino. The system dike: Towards the semi-automatic synthesis of cooperative information systems and data warehouses. In *Proceeding of ADBIS-DASFAA*, pages 108–117, 2000.

[Pao, 1989] Y.H. Pao. *Adaptive Pattern Recognition and Neural networks*. Addison-Wesley., 1989.

[Papadimitriou and Steiglitz, 1998] Christos Papadimitriou and Kenneth Steiglitz. *Combinatorial Optimization : Algorithms and Complexity*. Prentice-Hall, 1998.

[Parent and Spaccapietra, 2000] Christine Parent and Stefano Spaccapietra. *Database integration: the key to data interoperability*. The MIT Press, Cambridge (MA US), 2000.

[Patel *et al.*, 2003] C. Patel, K. Supekar, and Y. Lee. Ontogenie: Extracting ontology instances from WWW. In *Human Language Technology for the Semantic Web and Web Services, ISWC'03*, Sanibel Island, Florida, 2003.

[Pedersen *et al.*, 2004] T. Pedersen, S. Patwardhan, and J. Michelizzi. Wordnet::similarity - measuring the relatedness of concepts. In *Appears in the Proceedings of the Nineteenth National Conference on Artificial Intelligence (AAAI-04)*, 2004.

[Porter, 1980] M.F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

[Rahm and Bernstein, 2001a] Erhard Rahm and Philip Bernstein. A survey of approaches to automatic schema matching. *VLDB Journal*, 10(4):334–350, 2001.

[Rahm and Bernstein, 2001b] Erhard Rahm and Philip A. Bernstein. A survey of approaches to automatic schema matching. *The VLDB Journal*, 10(4):334–350, 2001.

[Resnik, 1995] Phillip Resnik. Using information content to evaluate semantic similarity in a taxonomy. In *Proc. 14th IJCAI*, pages 448–453, Montréal, Canada, 1995.

[Sager, 1990] J. C. Sager. *A Practical Course in Terminology Processing*. John Benjamins, 1990.

[Sheth *et al.*, 1988] Amit P. Sheth, James A. Larson, Aloysius Cornelio, and Shamkant B. Navathe. A tool for integrating conceptual schemas and user views. In *Proceedings of the Fourth International Conference on Data Engineering*, pages 176–183. IEEE Computer Society, 1988.

[Shvaiko, 2004] Pavel Shvaiko. Iterative schema-based semantic matching. Technical Report DIT-04-020, University of Trento (IT), 2004.

[Silva and Rocha, 2003] Nuno Silva and J. Rocha. An ontology mapping framework for the semantic web. In *Proc. 6th International Conference on Business Information Systems*, Colorado Springs, USA, 2003.

[Sowa, 1984] John Sowa. *Conceptual Structures: Information Processing In Mind and Machine*. Addison-Wesley, 1984.

[Staab and Studer, 2004] Steffen Staab and Rudi Studer. *Handbook of ontologies*. International handbooks on information systems. Springer Verlag, Berlin (DE), 2004.

[Stumme and Mädche, 2001] Gerd Stumme and Alexander Mädche. FCA-merge: bottom-up merging of ontologies. In *Proc. 17th IJCAI, Seattle (WA US)*, pages 225–230, 2001.

[Su and Gulla, 2003] X. Su and J. A. Gulla. Semantic enrichment for ontology mapping. In *Proceedings of the 9th International Conference on Applications of Natural Language to Information Systems*, Manchester, UK, 2003.

[Tverski, 1977] Amos Tverski. Features of similarity. *Psychological review*, 84:327–352, 1977.

[Uehara and Fujise, 1992] K. Uehara and M. Fujise. Multistage fuzzy inference formulated as linguistic-truth-value propagation and its learning algorithm based on back-propagating error information. *IEEE Transactions on Fuzzy Systems*, 3, 1992.

[Valtchev and Euzenat, 1997] Petko Valtchev and Jérôme Euzenat. Dissimilarity measure for collections of objects and values. In P. Coen X. Liu and M. Berthold, editors, *Proc. 2nd Symposium on Intelligent Data Analysis.*, volume 1280, pages 259–272, 1997.

[Valtchev, 1999] Petko Valtchev. *Construction automatique de taxonomies pour l'aide à la représentation de connaissances par objets*. Thèse d'informatique, Université Grenoble 1, 1999.

[Visser *et al.*, 2002] Ubbo Visser, Thomas Vögele, and Christoph Schlieder. Spatio-terminological information retrieval using the buster system. In *Proceedings of the EnviroInfo*, pages 93–100, Wien (AT), 2002.

[Vögele *et al.*, 2003] Thomas Vögele, Sebastian Hübner, and Gerhard Schuster. Buster - an information broker for the semantic web. *Künstliche Intelligenz*, 3:31–34, July 2003.

[Yager, 1992] Ronald Yager. OWA neurons: a new class of fuzzy neurons. In *Proc. Int. Joint Conf. Neural Networks*, volume 1, pages 226–231, 1992.

Related deliverables

A number of Knowledge web deliverable are clearly related to this one:

Project	Number	**Title** and relationship
KW	D2.1.1	**Survey of scalability techniques for reasoning with ontologies** study the use of modularity for the purpose of scalability. The composition of modules can raise heterogeneity problems that are naturally solved by using alignment results. The techniques for this are found in the present deliverable.
KW	D2.2.1	**Specification of a common framework for characterizing alignment** proposes a framework for defining what alignment are. It circumscribe the action of the techniques presented here from the inside.
KW	D2.4.2	**Heterogeneity in the context of Semantic web services - use cases and usage scenarios** provides a more detailed description of use cases in the context of semantic web services.
SDK	D19.2	**WSMO in Knowledge web** describes the involvement work package in the wider WSMO context.
SDK	D13.3	**Mediation** describes the approach to mediation in the Web Service Execution Environment. This is a detailled test case for the use of alignment techniques in the