Case-based recommendation of matching tools and techniques

Jérôme Euzenat, Marc Ehrig, Anja Jentzsch, Malgorzata Mochol, Pavel Shvaiko

D1.2.2.2.1: Case-based recommendation of matching tools and techniques

Coordinator: Jérôme Euzenat (INRIA)
Marc Ehrig (U. Karlsruhe), Anja Jentzsch (FUBerlin), Malgorzata Mochol (FUBerlin), Pavel Shvaiko (U. Trento)

Abstract.

Choosing a matching tool adapted to a particular application can be very difficult. This document analyses the choice criteria from the application viewpoint and their fulfilment by the candidate matching systems. Different methods (paper analysis, questionnaire, empirical evaluation and decision making techniques) are used for assessing them. We evaluate how these criteria can be combined and how they can help particular users to decide in favour or against some matching system.
Keyword list: ontology matching, ontology alignment, ontology mapping, evaluation, benchmarking, contest, performance measure.

Document Identifier	KWEB/2004/D1.2.2.2.1/v1.1
Project	IST-2004-507482 (KWEB)
Version	v1.1
Date	March 1, 2007
State	final
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
0.1	13.02.2006	Jérôme Euzenat	creation
0.2	20.06.2006	Jérôme Euzenat	input from Budva discussion
0.3	06.10.2006	Jérôme Euzenat	integrated summer work from PS and AM.
0.4	25.10.2006	Jérôme Euzenat	Further improved content.
0.5	21.11.2006	Malgorzata Mochol	changes in the structure; changes in Chapter 3, 6 and integration of comments (regarding Chapter 3) from PS
0.6	29.11.2006	Jérôme Euzenat	filled missing parts (all chapters)
0.7	18.12.2006	Malgorzata Mochol	filled missing sections (§5.2 & 7.2)
0.8	21.12.2006	Jérôme Euzenat	Pavel's comments + filled missing sections (§4.2, 5.3, Exec summary and conclusion)
0.9	07.01.2007	Jérôme Euzenat	Improved sections (§4.2, 5.3, 7.1)
1.0	31.01.2007	Jérôme Euzenat, Malgorzata Mochol	Taken QC and Raúl Garcia Castro comments into account
1.1	1.03.2007	Malgorzata Mochol	Taken all comments into account

Executive summary

Choosing a matching tool adapted to a particular application can be very difficult. Even when the application needs are very precise, there are many criteria that can be used for choosing an adequate matcher and all criteria cannot be assessed in the same way. It is also difficult to obtain all the information about each matcher (the information is most often fragmentary).

This document analyses the choice criteria from the application viewpoint and their fulfilment by the candidate matching systems. It is not based on abstract categories of matchers and potential application, we consider real application as test cases and existing matchers as the possible choices. However, we use a systematic path which leads us to offer categories in which these fall naturally.

In order to help identifying matching solutions to particular applications, we identify basic application requirements based on a very general analysis of the application needs. This allow to recognise that there is a large variety of needs and that they are not the same for all applications.

Then we identify in details the relevant characteristics of matchers. Grouped in 6 dimensions (input, approach, usage, output, documentation and costs) we found 100 characteristics to be accounted for. We also present techniques for assessing the value of some of these characteristics through evaluation of matching systems.

This work permits to study the profile of actual matchers with regard to these characteristics. This has been obtained through different methods. We provide a global analysis, based on litterature survey of 48 matching systems. We use the result of evaluation of more precisely characterising 17 systems accepting OWL as input with regard to the primitive used in the ontologies. Finally, we have a finer analysis of 8 system through answering 37 questions related to the identified characteristics.

Given this data we provide two methods for finding which matcher is usable for which application. The first naive method is based on weighted aggregation of the characteristics depending on the expressed needs of applications. The more elaborate Analytic hierarchy process method (AHP) is a decision support method that take advantage of direct relative preference on criteria. We describe a tool able to implement this process for making a decision.

This work is applied to the general categories of applications as described before as well as more specifically to the Knowledge web Human resource use case in order to demonstrate the choice process.

In conclusion, we provide a brief methodological guide on how to take advantage of the various resources provided in this deliverable.

Contents

Chapter 1

Introduction

In the ontology matching field, there is no overarching matching algorithm for ontologies that is capable of serving all (heterogeneous) ontological sources. Most of the research in this area proposes new approaches based on different principles and relies on various features. These new approaches only solve small parts of "global" problems in the matching field or fill some open matching gaps [Fürst and Trichet, 2005]. Therefore in general, when implementing an application using a matching approach, the corresponding algorithm is typically built from scratch and only small marginal attempt to reuse existing methods is made.

Despite an impressive number of research initiatives in the matching field, current matching approaches still feature major limitations. For example, the majority of existing approaches to ontology matching are (implicitly) restricted to processing particular classes of ontologies and thus they are unable to guarantee a predictable quality of results on arbitrary inputs. What is required are appropriate ontology matching techniques capable of coping with different levels of detail in concept descriptions [Castano *et al.*, 2004].

Hence, finding the most suited matching system for a particular application is a difficult task because there are so many different systems and so many different application characteristics. This double variability can be seen positively or negatively depending on one's capacity to take advantage of it.

The goal of this deliverable is to provide elements for choosing a matching system depending on the application to be developed. Most of these elements are of generic nature: we describe methodologies for evaluating the adequacy between matchers and applications. We also apply these methodologies to actual matching systems and a use case taken from Knowledge web use cases [Léger *et al.*, 2005].

The methodology followed in this deliverable is summarised by Figure 1.1. Our goal is to define matching system profiles (Chapter 5) and application profiles (Chapter 2) and to base recommendations on the matching of these two types of profiles (Chapter 7). For obtaining the matcher profiles we ground our work on:

- identifying what are the characteristics of these matchers as well as the needs for the applications (Chapter 3);
- designing strategies for obtaining the actual profiles, for instance through evaluation (Chapter 4);
- extracting matcher profiles through either literature review, developer survey or evaluation results (Chapter 5).

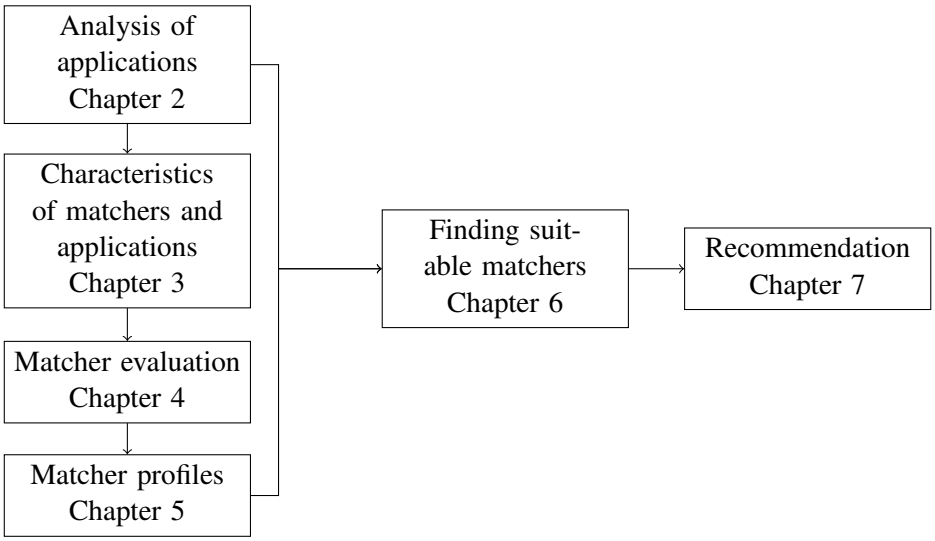


Figure 1.1: The process adopted by this deliverable (as well as set of chapters).

The matcher profiles can be compared to individual and generic application profiles through multi-criteria decision methods (Chapter 6). This methodology is applied to use cases introduced in Deliverable 1.1.4 and more specifically to use case 1 (Chapter 7).

Chapter 2

Analysis of applications

We provide here a summary analysis of the requirements of applications with regard to ontology matching. We first consider a typology of applications (§2.1), examine the requirements that these applications may raise (§2.2) and apply these results to the relevant Knowledge web use cases (§2.3).

2.1 Types of applications

In previous deliverables [Euzenat *et al.*, 2004a] and [Euzenat and Shvaiko, 2007], several classes of applications are considered (they are thoroughly described in the above references, we only summarized them here). They are the following:

Ontology evolution uses matching for finding the changes that have occured between two ontology versions;

Schema integration uses matching for integrating the schemas of different databases under a single view;

Catalog integration uses matching for offering a integrated access to online catalogs;

Data integration uses matching for integrating the content of different databases under a single database;

P2P information sharing uses matching for finding the relations of ontologies used by different peers;

Web service composition uses matching between ontologies describing service interfaces in order to compose web services by connecting their interfaces;

Multi agent communication use matching for finding the relations between the ontologies used by two agents and translating the messages they exchange;

Context matching in ambient computing uses matching of application needs and context information when application and devices have been developed independently and use different ontologies;

Query answering uses ontology matching for translating user queries about the web;

Semantic web browsing uses matching for dynamically (while browsing) annotating web pages with partially overlapping ontologies.

These kinds of applications have been analysed in order to establish their requirements with regard to matching systems.

2.2 Application requirements

This analysis leads to different requirements for different applications. We summarise in Table 2.1 what we have found to be the most important requirements to matching solutions in the considered applications. These requirements concern:

- the type of available input a matching system can rely on, such as schema or instance information. There are cases when data instances are not available, for instance due to security reasons [Clifton *et al.*, 1997] or when there are no instances given beforehand. Therefore, these applications require only a matching solution able to work without instances (here schema-based method).

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

- the use of the matching result as described above. In particular, how the identified alignment is going to be processed, e.g., by merging the data or conceptual models under consideration or by translating data instances among them.

The above requirements together with applications from which they come are summarised in Table 2.1. Ontology evolution is typically used at design time for transforming an existing ontology which may have instances available. It requires an accurate (i.e., correct and complete) matching, but can be performed with the help of users. Schema, Catalogue and Data integration are also performed off line but can be used for different purpose: translating data from one base to another, merging two databases or generating a mediator that will be used for answering queries. They also will be supervised by a human user and can provide instances. Other applications are rather performed at runtime. Some of these like P2P information sharing, query answering and semantic web browsing are achieved in presence of users who can support the process. They are also less demanding in terms of correctness and completeness because the user will directly sort out the results. On the other hand, web-service composition, multiagent communication and context matching in ambient computing require matching to be performed automatically without assistance of a human being. Since, the systems will use the result of matching for performing some action (mediating or translating data) which will be feed in other processes, correctness is required. Moreover, usually these applications do not have instance data available.

Some of these hard requirements can be derived into comparative (or non-functional) requirements such as *speed*, resource consumption (in particular memory requirements), degree of correctness or completeness. They are useful for comparing solutions on a scale instead of absolutely. Moreover, they allow to trade a requirement, e.g., completeness, for another more important one, e.g., speed.

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

- ontology engineering requires the ability to *transform* relevant ontologies or some parts of these ontologies into an ontology focusing on a domain of interest being modeled or to generate a set of bridge axioms that will help in identifying corresponding concepts (the transformations apply at the ontological level);

Application	instances	run time	automatic	correct	complete	operation
Ontology evolution	√			√	√	transformation
Schema integration	√			√	√	merging
Catalog integration	√			√	√	data translation
Data integration	√			√	√	query mediation
P2P information sharing		√				query mediation
Web service composition		√	√	√		data mediation
Multi agent communication		√	√	√	√	data translation
Context matching in ambient computing		√	√	√		data translation
Query answering	√	√				query reformulation
Semantic web browsing	√	√		√		navigation

Table 2.1: Summary of applications requirements.

- schema integration requires the ability to *merge* the schemas under consideration into a single schema (the transformations apply at the ontological level);
- data integration requires the ability to *translate data* instances residing in multiple local schemas according to a global schema definition in order to enable query mediation over the global schema;
- peer-to-peer systems and more generally query mediation systems require bidirectional *mediators* able to translate queries (ontological level) and translate back answers (data level);
- agent communication requires *translators* for messages sent from one agent to another, which apply at the data level; similarly, semantic web services require one-way data translations for composing services.

These requirements are reported in the "operation" column of Table 2.1.

2.3 Application to use cases

The use cases that have been provided in [Léger *et al.*, 2005], are recalled in Table 2.2 and classified with regard to the kind of application they are with regard to their ontology matching needs. This classification reflects only the main category of needs for matching. Some of these application have in reality several heterogeneity problems that can be solved differently and most of them cannot be reduced to ontology matching and are far wider.

In the present deliverable, we will more closely consider the first use case in Recruitements and job finding.

2.4 Summary

We have now a first idea of the application needs in order to define more finely the characteristics of matchers in the next chapter. These are generic requirements that apply to a whole class of applications and must be refined into specific requirements applying to a particular application.

Use case	Company	Matching	Type
Recruitments	WoldwideJobs	$\surd$	Data integration
B2C marketplace for tourism	France Telecom	$\surd$	Query answering
News aggregation	Neofonie	$\surd$	Semantic web browsing
Product lifecycle management	Semtation		
Real estate management	Trenitalia	$\surd$	P2P information sharing
Integrated access to biological data	Robotiker	$\surd$	Data integration
Geoscience project memory	IFP		
Hospital information system	L&C Global		

Table 2.2: Identified Knowledge web use cases and their correspondences in the classification of Table 2.1

Specific requirements for applications will be more thoroughly defined in Chapter 3 by characterizing matcher characteristics.

These generic requirements are used in Chapter 3 to be a basic set of requirements, in Chapter 4 for designing evaluation procedures related to applications as well as in Chapter 5 to establish matcher profiles.

Chapter 3

Characteristics of matching approaches (and applications)

Having a definition of the problem to be solved and the particular requirements regarding the final application, one must decide which matching algorithms are to be applied to satisfy these specification and to obtain the desired output. This depends on the characteristics of the applications (about input, output, process, etc.) and those of the available matchers. Possible attributes, that could have an impact on the selection of an adequate matching approach, must be defined in order to resolve this issue. Accounting for the empirical findings of different case studies in ontology engineering [Mochol and Paslaru Bontas Simperl, 2006; Paslaru Bontas and Mochol, 2005; Paslaru Bontas *et al.*, 2005], and regarding the requirements (cf. Chapter 2) collected during the development of different Semantic Web application scenarios [Bizer *et al.*, 2005; Garbers *et al.*, 2006][1], as well as during the intensive collaborations with ontology and software engineers, six groups of factors, called dimensions, have been defined as relevant for the matching selection process.

These dimensions are the main aspects that must be taken into account during the examination of the suitability of a single matching approach for the solving of a given problem: (i) *input characteristics* that takes into account the ontologies to be matched; (ii) *approach characteristics* describes the matching algorithms themselves; (iii) *output characteristics* defines the desired result of the matching execution; (iv) *usage characteristics* takes into account the different situations where the approaches have been used; (v) *documentation characteristics* points out the existence and type of the documentation; and (vi) *cost characteristics* addresses the costs which have to be paid for the usage of the algorithm.

The dimensions constitute the superficial collection for matcher attributes: they are the first level of the *multilevel characteristics for matching approaches*. The multilevel characteristics is organized in the form of a taxonomy where dimensions are defined by sets of factors and these are described by the attributes. These characteristics can be illustrated as a taxonomy (cf. Figure 3.1) where the child nodes describe and represent the parent nodes' properties [Lozano Tello and Gomez Perez, 2004].

In the following sections we briefly describe some of the factors of each dimension and state others in the form of tables (cf. Table 3.1, 3.2, 3.3, 3.4, 3.5, 3.6). The content of the tables are arguable and could have been presented in another way. However, these tables are the basis for the

[1]Projects: (i) Wissensnetze,[2], (ii) Reisewissen, [3], (iii) SWPatho, [4], (iv) Knowledge Web [5]

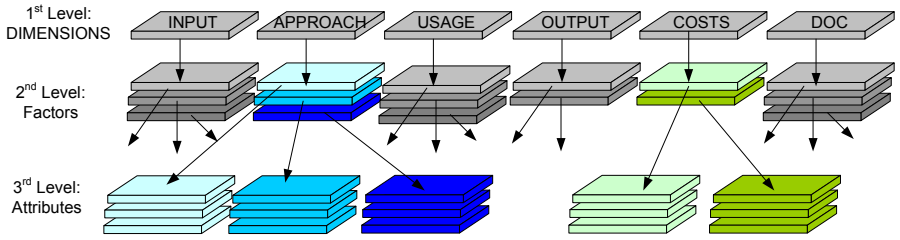


Figure 3.1: Multilevel characteristics with dimensions, factors and attributes

questionnaire that is used in §6.3. The questionnaire offered more details about what was expected than these table content. We did not tried to improve a posteriori the tables, but rather presented them as such. The questionnaires may be improved in the future.

3.1 Input characteristics

The first step towards the analysis of the matching characteristics is the examination of the matching input. In our opinion, the attributes that describe the input are the most important and relevant criteria that play a crucial role in the selection of the appropriate algorithm. Despite the relatively large number of promising matching approaches their limitations with respect to certain ontology characteristics have often been emphasized in recent literature [Giuchiglia and Shvaiko, 2004; Madhavan *et al.*, 2001a; Melnik *et al.*, 2002a; Shvaiko, 2004b; Shvaiko and Euzenat, 2005a]. The *input characteristic* dimension describes not only the heterogeneity of the sources that are to be matched, e.g. *size* (some matchers perform well on relatively small inputs), *natural language* used for the definition of concepts (some algorithms require certain natural language) and *input structure* (some matchers do not perform well on heterogeneous structures [Giuchiglia and Shvaiko, 2004]), but also takes into account *external sources*, which a matching algorithm can use for its execution (cf. Table 3.1).

Attribute	**Description**
Factor: Input Size (algorithm capable of handling:)	
number of ontologies	number of different ontologies to be matched (two or more)
size of input	number of ontological primitives (concepts, properties, axioms) to be matched: *small* (up to 100 primitives), *middle* (101 - 500 primitives), *large* (501 - 1,000 primitives), *extra large* (over 1,000 primitives)
size of instances	number of instances to be matched: *no instances*, *small* (up to 500 instances), *medium* (501 - 1,000 instances), *large* (1,001 - 10,000 instances), *extra large* (over 10,000 instances)
number of concepts	number of concepts to be matched: *small* (up to 100 concepts), *medium* (100 - 500 concepts), *large* (500 - 1,000 concepts), *extra large* (over 1.000 concepts)
number of relations	number of relations to be matched: *small* (up to 30 relations), *medium* (31 - 100 relations), *large* (100 - 1,000 relations), *extra large* (over 1,000 relations)
number of axioms	number of axioms to be matched: *no axioms*, *small* (up to 30 axioms), *medium* (31 - 100 axioms), *large* (100 - 1,000 axioms), *large* (over 1,000 axioms)
Factor: Input category (algorithm capable of handling:)	
glossary	a list of terms with their definitions
thesaurus	a list of important terms (single-word or multi-word) in a given domain and a set of related terms for each term in the list
taxonomy	indicates only class/subclass relationship (hierarchy) [Dogac *et al.*, 2002]
to be continued ...	

Attribute	Description
DBschema	often does not provide explicit semantics for their data
ontology	an explicit specification of a conceptual [Gruber, 1995]; describes a domain completely [Dogac *et al.*, 2002]
Factor: Input formality level [Uschold and Jasper, 1999] (algorithm capable of handling:)	
(highly/semi) informal ontology	expressed loosely in natural language or in a restricted and structured form of natural language
semi-formal ontology	expressed in an artificial, formally defined language
(rigorously) formal ontology	meticulously defined terms with formal semantics, theorems and proofs of such properties as soundness and completeness
Factor: Input model type [Guarino, 1998] (algorithm capable of handling:)	
task ontology	model build for a generic task (e.g. diagnosing)
domain ontology	model of a generic domain (e.g. medicine) or part of the world
application ontology	model built for a specific application; concepts depend on both a particular domain and task, which are often specializations of both the related ontologies
upper-level ontology	model of the common objects that are generally applicable across a wide range of domain ontologies; it describes very general concepts (e.g. space, time)
Factor:Input type (algorithm capable of handling:)	
scheme	schema-based matcher
instance	instance/content-based matchers
Factor: External sources (algorithm is able to handle /to provide:)	
additional user input	most matchers rely not only on the input to be matched (like schemas or instances) but also on auxiliary information
previous matching decision	
training matches	
domain specific resources/ constrains	
domain constrains	
list of valid domain values	
dictionary	
missmatch information	
matching rules	
global schemas	
Factor: Input natural language (NL) (algorithm is:)	
NL-specific (one language)	the approach is dependent on one natural language
NL-specific (many languages)	the approach is dependent on more than one natural languages
NL-independent	the approach is language independent
Factor: Input representation language (RL) [Uschold and Jasper, 1999] (algorithm is:)	
RL-specific (one language)	the approach is dependent on one rep. language
RL-specific (many languages)	the approach is dependent on more then one rep. languages
RL-independent	the approach is independent on rep. language
Factor: Input structure (algorithm capable of handling:)	
tree structure	the approach can handle only tree-structures
DAGs structure	the approach can handle directed acyclic graphs structures
graph structure	the approach can handle (heterogenous) graph structures
is-a relations	the approach can handle is-a relations
heterogeneous relations	the approach can perform additionally to the "is-a relations" also on other relations

Table 3.1: Input characteristics

3.2 Approach characteristics

The second crucial dimension characterizes the matching approaches themselves. The corresponding factors and attributes compile a list of matcher features that are empirically proved to have an impact on the quality of matching tasks. They consider e.g. the common classification of the approaches [Do *et al.*, 2002; Rham and Bernstein, 2001; Shvaiko, 2004b] and distinguish between *individual algorithms* [Giuchiglia and Shvaiko, 2004; Stumme and Mädche, 2001a] and combina-

tions of the individual algorithms: *hybrid* and *composite solutions*. A hybrid approach [Madhavan *et al.*, 2001a] follows a *black box paradigm*, in which various individual matchers are synthesized into a new algorithm, while the composite matchers allow an increased user interaction [Do and Rahm, 2002a; Doan *et al.*, 2004a]. The *approach characteristics* also takes into account issues like processing type, matching ground and execution parameter (cf. Table 3.2).

Attribute	**Description**
Factor: Matcher Type (algorithm is a(n):	
individual matcher	computes a mapping based on a single matching criteria
combined matcher	uses multiple individual matchers
Factor: Processing (algorithm supports:)	
manual execution	manual execution
gray box paradigm	semi-automatic execution where the human intervention is possible
black box paradigm	automatic execution without human intervention
manual preprocessing allowed/required	human intervention before execution is allowed or even required
manual postprocessing allowed/required	human intervention after execution is allowed or even required
Factor: Execution Type (algorithm supports:)	
simultaneous execution	the single matching algorithms (within a composite matcher) can be executed simultaneously
sequential execution	the single matching algorithms (within a composite matcher) can be executed sequentially
Factor: Kind of Similarity Relation (algorithm performs:)	
syntactic matching	similarity based on syntax driven techniques and syntactic similarity measures; relation computed between labels at nodes [Shvaiko, 2004b]
semantic matching	relation computed between concepts at nodes [Shvaiko, 2004b]
Factor: Matcher Level (algorithm can perform on:)	
element level	match performed for individual schema elements
structure level	match performed for complex schema structures
atomic level	elements at the finest level of granularity are considered e.g. attributes in an XML schema [Rham and Bernstein, 2001]
non-atomic (higher) level	e.g. XML elements
Factor: Matching Ground	
heuristic	"guessing" relations between similar labels or graph structures [Shvaiko, 2004a]
formal	uses formal techniques (e.g. can have model-theoretic semantics which is used to justify the results) [Shvaiko, 2004a]
Factor: Semantic Codification Type(algorithm uses:)	
implicit techniques	syntax driven techniques [Shvaiko, 2004a](e.g. considers labels as strings)
explicit techniques	exploit the semantics of labels [Shvaiko, 2004a]; uses an external sources for assessing the meaning of labels
Factor: Execution Parameter (algorithm needs:)	
max time of execution	describes the maximal time needed for execution
max disc space for execution	describes the maximal disc space needed
precision	expresses the proportion of relevant retrieved matches [van Rijsbergen, 1979]
recall	expresses the proportion of relevant documents retrieved [van Rijsbergen, 1979]

Table 3.2: Approach characteristics

3.3 Usage characteristics

One of the fundamental requirements for the realization of the vision of the fully developed Semantic Web are proven ontology matching algorithms. Though containing valuable ideas and techniques some of the current matching approaches lack exhaustive testing in real world scenar-

ios. Considering this problem and additionally making allowance for the fact that some of the algorithms cannot be applied across various domains to the same effect [Giuchiglia and Shvaiko, 2004], it is important to know, if a particular approach has already been successfully adapted for different *domains*, *applications* and *tasks*. Additionally, the *usage characteristics* dimension also considers *different types of users*: ontology engineers who e.g. look for means to compare sources for building a new ontology or Web Services seeking automatized methods to generate mediation ontologies (cf. Table 3.3).

Attribute	**Description**
Factor: Usage goal (algorithm is built for:)	
local use	the matcher is run on the local machine
network use	the matcher is accessible on a network
internet-based use	the matcher service is available through internet
Factor: Application Area (algorithm is built for:)	
reuse of sources	the matching approach is deployed for ontology reuse which may be defined as a process in which available knowledge is used to generate new ontologies
usage of sources	the matching approach is applied for use ontologies (within an application) e.g. to compare profiles
integration	reuse of available source ontologies within a range in order to build a new ontology which serves at a higher level in the application than that of various ontologies in ontology libraries [Li *et al.*, 2005]
transformation	associated with the ontology evolution that uses matching for finding the changes that have occurred between two ontology versions
merging	the matching approach is used for integrating the schemas of different databases under a single view
translation	ontology translation is required to translate data sets, generate ontology extensions and query through different ontologies [Dou *et al.*, 2003] e.g. (i) catalog integration - matchers are used to offer a integrated access to online catalogs, (ii) multi agent communication - matchers are used to find the relations between the ontologies used by two agents and translating the messages they exchange, (iii) context matching in ambient computing uses matching approaches of application needs and context information when application and devices have been developed independently and use different ontologies
query answering	can be applied to data integration or P2P information sharing where matching approaches are used to find the relations of ontologies used by different peers
data mediation	applied within web service composition in which matchers are used between ontologies to describe service interfaces in order to compose web services by connecting their interfaces;
query reformulation	uses matcher to translate user queries about the web
navigation	uses matchers to annotate web pages with partially overlapping ontologies
Factor: Usage type (algorithm is:)	
human applicable	approach can be used only by humans (human interaction indispensable)
machine applicable	approach can be used by machine as a service
Factor: Adaptation ability (algorithm has been applied for:)	
number of domains	number of different domains the matching approach was applied to
number of applications	number of different applications the matching approach was applied to
number of tasks	number of different tasks the matching approach was applied to
reference of usage	the approach as has been utilized by other users

Table 3.3: Usage characteristics

3.4 Output characteristics

In addition to the input, approach and usage dimensions, the *output characteristics* (cf. Table 3.4) plays a decisive role in the process of selecting the suitable matching algorithm. Depending on the given requirements, an application may need a matcher that considers only some of elements of

the schemas, while other systems may mandate a match for all elements. One of the key factors in this dimension is the *cardinality* (global vs. local cardinality) which specifies whether a matcher compares one or more elements of one scheme with one or more elements of another scheme (in some cases the results are based on a one-to-one mapping between taxonomies [Doan *et al.*, 2001a] and in others on one-to-many).

Attribute	Description
Factor: Output type	
deliver relations	the output is not restricted to correspondence of equivalence
deliver value	e.g. matcher used to determine the semantic similarity between concepts
deliver understandable (for humans) results	matcher delivers some explanations of the results
Factor: Matching Cardinality	
global 1:1	relationship cardinalities between entities w.r.t different entities [Rham and Bernstein, 2001] e.g. one-to-one or many-to-many alignments.
global n:1	
global 1:m	
global n:m	
local 1:1	relationship cardinalities between entities w.r.t an individual correspondence [Rham and Bernstein, 2001] e.g., simple or complex correspondences
local n:1	
local 1:m	
local n:m	
Factor: Execution Completeness	
full match	considers all elements of the schemes
partial match	considers only some elements of the schemes
injective match	distinct elements of the domain is mapped to distinct elements of the range
surjective match	all elements of the range are mapped to elements of the domain

Table 3.4: Output characteristics

3.5 Documentation characteristics

Due to the fact that documentation is an essential part of every software product and, for engineering purposes, often more important than the program code [Humphrey, 1999] the information about its quality and clarity can be significant for the selection of an approach. Furthermore, since one of the goals of documentation is to provide sufficient information so that an architecture can be analyzed for suitability to the purpose [Clements *et al.*, 2002], it could be a determining coefficient for the selection of a particular algorithm, especially if the algorithm is to be reused in a different context from the domain or application it was originally developed for (cf. Table 3.5).

Attribute	Description
quality of documentation	quality of the available documentation
clarity of documentation	clarity of the available documentation
clarity of maturity description	clarity of the description of the approach's maturity
availability of examples	are examples of the approach available

Table 3.5: Documentation characteristics.

3.6 Cost characteristics

The last dimension, *cost characteristics*, describes the financial factors regarding the (commercial[6]) usage of a single matching approach like the matcher licence or the access to the appropriate matcher interface (cf. Table 3.6).

Attribute	Description
costs of matcher licence	the costs entailed to acquire the matcher licence
costs of matcher tool licence	the costs entailed to acquire the tools matcher have been developed with
costs of access matcher interface	the costs entailed to acquire the use of the interface

Table 3.6: Cost characteristics.

3.7 Summary

We have investigated the various dimensions of matchers and the characteristics of these matchers along these dimensions. This inventory is very precise and goes beyond most published characterizations of matching systems in terms of attributes [Rahm and Bernstein, 2001; Kalfoglou and Schorlemmer, 2003b; Shvaiko and Euzenat, 2005b; Huza *et al.*, 2006; Euzenat and Shvaiko, 2007]. Its goal is to be able to establish more precisely the adequacy of a matcher with an application. It will be used, in particular for gathering information about available matching systems and choosing one of these (see Chapter 5.2 and 6.3).

[6] At the moment there is no commercial automatic matching tool. Nevertheless to be able to deal with such problem in the future we are taking already now these criteria into account.

Chapter 4

Benchmarking matching systems

Once we have determined the characteristics on which matchers can be compared, it is necessary to assess the value of these characteristics applying to matchers. This can be achieved through literature review, direct survey of developers or evaluating matching systems. We present here two ways to evaluate matching systems. The first way is benchmarking, i.e., comparing systems on a range of tests that allows to assess the behaviour of systems in a number of well-characterized conditions. Another way consists of designing an evaluation procedure which is fully related to the application to be developed.

We provide an extensive presentation of benchmarking through the first three sections and a possible specific application-specific evaluation related to Knowledge web use case 1.

4.1 Principles

In order to evaluate the capability of matching systems to really work with some kind of input, we have developed a complete set of benchmarks which goal is to finely test the results of these systems in well identified situations.

These tests have proved useful at the beginning of the Ontology Alignment Evaluation Initiative and their first version had been presented in [Euzenat *et al.*, 2004b]. We describe here the extended version that has been used for two years. Their results in finding the adequacy of a matching system for an application is given in Section 5.3.

The tests consist of choosing some ontology and systematically generating a new ontology by discarding some information from it, e.g., names, properties, subclass relation. A reference alignment between the two ontologies can be generated in a similar way since what corresponds to what is clear. Matching system must provide an alignment between these two ontologies tht will be compared with the reference alignment. This is useful to evaluate how the system behave when this information is lacking.

The domain of this reference ontology is Bibliographic references. It is, of course, based on a subjective view of what must be a bibliographic ontology. There can be many different classifications of publications (based on area, quality, etc.). We choose the one common among scholars based on mean of publications; as many ontologies below (tests #301-304), it is reminiscent to BibTeX.

The complete ontology is that of test #101. This reference ontology contains 33 named classes, 24 object properties, 40 data properties, 56 named individuals and 20 anonymous individuals.

It is based on the one of the first EON Ontology Alignment Contest benchmark which has been improved by comprising a number of circular relations that were missing from the first test. This set is also more complete (the 2004 benchmark only had the 16 easier ontology pairs). Test numbering (almost) fully preserves the numbering of the first EON contest so that results can be compared.

The reference ontology is put in the context of the semantic web by using other external resources for expressing non bibliographic information. It takes advantage of FOAF[1] and iCalendar[2] for expressing the People, Organization and Event concepts. Here are the external reference used:

http://www.w3.org/2002/12/cal/#:Vevent (defined in http://www.w3.org/2002/12/cal/ical.n3 and supposedly in http://www.w3.org/2002/12/cal/ical.rdf).

http://xmlns.com/foaf/0.1/#:Person (defined in http://xmlns.com/foaf/0.1/index.rdf)

http://xmlns.com/foaf/0.1/#:Organization (defined in http://xmlns.com/foaf/0.1/index.rdf).

This reference ontology is a bit limited in the sense that it does not contain subclasses of several non comparable classes. Similarly the kind of proposed alignments is still limited: they only match named classes and properties, they mostly use the "=" relation with confidence of 1. The ontologies are described in OWL-DL and serialized in the RDF/XML format.

This data set can be considered as correct by construction. It is not realistic nor very hard: it is based on small tests and offers some easy ways to reach the correct result. It is especially made for evaluating the strengths and weaknesses of the matching systems.

4.2 The test set

Table 4.1 summarizes what has been retracted from the reference ontology. There are here 6 categories of alteration:

Entity labels Name of entities that can be replaced by (R/N) random strings, (S)ynonyms, name with different (C)onventions, (F) strings in another language than English.

Comments Comments can be (N) suppressed or (F) translated in another language.

Specialization Hierarchy can be (N) suppressed, (E)xpanded or (F)lattened.

Instances can be (N) suppressed

Properties can be (N) suppressed or (R) having the restrictions on classes discarded.

Class composition can be (E)xpanded, i.e., replaced by several classes or (F)latened.

The alteration results in a set of 46 pairs of ontologies (out of $2^6 = 64$ ontology pairs that could be obtained by all the Boolean combinations of alterations or the full set of $5\times3\times4\times2\times3\times3 = 1080$ combinations). This set is more restricted because some of the elaborate modifications have only been applied once. This ensures a good coverage of all the situations in which an alignment algorithm can find itself and provides a good view of strengths and weaknesses of algorithms.

[1] http://xmlns.com/foaf/0.1/

[2] http://www.w3.org/2002/12/cal/

Test	Entity labels	Comments	Subsumption hierarchy	Instances	Properties	Class composition	Comment	Used	2004	Aggregate
101							Reference alignment	√	√	liph
102							Irrelevant ontology		√	
103							Language generalization	√	√	
104							Language restriction	√	√	
201	R						No names	√	√	phi
202	R	N					No names, no comments	√	√	phi
203		N					No comments (was misspelling)	√		phi
204	C						Naming conventions	√	√	phi
205	S						Synonyms	√	√	phi
206	F	F					Foreign names	√	√	phi
207	F							√		phi
208	C	N						√		phi
209	S	N						√		phi
210	F	N						√		phi
221			N				No specialisation	√	√	lip
222			F				Flatenned hierarchy	√	√	lip
223			E				Expanded hierarchy	√	√	lip
224				N			No instance	√	√	lph
225					R		No restrictions	√	√	hil
226							No datatypes			
227							Unit difference			
228					N		No properties	√	√	hil
229							Class vs instances			
230						F	Flattened classes	√	√	hil
231						E	Expanded classes	√		hil
232			N	N				√		lp
233			N		N			√		li
236				N	N			√		lh
237			F	N				√		lp
238			E	N				√		lp
239			F		N			√		li
240			E		N			√		li
241			N	N	N			√		l
243			N	N						
244			N		N					
246			F	N	N			√		l
247			E	N	N			√		l
248	N	N	N					√		ip
249	N	N		N				√		ph
250	N	N			N			√		hi
251	N	N	F					√		ip
252	N	N	E					√		ip
253	N	N	N	N				√		p
254	N	N	N		N			√		i
257	N	N		N	N			√		h
258	N	N	F	N				√		p
259	N	N	E	N				√		p
260	N	N	F		N			√		i
261	N	N	E		N			√		i
262	N	N	N	N	N			√		∅
265	N	N	F	N	N			√		∅
266	N	N	E	N	N			√		∅

Table 4.1: Table of tests and altered features (test numbers are briefly described in the corresponding section). Some of these tests where not used in 2004 and some other are still not used.

Table 4.1 present all the tests and their generation principles. In order to provide a readable evaluation in Chapter 5.3, we have grouped together several characteristics of these tests in order to evaluate the dependency of each system on particular characteristics. Entity labels and comments are considered as (l)abels, subsumption (h)ierarchy is a category alone, (i)nstances and (p)roperties and class composition are grouped together. When these features have not been altered, the cell corresponding to this characteristics is left blank. This is reflected by the aggregate column of Table 4.1.

We provide below the description of all the tests as they where used in the first experiment, this should provide a raw blueprint on what is to be expected of such a competence benchmark tests. The subsection number is the number of the test in Table 4.1.

4.2.101 Identity

This simple test consists of aligning the reference ontology with itself.

4.2.201 No names

Each label or identifier is replaced by a random one.

4.2.202 No names, no comment

Each label or identifier is replaced by a random one. Comments (rdfs:comment and dc:description) have been suppressed as well.

4.2.203 Misspelling of names

Each label or identifier is replaced by a misspelled one. Comments (rdfs:comment and dc:description) have been suppressed as well.

4.2.204 Naming conventions

Different naming conventions (Uppercasing, underscore, dash, etc.) are used for labels. Comments have been suppressed.

4.2.205 Synonyms

Labels are replaced by synonyms. Comments have been suppressed.

4.2.206 Foreign names

The complete ontology is translated to another language than English (French in the current case, but other languages would be fine).

4.2.221 No hierarchy

All subclass assertions to named classes are suppressed.

4.2.222 Flattened hierarchy

A hierarchy still exists but has been strictly reduced (by suppressing one intermediate class out of two).

4.2.223 Expanded hierarchy

Numerous intermediate classes are introduced within the hierarchy.

4.2.224 No instances

All individuals have been suppressed from the ontology.

4.2.225 No restrictions

All local restrictions on properties have been suppressed from the ontology.

4.2.226 No datatypes

In this test all datatypes are converted to xsd:string.

4.2.227 Unit differences

(Measurable) values are expressed in different datatypes.

4.2.228 No properties

Properties and relations between objects have been completely suppressed.

4.2.229 Class vs instances

Some classes have become instances.

4.2.230 Flattening entities

Some components of classes are expanded in the class structure itself (e.g., year, month, day attributes instead of date).

4.2.231 Multiplying entities

Some classes are spread over several classes. This is the opposite as the previous test: intermediate classes and objects aggregating properties are created (e.g., an inproceedings reference with booktitle and conference properties is transformed into a inproceedings that refers to a proceedings reference that refers to a conference object).

4.3 Application-specific evaluation

So far matcher evaluation has been considered in general. However, the evaluation could also be considered in the context of a particular application or a particular kind of applications. Application specific evaluation is dedicated to find a suitable system for a particular task. This is especially useful for application designers who need to integrate a matching system.

Application specific evaluation can be carried out by having a specific evaluation setting. This has the advantage of being more realistic than artificial testbenches and of providing very specific information, but the drawback to be changed for each different application.

An application specific evaluation has to start with a selection of the task corresponding to the application. It is moreover useful to set up experiments which do not stop at the delivery of alignments but carry on with the particular task. This is especially true when there is a clear measure of success of the overall task. Such a setting assists in focusing on the most useful issues for the task. For instance, it may be the case that the gain in accuracy in one algorithm over another is not useful for the task while the gain in speed of the latter really matters. If no clear measure is available, then using a weighted aggregation measure like suggested above would help.

Nevertheless, it is extremely difficult to determine the evaluation value of the matching process independently. The effects of other components of the overall application have to be carefully filtered out.

There are several problems associated with this approach:

- It will be difficult to account for the performances of matching algorithms if the systems which carry the task are different. If these system are a single system, then the evaluation would be simpler in comparing the alignments and applying a specific metric;
- Very often the matching systems are considered as semi-automatic (i.e., a user must control the result). The task cannot be accomplished in isolation (and this brings back the issue of involving the user in the evaluation loop).
- This would require a specific setting for each task.

As an example, let us consider Knowledge web use case 1 dedicated to human resources. Its main role is to match job offers to applications. The success criterion here is matching adequate applications to job offers. Deciding which application to choose is difficult but there is no difficulty in discarding non relevant applications.

In this recruitment application the data exchange between employers, job applicants and job portals is based on a set of shared vocabularies describing domain relevant terms: occupations, industrial sectors and skills. These commonly used vocabularies can be formally defined by means of a human resource ontology. The ontology represents domain specific knowledge and might be used to determine semantic similarity between data describing job applications and job offers. Furthermore, semantic matching is a matching technique that can combine annotations using controlled vocabularies with background knowledge about a certain application domain. In addition, to enable an accurate scheme validation of the incoming data the human resource ontology is modelled using OWL which contains over 8.000 concepts and less than 100 different properties[3]. The descriptions of job postings and applicants' profiles (instance data) are stored in RDF using the vocabulary defined by the human resource ontology which does not contain any axioms.

[3]These ontologies are not freely accessible and cannot be presented in more depth.

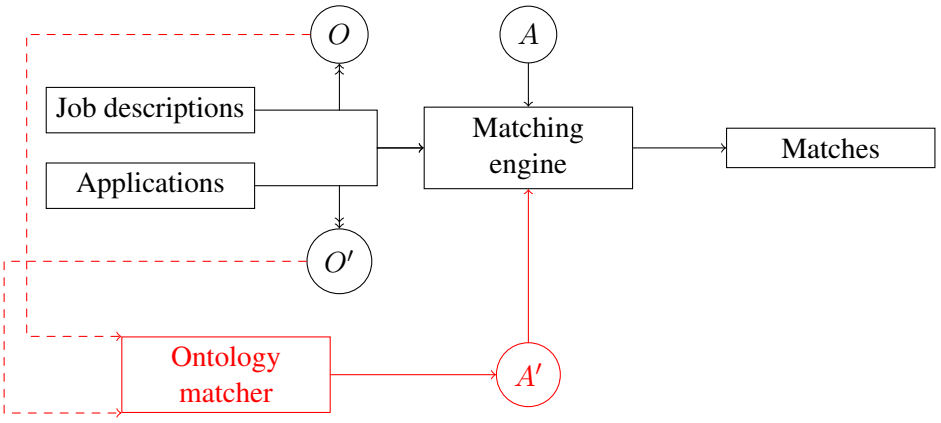


Figure 4.1: The job-application matching cases. The existing application (in black) match job descriptions to applications by using a wired ontology alignment (A). The modified application (in red) uses ontology matching for generating an alignment from the ontologies (A').

To pinpoint the appropriate job for an applicant or a suitable candidate for a job opening, we needed semantic matching approaches which can deal with the highly formal human resource ontology and with the specific application requirements. The job portal developed for a particular country (e.g. Germany) must automatically compare a particular applicant profile against a multitude of job openings (or one job description against a number job seekers). For this purpose, the Matching engine uses a wired alignment (A) between the human resource ontology (O) and and the terms used in the different domains (O').

An experimental setting for this application is provided in Figure 4.1. It consists in using ontology matchers in order to generate an alignment that can be used by the matching engine in replacement of the static one and comparing the results of the whole process on some reference set.

This would provide a very precise evaluation of matcher performances in the context of the application. However, in the case of the human resource use case, this requires important modifications in the matching engine.

4.4 Summary

We have presented evaluation methods and principles for matching systems. The purpose of these tests is to allow assessing the behaviour of matching systems on a precise type of problem, i.e., when some part of the ontologies are mismatching.

This will be exploited in Section 5.3 where their results are used in order to produce profiles of matchers with regard to these characteristics.

Chapter 5

Matcher profiles

We analyse here the matchers with three different techniques in order to assess the adequation of matchers on the identified dimensions. These techniques are: literature analysis for finding the properties presented in papers (partially based on [Euzenat *et al.*, 2004a]); exploitation of questionnaires as well as by intensive collaborations with developer of matching approaches, results of the evaluations performed according to Chapter 4.

5.1 Analysis from literature

The panorama of matching systems is as large as diverse. This is not our purpose here to do yet another description of these systems. The interested reader is invited to consult [Euzenat *et al.*, 2004a; Euzenat and Shvaiko, 2007]. We here compare these systems and classify them with regard to the criteria identified in Chapter 2.

There are a number of constant features that are shared by the majority of systems. Also, usually each individual system innovates on a particular aspect. Let us summarise some global observations concerning the systems found in the literature and presented in [Euzenat and Shvaiko, 2007]:

- based on the number of systems considered, we can conclude that schema-based matching solutions have been so far investigated more intensively than the instance-based solutions. We believe that this is an objective trend, since we have striven to cover state of the art systems without bias towards any particular kind of solutions.
- most of the systems under consideration focus on specific application domains, such as books and music, as well as on dealing with particular ontology types, such as DTDs, relational schemas and OWL ontologies. Only a small number of systems aim at being general, i.e., suit various application domains, and generic, i.e., handle multiple types of ontologies. Some examples of the latter include Cupid [Madhavan *et al.*, 2001b], COMA and COMA++ [Do and Rahm, 2002b], Similarity Flooding [Melnik *et al.*, 2002b], and S-Match [Giunchiglia and Shvaiko, 2003].
- most of the approaches take as input a pair of ontologies, while only a small number of systems take as input multiple ontologies. Some examples of the latter include DCM [Chang *et al.*, 2005] and Wise-Integrator [He *et al.*, 2004].
- most of the approaches handle only tree-like structures, while only a small number of systems handle graphs. Some examples of the latter include Cupid [Madhavan *et al.*, 2001b],

COMA and COMA++ [Do and Rahm, 2002b], and OLA [Euzenat and Valtchev, 2004].
- most of the systems focus on discovery of one-to-one alignments, while only a small number of systems have tried to address the problem of discovering more complex correspondences, such as one-to-many and many-to-many, e.g., iMAP [Dhamankar *et al.*, 2004] and DCM [Chang *et al.*, 2005].
- most of the systems focus on computing confidence measures in $[0\ 1]$ range, most often standing for the fact that the equivalence relation holds between ontology entities. Only a small number of systems compute logical relations between ontology entities, such as equivalence, subsumption. Some examples of the latter include CtxMatch [Bouquet *et al.*, 2006] and S-Match [Giunchiglia and Shvaiko, 2003].

In conclusion, there is a large set of systems sharing the same type of features: schema-based matchers taking OWL or database schema as input and outputting equivalence correspondences. If this is what the application requires, then there is a wealth of systems to choose from. If the application has different requirements, there may be very few systems to choose from, if any.

Table 5.1 summarises how the matching systems cover the solution space in terms of the classifications of [Shvaiko and Euzenat, 2005b]. For example, S-Match [Giunchiglia and Shvaiko, 2003] exploits string-based element-level matchers, external matchers based on WordNet, propositional satisfiability techniques, etc. OLA [Euzenat and Valtchev, 2004], in turn, exploits, besides string-based element-level matchers, also a matcher based on WordNet, iterative fix-point computation, etc. Table 5.1 also testifies that ontology matching research so far was mainly focused on syntactic and external techniques. In fact, many systems rely on the same string-based techniques. Similar observation can be also made concerning the use of WordNet as an external resource of common knowledge. In turn, semantic techniques have rarely been exploited, this is only done by CtxMatch [Bouquet *et al.*, 2006], S-Match [Giunchiglia and Shvaiko, 2003] and OntoMerge [Dou *et al.*, 2005]. Concerning instance-based system, techniques based on Naive Bayes classifier and common value patterns are the most prominent.

Table 5.1: Basic matchers used by the different systems.

	Element-level		**Structure-level**	
	Syntactic	**External**	**Syntactic**	**Semantic**
Hovy [Hovy, 1998]	string-based, language-based	-	taxonomic structure	-
TranScm [Milo and Zohar, 1998]	string-based	built-in thesaurus	taxonomic structure, matching of neighbourhood	-
DIKE [Palopoli *et al.*, 2003]	string-based, domain compatibility	WordNet	matching of neighbourhood	-
SKAT [Mitra *et al.*, 1999]	string-based	auxiliary thesaurus, corpus-based	taxonomic structure, matching of neighbourhood	-
Artemis [Castano *et al.*, 2000]	domain compatibility, language-based	common thesaurus (CT)	matching of neighbours via CT, clustering	-
H-Match [Castano *et al.*, 2006]	domain compatibility, language-based, domains and ranges	common thesaurus (CT)	matching of neighbours via CT, relations	-
Anchor-Prompt [Noy and Musen, 2001]	string-based, domains and ranges	-	bounded paths matching: (arbitrary links), taxonomic structure	-

Table 5.1: Basic matchers used by the different systems (continued).

	Element-level		Structure-level	
	Syntactic	**External**	**Syntactic**	**Semantic**
OntoBuilder [Modica *et al.*, 2001]	string-based, language-based	thesaurus look up	-	-
Cupid [Madhavan *et al.*, 2001b]	string-based, language-based, datatypes, key properties	auxiliary thesauri	tree matching weighted by leaves	-
COMA & COMA++ [Do and Rahm, 2002b]	string-based, language-based, datatypes	auxiliary thesauri, alignment reuse, repository of structures	DAG (tree) matching with a bias towards various structures (e.g., leaves)	-
SF [Melnik *et al.*, 2002b]	string-based, datatypes, key properties	-	iterative fix-point computation	-
XClust [Lee *et al.*, 2002]	cardinality constraints	WordNet	paths, children, leaves, clustering	-
CtxMatch/CtxMatch2 [Bouquet *et al.*, 2006]	string-based, language-based	WordNet	-	based on description logics
S-Match [Giunchiglia and Shvaiko, 2003]	string-based, language-based	WordNet	-	propositional SAT
ASCO [Bach *et al.*, 2004]	string-based, language-based	WordNet	iterative similarity propagation	-
BayesOWL [Pan *et al.*, 2005]	text classifier	Google	Bayesian inference	-
Chang et al. [Chang *et al.*, 2005]	-	-	correlation mining	-
T-tree [Euzenat, 1994]	-	-	correlation mining	-
LSD/GLUE/ iMAP [Doan *et al.*, 2001b; 2004b] [Dhamankar *et al.*, 2004]	WHIRL, Naive Bayes	-	-	-
Kang & Naughton [Kang and Naughton, 2003]	information entropy	-	mutual information, dependency graph matching	-
sPLMap [Nottelmann and Straccia, 2005]	Naive Bayes, kNN classifier, string-based	-	-	-
SEMINT [Li and Clifton, 1994; 2000]	neural network, datatypes, value patterns	-	-	-
Clio [Miller *et al.*, 2000; Haas *et al.*, 2005]	string-based, language-based, Naive Bayes	-	-	-
NOM & QOM [Ehrig and Sure, 2004] [Ehrig and Staab, 2004]	string-based, domains and ranges	application-specific vocabulary	matching of neighbours, taxonomic structure	-
OLA [Euzenat and Valtchev, 2004]	string-based, language-based, datatypes	WordNet	iterative fix-point computation, matching of neighbours, taxonomic structure	-
Wise-Integrator [He *et al.*, 2004; 2005]	string-based, language-based, datatypes, value patterns	WordNet	clustering	-

Table 5.2 summarises the position of these systems with regard to some of the requirements of Chapter 2 (namely those requirements that can be given in the specification of the system rather than being measured). In Table 5.2, the *Input* column stands for the input taken by the systems.

In particular, it mentions the languages that the systems are able to handle (if this information was not available form the articles describing the corresponding systems we used general terms, such as database schema and ontology instead). This is, of course, very important for someone who has a certain type of ontology to match and is looking for a system. The *Needs* column stands for the resources that must be available for the system to work. This covers the automatic aspect of Chapter 2, which is here denoted by *user* when user feedback is required, *semi* when the system can take advantage of user feedback but can operate without it and *auto* when the system works without user intervention (of course, the user can influence the system by providing the initial input or evaluating the results afterwards, but this is not taken into account). Similarly, the *instances* value specifies that the system requires instances to work. Also some systems require *training* before the actual matching as well as *alignment* to be improved.

Table 5.2: Position of the presented systems with regard to some criteria of Chapter 2.

System	Input	Needs	Output	Operation
DELTA [Clifton *et al.*, 1997]	relational schema, EER	user	alignment	-
Hovy [Hovy, 1998]	ontology	semi	alignment	-
TranScm [Milo and Zohar, 1998]	SGML, OO	semi	translator	data translation
DIKE [Palopoli *et al.*, 2003]	ER	semi	merge	query mediation
SKAT [Mitra *et al.*, 1999]	RDF	semi	bridge rules	data translation
Artemis [Castano *et al.*, 2000]	relational schema, OO, ER	auto	views	query mediation
H-Match [Castano *et al.*, 2006]	OWL	auto	alignment	P2P query mediation
Tess [Lerner, 2000]	database schema	auto	rules	version matching
MapOnto [An *et al.*, 2005a; 2005b]	relational schema, XML schema, OWL	alignment	rules	data translation
Anchor-Prompt [Noy and Musen, 2001]	OWL, RDF	user	axioms (OWL/RDF)	ontology merging
OntoBuilder [Modica *et al.*, 2001]	web forms, XML schema	user	mediator	query mediation
Cupid [Madhavan *et al.*, 2001b]	XML schema, relational schema	auto	alignment	-
COMA & COMA++ [Do and Rahm, 2002b]	relational schema, XML schema, OWL	user	alignment	data translation
SF [Melnik *et al.*, 2002b]	XML schema, relational schema	user	alignment	-
XClust [Lee *et al.*, 2002]	DTD	auto	multi-alignment	-
ToMAS [Velegrakis *et al.*, 2004]	relational schema, XML schema	query, alignment	query, alignment	data transformation
OntoMerge [Dou *et al.*, 2005]	OWL	alignment	ontology	ontology merging
CtxMatch/CtxMatch2 [Bouquet *et al.*, 2006]	classification, OWL	user	alignment	-
S-Match [Giunchiglia and Shvaiko, 2003]	classification, XML schema, OWL	auto	alignment	-
HCONE-merge [Kotis *et al.*, 2006]	OWL	semi, user	ontology	ontology merging
MoA [Kim *et al.*, 2005]	OWL	auto	axioms OWL	-

Table 5.2: Position of these systems with regard to the requirements of Chapter 2 (continued).

System	Input	Needs	Output	Operation
ASCO [Bach *et al.*, 2004]	RDFS, OWL	auto	alignment	-
BayesOWL [Pan *et al.*, 2005]	classification, OWL	auto	alignment	
OMEN [Mitra *et al.*, 2005]	OWL	auto	alignment	-
DCM [Chang *et al.*, 2005]	web form	auto	multi-alignment	data integration
T-tree [Euzenat, 1994]	ontology	auto, instances	alignment	-
CAIMAN [Lacher and Groh, 2001]	classification	semi, instances, training	alignment	-
FCA-merge [Stumme and Mädche, 2001b]	ontology	user, instances	ontology	ontology merging
LSD/GLUE [Doan *et al.*, 2001b; 2004b]	relational schema, XML schema, taxonomy	auto, instances, training	alignment	-
iMAP [Dhamankar *et al.*, 2004]	relational schema, XML schema	auto, instances, training	mediator	-
Automatch [Berlin and Motro, 2002]	relational schema	auto, instances, training	alignment	-
SBI&NB [Ichise *et al.*, 2004]	classification	auto, instances, training	alignment	-
Kang & Naughton [Kang and Naughton, 2003]	relational schema	instances	alignment	-
Dumas [Bilke and Naumann, 2005]	relational schema	instances	alignment	-
Wang & al. [Wang *et al.*, 2004]	web form	instances	alignment	data integration
sPLMap [Nottelmann and Straccia, 2005]	database schema	auto, instances, training	rules	data translation
SEMINT [Li and Clifton, 2000]	relational schema	auto, instances (opt.), training	alignment	-
Clio [Miller *et al.*, 2000; Haas *et al.*, 2005]	relational schema, XML schema	semi, instances (opt.),	query transformation	data translation
IF-Map [Kalfoglou and Schorlemmer, 2003a]	KIF, RDF	auto, instances, common reference	alignment	-
NOM & QOM [Ehrig and Sure, 2004; Ehrig and Staab, 2004]	RDF, OWL	auto, instances (opt.)	alignment	-
oMap [Straccia and Troncy, 2005]	OWL	auto, instances (opt.), training	alignment	query answering
Xu & al. [Xu and Embley, 2003; Embley *et al.*, 2004]	XML schema, taxonomy	auto, instances (opt.), training	alignment	-
Wise-Integrator [He *et al.*, 2004; 2005]	web form	auto	mediator	data integration
OLA [Euzenat and Valtchev, 2004]	RDF, OWL	auto, instances (opt.)	alignment	-
Falcon-AO [Hu *et al.*, 2005]	OWL	auto instances (opt.)	alignment	-
Corpus-based matching [Madhavan *et al.*, 2005]	relational schema, web form	text corpora, instances, training	alignment	-
APFEL [Ehrig *et al.*, 2005]	RDF, OWL	user	alignment	-
eTuner [Sayyadian *et al.*, 2005]	relational schema, taxonomy	auto	alignment	-

The *Output* delivered by a system is very important because it shows the capability of the system to be used for some applications, e.g., a system delivering views and data translators cannot be used for merging ontologies. It is remarkable that many systems deliver alignments. As such, they are not fully committed to any kind of operation to be performed and can be used in a variety of applications. This could be viewed as a sign of possible interoperability between systems. However, due to the lack of a common alignment format, each system uses its own way to deliver alignments (as lists of URIs, tables, etc.). Finally, the *Operation* column describes the ways in which a system can process alignments.

Not all the requirements are addressed in Table 5.2. Indeed, completeness, correctness, runtime should not be judged from the claims of system developers. No meaningful system can be proved to be complete, correct or as fast as possible in a task like ontology matching. Therefore, the degree of fulfillment of these requirements must be evaluated and compared across systems. Moreover, different applications have different priorities regarding these requirements, hence, they may need different systems. Thus, this evaluation depends on an application in which the system is to be used.

5.2 Analysis from questionnaire

In order to collect data regarding existing matchers we have developed an online questionnaire based on the characteristics of matching approaches defined in Chapter 3. The survey allows us, on the one hand, to analyze the existing matchers, and, on the other hand, provide data important for the process of selection of the suitable matching approaches described in §6.3. Furthermore, the analysis of the data gathered enables exploitation of the valuable ideas embedded in current matching approaches and at the same time accounts for their limitations.

The matcher developers were asked to rate their algorithms by answering the 37 questions, which are divided into 8 groups:

- *introductory questions*, which are related to the developer team and its institution;
- *matcher input size-related questions* aim to collect information about the size of the input, which are served by the particular matching approach (e.g. How well does your approach support matching of sources with small number of axioms?)
- *matcher input type-related questions* gather information about the inputs the matcher handles (e.g. How well does your approach match formal ontologies?)
- *matcher approach-related questions* collect data concerning the matching approach itself (e.g. How well does your approach support black box paradigm?);
- *matcher usability-related questions* provide information regarding the applications and usage of the given matching approach (e.g. How well does your approach support sources integration?);
- *matcher output-related questions* gather data concerning the output delivered by the matching approach (How well does your approach support the global / local cardinality?)

– *matcher documentation-related questions* gather data regarding the quality of the available matcher documentation (e.g. How high is the quality of the matcher documentation?)

– *matcher costs-related questions* collect information about costs of the usage of your matching approach (e.g. How high are the costs for the matcher licence?).

The online questionnaire[1] to be filled out by the developers of the particular matchers or by the experts in the matching domain allows the addition and ranking (by using a predefined scale) of matcher alternatives. The developers weighted their algorithms according to the defined questions by using a scale between 0 and 8 (cf. Table 5.3).

Rating	Meaning
0	no answer / the approach does not support the feature
1	between no support and slightly support
2	slightly support
3	between slightly and well support
4	well support
5	between well and very well support
6	very well support
7	between very well and extremely well support
8	extremely well support

Table 5.3: Rating scale.

In the following sections we briefly analyze some data provided by the developers whose matcher took part in the Ontology Alignment Evaluation Initiative (OAEI) 2006 Campaign[2]. For certain characteristics we provide a diagram which shows a weighting of a particular matcher concerning the property given and within the description of the results we use the scale mentioned above.

5.2.1 Size-related questions

In the first set of questions the developers rated their approaches regarding the size of the input their matcher can handle. According to our analysis of the provided answers, all algorithms can *well* to *extremely well* deal with two incoming sources but only MOAM and HMatch are able to serve more then two ontologies *well* or *very well*, respectively.

Taking into account the number of ontological primitives within the sources all the approaches can handle *very well* to *extremely well* small (up to 100 primitives) and five of them (Falcon-AO, PRIOR, RiMOM, AOAS and HMatch) can still deal with medium (up to 500) size ontologies. The situation looks a bit different if we consider the large (up to 1,000) and extremely large (over 1,000) sources. Even these ontologies can be handled by the five approaches mentioned only PRIOR can extremely well tackle the problem while the others are only at an average (cf. Figure 5.1).

While analyzing the number of different types of ontological primitives, we found it striking that matching of ontologies, which contain axioms, is still a challenging issue since only four (Falcon, OWL-CtxMatch, RiMOM, HMatch) algorithms were able to handle this type of primitives (cf. Figure 5.2).

[1] `http://matching.ag-nbi.de`

[2] AOAS (NIH), Automs, Falcon-AO, ISLab HMatch, MAOM-QA, PRIOR, RIMOM, OWL CtxMatch

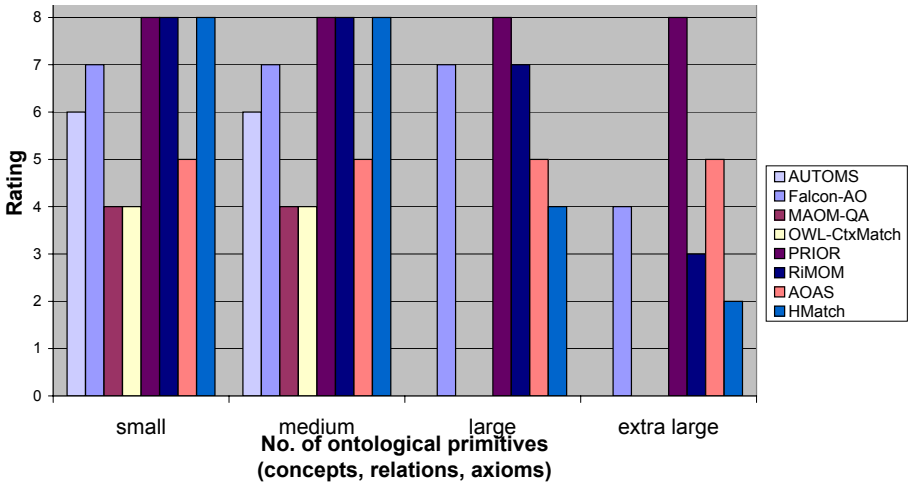


Figure 5.1: How well does the approach serve small/medium/large/extra large ontologies?

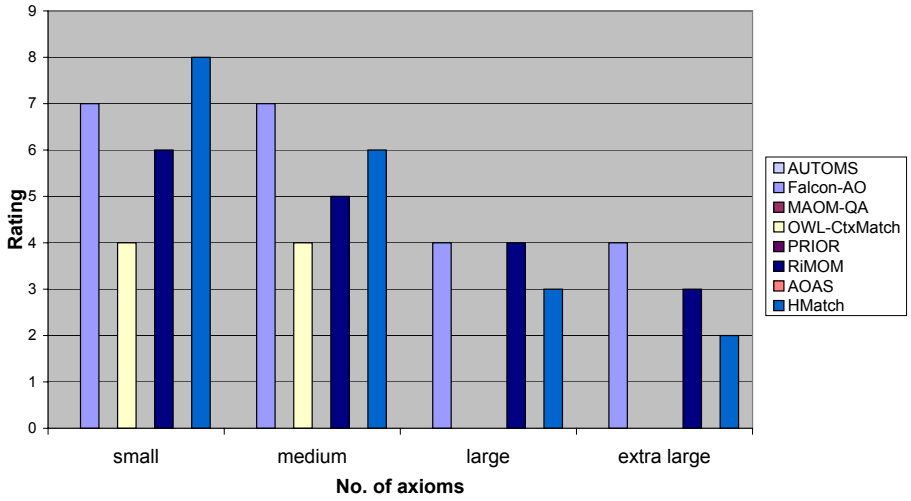


Figure 5.2: How well does the approach match ontologies with different number of axioms?

5.2.2 Input type-related questions

One of the important properties of matching approaches is the type of input on which they can operate successfully. To be able to review the "matchability" of the existing approaches regarding the heterogeneity of the incoming sources, in the subsequent set of questions we concentrated on attributes which characterize the matcher input like input category, formality level (highly/semi-informal ontology, semi-formal ontology, rigorously formal ontology, cf. §3.1), input type (instance, schema) as well as natural and representation languages.

First of all, let's analyze the existing matchers considering sources with and without instances: while almost all approaches are able to match schemes only Automs, Prior and RiMOM can handle instances. Furthermore, if we take a look at the different input categories, we notice that all algorithms can (at least well) deal with taxonomies and ontologies but only four of them (Falcon, RiMOM, AOAS, HMatch) are able to match thesauruses and, in addition, Falcon and RiMOM handle glossaries (cf. Figure 5.3).

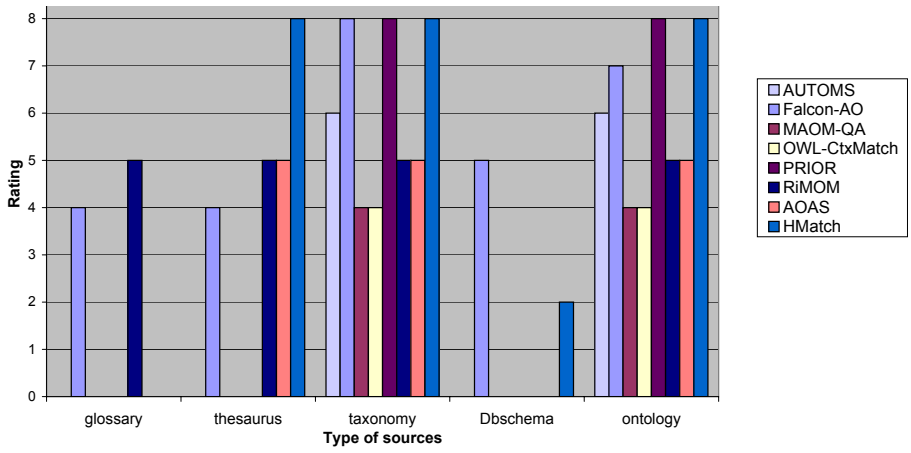


Figure 5.3: How well does the approach handle different types of sources?

Beyond this, if we analyze the heterogeneity of the input sources regarding the natural and representation languages, the best results concerning the variety of suitable matching approaches can be achieved by sources which use only one natural or representation language (Figure 5.4). Only RiMOM is a (natural) language independent approach which means that it can also serve multilingual ontologies.

5.2.3 Approach-related questions

During the ability and suitability examination of different approaches, not only the features of the input but especially the characteristics of the particular matcher itself play a crucial role. Figure 5.5 illustrates how well the approaches take into account the different processing types. While almost all systems follow the black box paradigm (i.e. automatic execution without human intervention) AOAS and RiMOM require some manual pre-processing, while the latter also needs post processing effort.

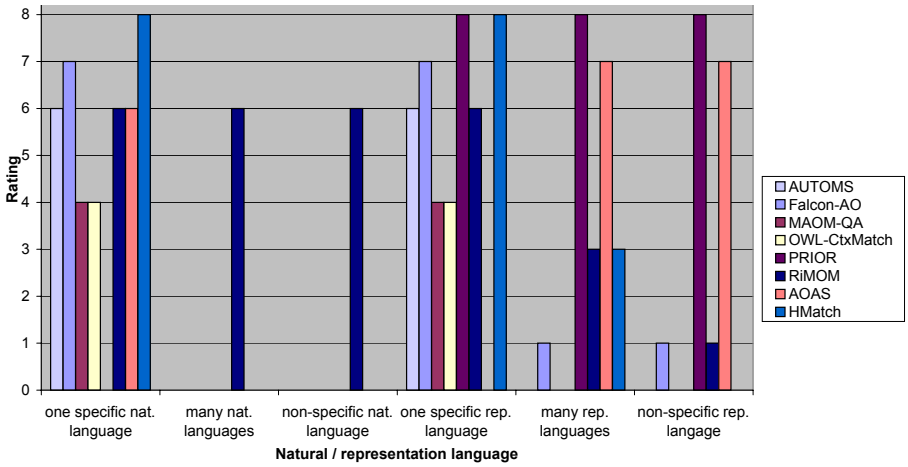


Figure 5.4: How well does the approach handle sources taking into account the rep. and nat. languages?

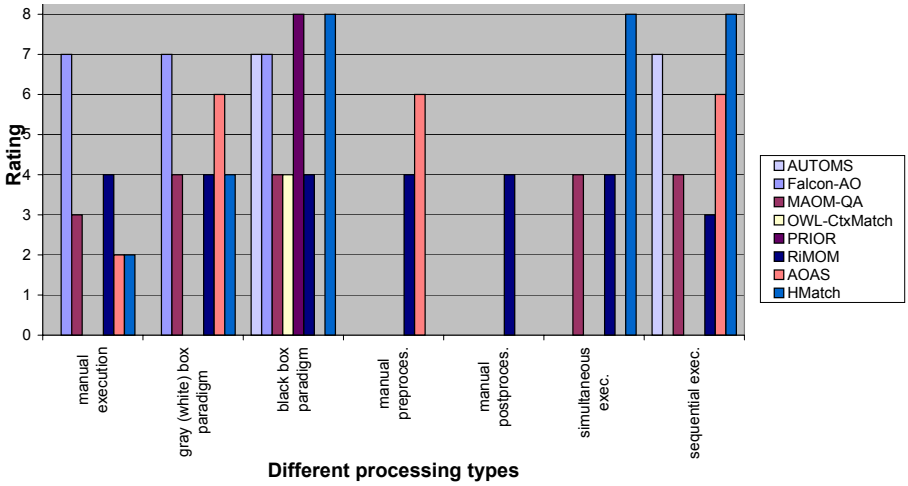


Figure 5.5: How good is the approach for different processing types?

5.2.4 Usability-related questions

Beside the features previously mentioned the matcher developers were ask to rate the approaches regarding their adaptation in the context of different application purposes (integration, translation, reuse of sources, etc.), various application goals (local use, internet use) as well as adaptation in different tasks, applications and domains. In Figure 5.6 we show as an example, how the approaches behave regarding the local, network or internet usage. Unfortunately the most of the approaches have been developed to be run on local machines and only four systems (Automs, Falcon, MAOM, and HMatch) can be accessible on a network or as an online service.

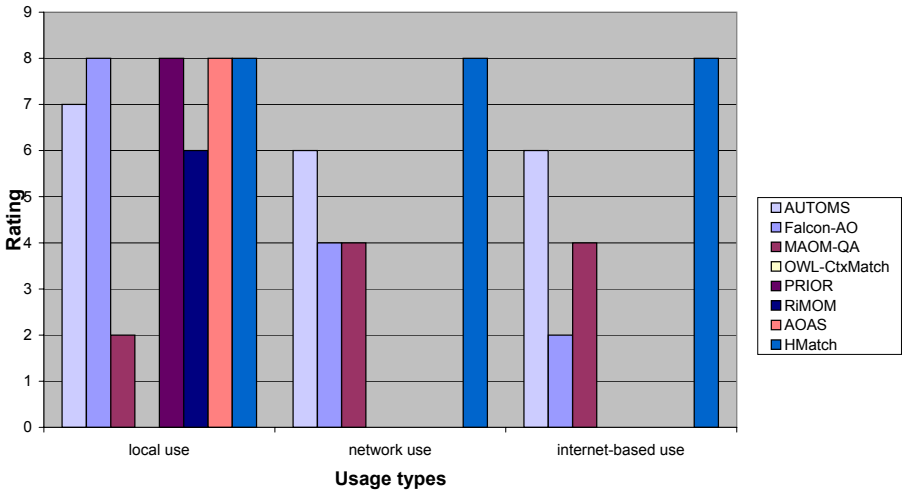


Figure 5.6: How good is the approach for different types of usage?

5.2.5 Output-related questions

The characteristics of the output is a constraint as well as those of the input. In Figure 5.7 we take a look at the output cardinality, which is one of the properties describing the output of the approaches. Falcon and HMatch (very well or extremely well respectively) support with the same intensity each type of the cardinality. Furthermore, AOAS serves only the global cardinality, RiMOM concentrates on (global and local) 1:1 while OWL-CtxMatch supports (global and local) n:m mapping.

5.2.6 Documentation-related questions

The most subjective characteristic to rate, in our opinion, is the quality and clarity of the matcher documentation. All the developers (except the OWL-CtxMatch developers) assign the rates between low (3) and high (5) for each asked aspects of the matcher documentation.

5.2.7 Conclusion

A fully developed Semantic Web will contain numerous distributed and ubiquitously available ontologies which are used in different tasks and disciplines. In turn, this means that a fundamental requirement for the realization of this vision are proved and tested ontology matching algorithms

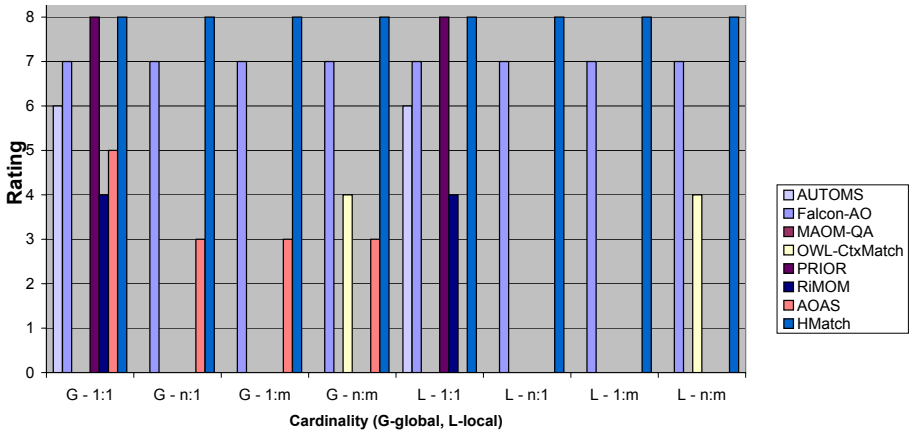


Figure 5.7: How well does the approach support global/local cardinality??

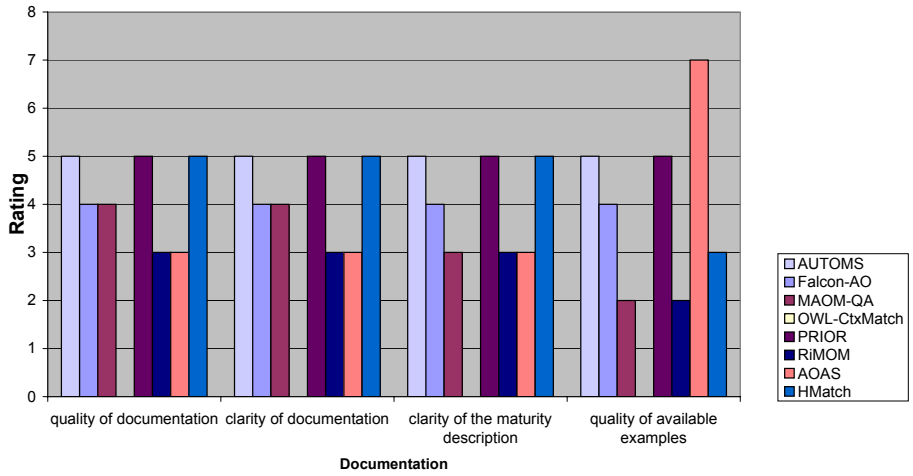


Figure 5.8: How high is the quality of the documentation?

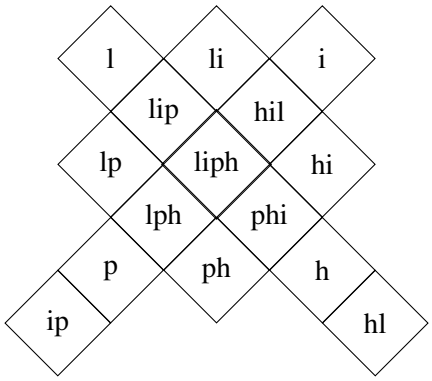


Figure 5.9: Diagram for presenting the results of matcher evaluations. Each cell corresponds to the availability of some features in the test (l=label and comments, p=properties, i=instances, h=hierarchy).

(matcher), capable of handling the heterogeneity of ontological sources available on the Web. In other words there is a need for matchers which can operate, for instance, within different contexts (i.e. different applications, tasks and domain), on different types of input (schemes, instances), on sources with various numbers of ontological primitives, and sources covering different formality levels. Taking into account the results of the survey developed, the matching of large and very large ontologies with and without axioms need to be more deeply addressed in the future. Furthermore, most matchers require specific representation and natural language, which means that approaches supporting multilingual sources and sources implemented in different representation languages are still a challenging and important issues especially with regard to the needs of people having ontologies in certain languages who wish to utilize the existing matchers for their applications.

Aside from this, Semantic Web-compatible matching methods are expected to satisfy two core requirements. First, they should be usable by ontology engineers to develop application ontologies as a combination of existing knowledge resources. This requirement is followed e.g. by the need to have, for one, a matcher output which is understandable not only to machines but also to humans, and, two, good documentation including well described examples. The second core requirement considers that matching methods are acknowledged as a core enabling technology for mediating Web Service interactions which in turn means that the approaches need to be processed automatically. Furthermore, they must be accessible not only locally but especially though the internet.

Generally speaking, despite the impressive number of research initiatives in the matching field containing valuable ideas and techniques there are still many open issues which need to be addressed in the near future.

5.3 Analysis from evaluation

We present below each system that has been evaluated with the tests of §4.2. These systems are thus able to take OWL as input and to generate alignments under the Alignment API. The presentation of each system comes either from the OM workshop proceedings (written by the participants to the evaluation) [Shvaiko *et al.*, 2006] or from [Euzenat and Shvaiko, 2007].

The results are synthesised in a diagram as explained in Figure 5.9. Each cell corresponds to a group of tests in which the ontologies to compare share a common set of characteristics (namely that the same set of features has been altered). Each cell is presented with a color representing the average of the F-measure in each of these tests. The darker the cell, the better the algorithm. The value is also presented within the diagram. Figure 5.10 shows these results for a simple measure of edit distance on class names. This shows, as expected, that such a method is only good when labels are preserved.

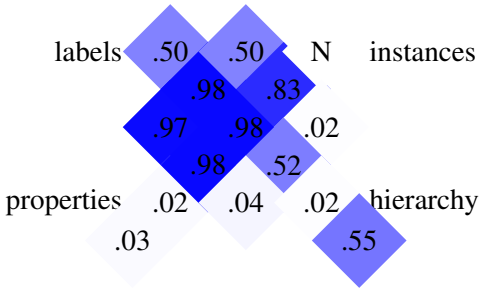


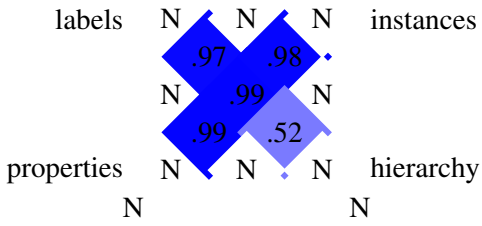
Figure 5.10: edna evaluation on F-measure (the darkest the best).

The cell in the diagram have been positioned so that the value for a cell should be related to that of its neighbours.

The use of such diagrams for an application developer consists of characterizing the cell in the diagram which correspond to the application data (by the presence/absence of labels, properties, instances or hierarchy) and to select the best matcher with regard to this cell. We applied here this technique to F-measure, however, it can be applied to precision, recall or other measures as well. Then the user can take advantage of the preference of one measure over another given in Table 6.1.

The two first systems have only a limited characterisation (restricted to 5 cells) because, they were evaluated in 2004 in which the set of tests was smaller.

5.3.1 Prompt (2004)



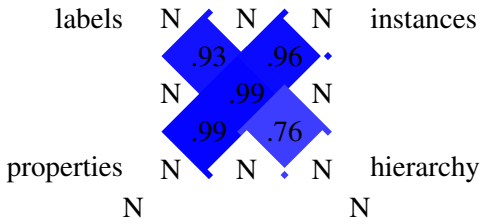
The Prompt Suite[3] is an interactive framework for comparing, matching, merging, maintaining versions, and translating between different knowledge representation formalisms [Noy and Musen, 2003; Noy, 2004]. The main tools of the suite include: an interactive ontology merging tool, called iPrompt [Noy and Musen, 2000] (formerly known as Prompt), an ontology matching tool, called Anchor-Prompt and [Noy and Musen, 2001]), an ontology-versioning tool, called PromptDiff [Noy and Musen, 2002], and a tool for factoring out semantically complete sub-ontologies, called PromptFactor.

The Prompt have used PromptDiff when participating to the Ontology Alignment Contest of 2004. PromptDiff compares ontology versions and identifies the changes. PromptDiff produces a structural diff between two versions based on heuristics. It borrows many of them from iPrompt in order to identify what has changed from one version of an ontology to another. These heuristics

[3] http://protege.stanford.edu/plugins/prompt/prompt.html

include various techniques such as analysis and comparison of concept names, slots attached to concepts. The PromptDiff approach has two parts: (i) an extensible set of heuristic matchers; and (ii) a fixed-point algorithm which combines the results of the matchers until they produce no more changes in the diff.

5.3.2 Fujitsu (2004)

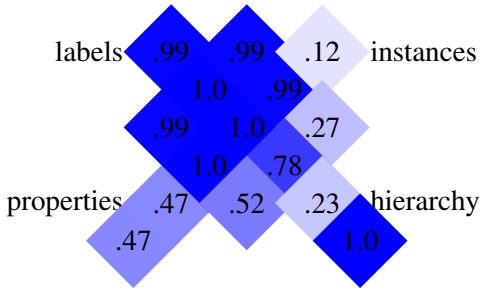


The system presented by Fujitsu implements semantic category matching (SCM) technology to the ontology alignment problem. Semantic category matching is based on a statistical approach that takes sample documents from each category and hierarchy description data, and outputs all category pairs that semantically correspond with the two classification systems. Ontology alignment is a problem designed to find couples of corresponding classes. While the purposes of SCM and ontology alignment are different, the problem structures of both are similar to each other, from the perspective of alignment between the hierarchy.

The system works through:

1. Hierarchical version of keyword extraction.
2. Similarity search category similarities, based on oblique coordinates, and
3. Structural consistency analysis.

5.3.3 OLA (2005)



OLA (OWL Lite Aligner) [Euzenat and Valtchev, 2004] is an ontology matching system which is designed with the idea of balancing the contribution of each of the components that compose an ontology, e.g., classes, constraints, data instances. OLA handles ontologies in OWL. It first compiles the input ontologies into graph structures, unveiling all relationships between entities. These graph structures produce the constraints for expressing a similarity between the elements of the ontologies. The similarity between nodes of the graphs follows two principles: (i) it depends on the category of node considered, e.g., class, property, and (ii) it takes into account all the features of this category, e.g., superclasses, properties.

The distance between nodes in the graph are expressed as a system of equations based on string-based, language-based and structure-based similarities. These distances are almost linearly aggregated (they are linearly aggregated modulo local matches of entities). For computing these distances, the algorithm starts with base distance measures computed from labels and concrete datatypes. Then, it iterates a fix-point algorithm until no improvement is produced. From that solution, an alignment is generated which satisfies some additional criterion on the alignment obtained and the distance between matched entities.

OLA also participated in 2004, but we retained the last figures, i.e., those of 2005.

5.3.4 Foam (2005)

FOAM [Ehrig, 2006], is a general tool for processing similarity-based ontology matching. It follows a general process made of the six following steps:

Feature engineering selects the features of the ontologies that will be used for comparing the entities.

Search step selection selects the pairs of elements from both ontologies that will be compared.

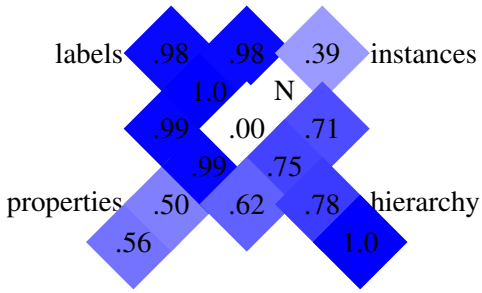
Similarity computation actually computes the similarity between the selected pairs using the selected features.

Similarity aggregation combines the similarities obtained as the result of the previous step for each pair of entities.

Interpretation extracts an alignment from the computed similarity.

Iteration iterates this process, if necessary taking advantage of the current computation.

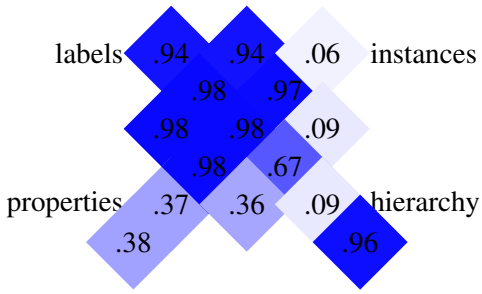
Foam adopted some of the idea of parallel composition of matchers from COMA (§5.3.10). Some innovations with respect to COMA are in the set of elementary matchers based on rules, exploiting explicitly codified knowledge in ontologies, such as information about super- and sub-concepts, super- and sub-properties, etc. At present the system supports 17 rules related to ontology features. For example, a rule states that if super-concepts are the same, the actual concepts are similar to each other. These rules use many terminological and structural techniques.



Foam also participated in 2004, but we retained the last figures, i.e., those of 2005.

5.3.5 oMap (2005)

oMap [Straccia and Troncy, 2005] is a system for matching OWL ontologies. It is built on top of the Alignment API and has been used for distributed information retrieval in [Straccia and Troncy, 2006]. oMap uses several matchers (called classifiers) that are used for giving a plausibility of a correspondence as a function of an input alignment between two ontologies. The matchers include (i) a classifier based on classic string similarity measure over normalised entity names; (ii) a Naive Bayes classifier used on instance data, and (iii) a "semantic" matcher which propagates initial weights through the ontology constructors used in the definitions of ontology entities. This last one starts with an input alignment associating plausibility to correspondences between primitive entities and computes the

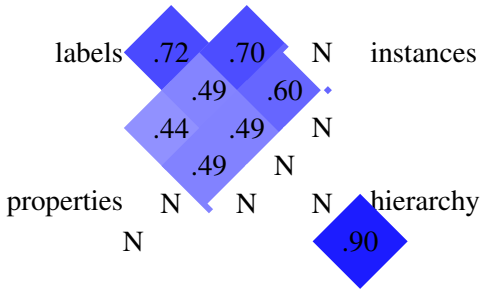


plausibility of a new alignment by propagating these measures through the definitions of the considered entities. The propagation rules depend on the ontology constructions, e.g., when passing through a conjunction, the plausibiliy will be minimised. Each matcher has its own threshold and they are ordered among themselves.

There are two ways in which matchers can work: (i) in parallel, in which case their results are aggregated through a weighted average, such that the weights correspond to the credit accorded to each of the classifiers, (ii) in sequence, in which case each matcher only adds new correspondences to the input ontologies. A typical order starts with string similarity, before Naive Bayes, and then the "semantic" matcher is used.

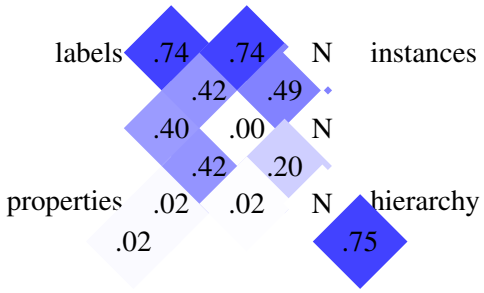
5.3.6 ctxMatch2 (2004)

CtxMatch2 [Bouquet *et al.*, 2003b; 2003a; 2006] uses the semantic matching approach. It translates the ontology matching problem into the logical validity problem and computes logical relations, such as equivalence, subsumption between concepts and properties. CtxMatch2 is a sequential system. At the element level it uses only WordNet to find initial matches for classes as well as for properties. At the structure level, it exploits description logic reasoners, such as Pellet[4] or FaCT[5] to compute the final alignment.



5.3.7 CMS (2004)

The CROSI Mapping System (hereafter, CMS) has been developed in the context of the CROSI project (which stands for Capturing Representing and Operationalising Semantic Iinteroperability). It has been evaluated at an early stage in the OAEI 2005 campaign. CMS is a structure matching system that capitalizes on the rich semantics of the OWL constructs found in source ontologies and on its modular architecture that allows the system to consult external linguistic resources.



The modular architecture of CMS employs a multi-strategy system comprising of four modules, namely, Feature Generation, Feature Selection and Processing, Aggregator and Evaluator. In this system, different features of the input data are generated and selected to fire off different sorts of feature matchers. The resultant similarity values are compiled by multiple similarity aggregators running in parallel or consecutive order. The overall similarity is then evaluated to initiate iterations that backtrack to different stages.

CMS uses various families of algorithms in order to compute similarity based on string distance. Sometimes, natural language processing methods are employed to cut down the number of string tokens that need to compute the similarity for. In particular, CMS uses tools that exploit synonymy at the: (1) lexical-level, e.g. the use of WordNet to provide a source of synonyms,

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

[5] http://www.cs.man.ac.uk/~horrocks/FaCT

hypernyms and hyponyms; the (2) phrase- and sentence-level, e.g. phrases and sentences in the active and passive forms but having the same meanings; and the (3) semantic-level synonymy. Structural information is exploited through a sophisticated structure traversing algorithm that collects evidence of similarities along its path. Finally, CMS uses the ontology structure in different ways. Heuristic rules is the most common way to take structures into account, e.g. identifying similarity of two entities based on the status of their parents and siblings.

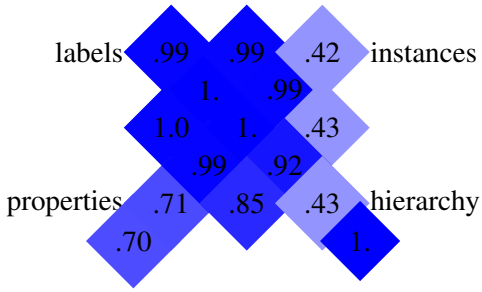
5.3.8 Falcon-AO (2006)

Falcon-AO is a system for matching OWL ontologies. It is made of two components:

LMO is a linguistic matcher. It associates with each ontology entity a bag of words which is built from the entity label, the entity annotations as well as the labels of connected entities. The similarity between entities is based on a TF/IDF computation [Qu *et al.*, 2006].

GMO is a bipartite graph matcher [Hu *et al.*, 2005]. It starts by considering the RDF representation of the ontologies as a bipartite graph which is represented by its adjacency matrix (A and A'). The distance between the ontologies is represented by a distance matrix (X) and the distance (or update) equations between two entities are simply a linear combination of all entities they are adjacent to (i.e., $X^{t+1} = AX^tA'^T + A^TX^tA'$). This process can be bootstraped with an initial distance matrix and the real process is more complex than described here because it distinguishes between external and internal entities as well as between classes, relations and instances.

These two components are used in sequence: First LMO is used for assessing the similarity between ontology entities on the basis of their name and text annotations. If the result has a high confidence, then it is directly returned for extracting an alignment. Otherwise, the result is used as input for the GMO matcher which tries to find an alignment on the basis of the relationships between entities [Jian *et al.*, 2005].

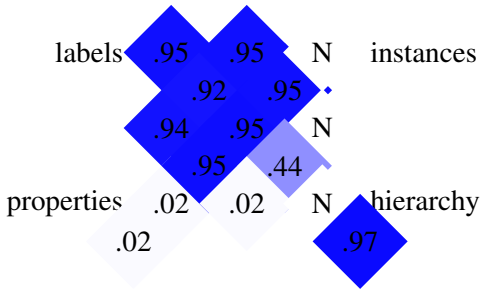


Falcon also participated in 2005, but we retained the last figures, i.e., those of 2006.

5.3.9 H-Match (2006)

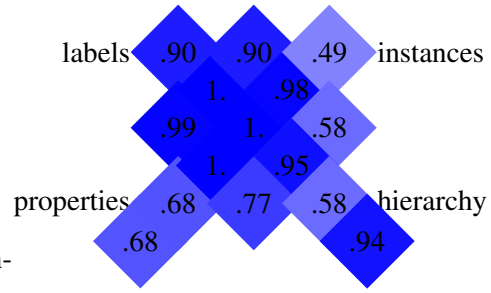
H-Match [Castano *et al.*, 2006] is an automated ontology matching system. It was designed to enable knowledge discovery and sharing in the settings of open networked systems, in particular within the Helios peer-to-peer framework [Castano *et al.*, 2005]. The system handles ontologies specified in OWL. Internally, these are encoded as graphs using the H-model representation [Castano *et al.*, 2005]. H-Match inputs two ontologies and outputs (one-to-one or one-to-many) correspondences between concepts of these ontologies with the same or closest intended meaning. The approach is based on a similarity analysis through affinity metrics, e.g., term to term affinity, datatype compatibility, and thresholds. H-Match computes two types of affinities (in the $[0\ 1]$ range), namely *linguistic* and *contextual* affinity. These are then combined by using weighting schemas, thus yielding a final measure, called *semantic* affinity. Linguistic affinity builds on

top of a thesaurus-based approach of the Artemis system. In particular, it extends the Artemis approach (i) by building a common thesaurus involving such relations among WordNet synsets as meronymy or coordinate terms, and (ii) by providing an automatic handler of compound terms (i.e., those composed by more than one token) that are not available from WordNet. Contextual affinity requires consideration of the neighbour concepts, e.g., linked via taxonomical or mereological relations, of the actual concept.



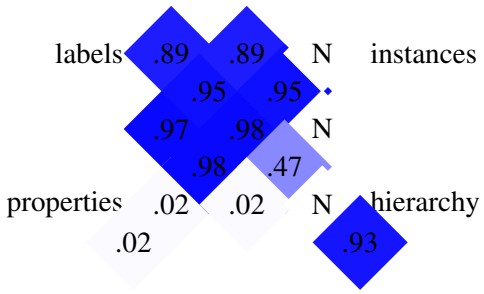
One of the major characteristics of H-Match is that it can be dynamically configured for adaptation to a particular matching task. Notice that in dynamic settings complexity of a matching task is not known in advance. This is achieved by means of four matching models. These are: *surface*, *shallow*,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

5.3.10 Coma (2006)



COMA (COmbination of MAtching algorithms) [Do and Rahm, 2002b] is a schema matching tool based on parallel composition of matchers. It provides an extensible library of matching algorithms, a framework for combining obtained results, and a platform for the evaluation of the effectiveness of the different matchers. As in [Do and Rahm, 2002b], COMA contains 6 elementary matchers, 5 hybrid matchers, and one reuse-oriented matcher. Most of them implement string-based techniques, such as affix, n-gram, edit distance; others share techniques with Cupid (thesauri look-up, etc.). Reuse-oriented is an original matcher, which tries to reuse previously obtained results for entire new schemas or for its fragments. Schemas are internally encoded as directed acyclic graphs, where elements are the paths. This aims at capturing contexts in which the elements occur. Distinct features of the COMA tool in respect to Cupid are a more flexible architecture and the possibility of performing iterations in the matching process. It presumes interaction with users who approve obtained matches and mismatches to gradually refine and improve the accuracy of match. COMA++ is built on top of COMA by elaborating in more detail the alignment reuse operation, provides a more efficient implementation of the COMA algorithms and a graphical user interface.

5.3.11 OWL CtxMatch (OCM, 2006)

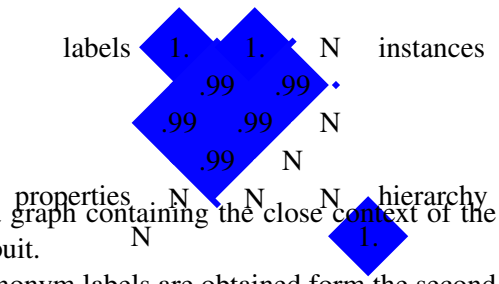


The OWL CtxMatch algorithm has been designed to match OWL DL ontologies. It is an OWL specialized version of the CtxMatch algorithm (like ctxMatch2), which is designed as a general algorithm to discover semantic relationships across distinct and autonomous generic structures. The main requirement it imposes on the structures being matched is the necessity of labelling them with natural language labels. The unique feature that both algorithms offer is that they perform so called "semantic matching" and as a result are able to recognize a broad range of relationships between matched entities, i.e. not only equivalence but also subsumption, disjointness and intersection. It assumes that the entities of given ontologies are named with commonly used words, not with some unintelligible symbols or randomly generated texts. OWL CtxMatch has already been used in one of the latest OWL-S matchmakers called Cobra1Matchmaker.

OWL CtxMatch similarly to CtxMatch realizes matching as a series of computations of relationships in which each computation is performed for every single pair of unfamiliar entities coming from both given structures. Each mapping is computed in two steps. At first internal representations of both entities are built, by means of which the algorithm stores recognized interpretations. These interpretations are defined in the form of appropriate logical formulas. In OWL CtxMatch there have been proposed description logics formulas, which are more expressive than propositional logics formulas used in original CtxMatch. The second step finds single mapping by computing what relationship holds between particular entities on the basis of their internal representations. Since OWL CtxMatch uses description logics formulas, this step in practice is realized by an external DL reasoner that initially merges both sets of formulas into one model, performs classification of it and finally determines what kind of relationship holds between counterparts for the particular entities.

The current version of the algorithm uses solely WordNet dictionary as a source of both lexical and domain knowledge and therefore is limited to the English language only. It uses Pellet as an OWL reasonner.

5.3.12 DSSim (2006)



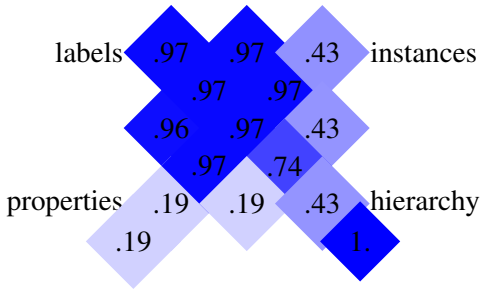
DSSim aims at producing alignments dynamically at runtime. Though, its goal is to be monitored through human interaction, the DSSim algorithm is an iterative closed loop which creates the mapping without any human interaction and works as follows:

1. For each ontology entity of the first ontology, a graph containing the close context of the entity, such as the concept and its properties, is buit.
2. Syntactically similar entities or those bearing synonym labels are obtained form the second ontology and a graph containing them is built.
3. The similarity between nodes is assessed through different similarity algorithms and their output is combined using the Dempster rule of combination.

4. The similiarity pairs bearing the highest belief function are selected and added to the alignment.

In order to avoid a complex graph of relationships, a limit on the number of synonyms, which are extracted from the WordNet, is defined.

5.3.13 Automs (2006)



Automs is the successor of the HCONE-merge system. HCONE-merge is an approach to domain ontology matching and merging exploiting different levels of interaction with a user [Kotis *et al.*, 2006; Vouros and Kotis, 2005; Kotis and Vouros, 2004]. First, an alignment between the two input ontologies is computed with the help of WordNet. Then, the alignment is processed straightforwardly by using some merging rules, e.g., renaming, into the new merged ontology. The HCONE basic matching algorithm works in six steps:

1. Chose a concept from one ontology.
2. Retrieve the WordNet senses of this concept;
3. Retrieve hypernyms and hyponyms of these senses.
4. Exaluate for the most frequent terms within the vicinity (senses, hponyms and herperonyms) their frequency of occurence.
5. Build a query from the related concepts related to the initial concept in the input ontology.
6. Use Latent Semantic Indexing for determining the best sense to be used in the query.

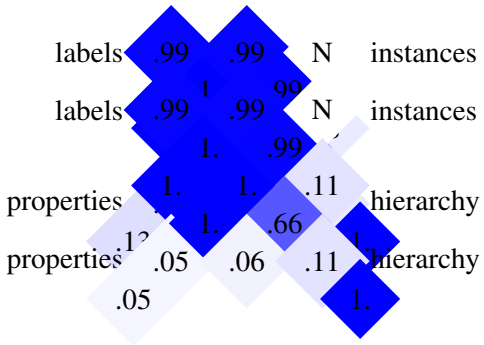
The highest graded sense expresses the most possible meaning for the concept under consideration. Finally, the relationship between concepts is computed. For instance, equivalence between two concepts holds if the same WordNet sense has been chosen for those concepts based on six steps procedure described above. The subsumption relation is computed between two concepts if a hypernym relation holds between the corresponding WordNet senses of these concepts. Based on the level at which the user is involved in the matching process, HCONE provides three algorithms to ontology matching. These are: fully automated, semi-automated and user-based. The user is involved in order to provide feedback on what is to be the correct WordNet sense on a one by one basis (user-based), or only in some limited number of cases, by exploiting some heuristics (semi-automated).

5.3.14 Onto-Mapology (2006)

Onto-Mapology is the Johns Hopkins University Applied Physics Lab (JHU/APL) ontology mapping software solution that was designed and developed with strong consideration for human participation in the mapping process. It integrates techniques based on string and text matching, graph structure matching, and rule-based semantic matching. It allows users to apply different combinations of these techniques, or a hybrid algorithm. This system is dedicated to OWL ontologies.

String matching is based on off-the-shefves solutions and in particular, Jaro-Winkler similarity, 2-gram based simiilarities and an information retrival indexing tool. Structural matching then compare the ontology entities in function of their neighbours matching or not (starting form the string matching results). If two entities have a large proportion of neighbours matchings, then they are considered matching. Finally, Onto-Mapology expresses matching rules that are applied to the current match in order to improve it (generally by providing more matches). This is called semantic matching.

5.3.15 Prior (2006)



The Prior system generalises the notion of virtual document used in Falcon-AO In Prior, the profile of a concept is a combination of all linguistic information of the concept, i.e. the profile of a concept = the concept's name + label + comment + property restriction + other descriptive information. The Profile Enrichment is a process of using a profile to represent a concept in the ontology, and thus enrich its information. The purpose of profile enrichment is based on the observation that though a name is always used to represent a concept, sometimes the information carried in a name is restricted. While, other descriptive information such as comments may contain words that better convey the meaning of the concept.

Profile Propagation exploits the neighboring information of each concept. The profile information is spread to ancestors, children or siblings of the ontology entities. The propagation is associated with weights which follows two rules: (1) The closer the concepts, the higher the weights, and (2) The weight of a parent is higher than the weight of a child and the weight of a child is higher than the weight of a sibling.

From these profiles, Prior uses cosine similarity in the vector space associated to profiles (as bag of words). Simultaneously, the String Mapper computes the similarity between the names of different concepts using Levenshtein distance. Both similarity are combined. Alternatively, when ontologies are too large, Prior uses an information retrieval system (Indri) each entity of each ontology are considered as queries against the other ontology and the top-ten answers are preserved. A Mapping Extractor extracts matched entities from the candidate matched entities and outputs the results.

5.3.16 RiMoM (2006)

RiMOM is a tool for ontology alignment by combining different strategies. Each strategy is defined based on one kind of information or one type of approach. In its current version, there are five strategies: edit-distance based strategy, statistical-learning based strategy, and similarity-propagation strategies.

There are six major steps in the alignment process of RiMOM:

1. Similarity factors estimation. Given two ontologies, it estimates two similarity factors, which respectively approximately represent the structure similarity and the label similarity of the two ontologies. The two factors are used in the next step of strategy selection.

2. Strategy selection. The basic idea of strategy selection is if two ontologies have high label similarity factor, then RiMOM will rely more on linguistic based strategies; while if the two ontologies have high structure similarity factor, then we will employ similarity-propagation based strategies on them.
3. Strategy execution. The selected strategies are used independently to find the alignments. Each strategy outputs an alignment.
4. Alignment combination. The alignment are combined through a linear-interpolation method.
5. Similarity propagation. If the two ontologies have high structure similarity factor, RiMOM employs an algorithm called similarity propagation to refine the found alignments and to find new alignments that can not be found using the other strategies. Similarity propagation makes use of structure information.
6. Alignment refinement. The resulting alignments are improved through several heuristic rules to remove the "unbelievable" alignments.

RiMOM offers various possible strategies such as linguistic based strategies (edit-distance and K-Nearest Neighbor statistical-learning strategies) and structural propagation strategies (concept-to-concept propagation strategy (CCP), property-to-property propagation strategy (PPP), and concept-to-property propagation strategy (CPP)).

Depending on their label and structure similarity factors, the algorithm will favour one or the other king of strategy. For this purposes it uses heuristic rules. For example, if the structure similarity factor is lower than some threshold (.25) then RiMOM does not use the CCP and PPP strategies. However, the CPP will always be used in the alignment process.

These characterisation of the evaluated algorithms can be particularly useful because they are very precise on what kaind of input the algorithms can handle well (or relatively well). It is clear that most algorithms are able to do very well when the labels remain unaltered, then when labels are modified – which is most of the real matching tasks – algorithms perform more or less correctly. We will see later how to take advantage of this.

5.4 Summary

In summary, these evaluations are very complementary: an application developer should first select the systems that can handle its data (input and output types). This can be achieved from the literature survey or the result of the questionnaire. Then it should find the best performer depending of her needs and the characteristics of her data. This can be achieved by looking closely at the cell that corresponds to the kind of data that has to be used by the system.

Most of the criteria are not comparable for various reasons. So it is difficult to compare the information gathered with the three techniques above. Moreover, information coming from these three sources can be useful for choosing a system. It is thus necessary to combine them when matching them with the application requirements.

This is considered in the next chapter.

Chapter 6

Finding suitable matchers

When the matcher profiles have been obtained and the application characteristics are known, it is necessary to match them in order to decide which matcher to use. This can be done through various methods. The difficulty comes from the large amount of non comparable characteristics to take into account.

In this chapter we provide two methods for supporting the decision. The first one (Section 6.1) is a very simple method that is based on the characterization of applications in large classes like in Chapter 2. The second one is a more elaborated framework that uses all the characteristics provided in Chapter 3 and offers a method for multi-criteria decision making called Analytic Hierarchy Process (AHP)(cf. Section 6.2) applied to the matching selection process (cf. Section 6.3).

6.1 Balancing criteria by aggregating measures

[Ehrig, 2006] provided an analysis of the different needs for evaluation depending of the considered applications. We have applied his technique to the requirement table (Table 2.1) of Chapter 2 [Euzenat and Shvaiko, 2007]. As a matter of fact, it can be rewritten in function of the measurements obtainable by evaluating the matchers. We used this technique to design Table 6.1. Therefore, different *application profiles* could be established to explicitly compare matching algorithms with respect to certain tasks.

Such a table can be useful for aggregating the measures corresponding to each of these aspects with different weights or to have an ordered way to interpret evaluation results.

Different measures suit different evaluation goals. If we want to improve systems, it is best to have as many indicators as possible. However, in order to single out the best system, it is generally better to focus on the very relevant factors. This can be achieved by only selecting the few very relevant factors, e.g., speed and precision for semantic web browsing, or by aggregating measures in relation with their relevance.

For aggregating measures depending on a particular application, its is possible to use weights corresponding to the values of Table 6.1, and thus respecting the importance of each factor. Weighted aggregation measures (weighted sum, product or average) can be used.

F-measure is already an aggregation of precision and recall. It can be generalised as a harmonic mean, for any number of measures. This requires to assign every measurement a weight, such that these weights sum to 1. Obviously the weights have to be chosen carefully, again depending on the goal.

Application	speed	automatic	precision	recall
Ontology evolution	medium	low	high	high
Schema integration	low	low	high	high
Catalog integration	low	low	high	high
Data integration	low	low	high	high
P2P information sharing	high	low	medium	medium
Web service composition	high	high	high	low
Multi agent communication	high	high	high	medium
Context matching in ambient computing	high	high	high	medium
Query answering	high	medium	medium	high
Semantic web browsing	high	medium	high	low

Table 6.1: Application requirements of Table 2.1 reinterpreted as measurement weights.

Definition 1 (Weighted harmonic mean)**.** *Given a reference alignment R, a set of measures $(M_i)_{i\in I}$ provided with a set of weights $(w_i)_{i\in I}$ between 0 and 1 such that their sum is 1, the weighted harmonic mean of some alignment A is given by*

$$H(A,R) = \frac{\prod_{i\in I} M_i(A,R)}{\sum_{i\in I} w_i \cdot M_i(A,R)}.$$

Example 1 (Weighted aggregation of evaluation measures)**.** *Consider that we need to choose among two available systems S_1 and S_2, for an application to peer-to-peer system, semantic web browsing or schema integration. We will apply weights corresponding to the criteria of Table 6.1. The weights will be $high = 5$, $medium = 3$ and $low = 1$. They are normalised (so as to sum to 1.) for each kind of application. The performance of the systems with regard to the criteria are given in the following table as well as the resulting harmonic means for each pair system/application:*

					S_1	S_2
				Automatic	*1.*	*1.*
				Speed	*.6*	*.7*
				Recall	*.6*	*.8*
	Automatic	*Speed*	*Recall*	*Precision*	*.6*	*.5*
Peer-to-peer system	*.08*	*.42*	*.25*	*.25*	*.62*	*.67*
Semantic web browsing	*.3*	*.1*	*.1*	*.5*	*.68*	*.64*
Schema integration	*.08*	*.08*	*.42*	*.42*	*.62*	*.64*

Those who need a matching system for a peer-to-peer system or schema integration should choose system S_2 ($.67$ and $.64$ are better than $.62$) and those who want to use it for semantic web browsing should use system S_1 ($.68$ is better than $.64$).

6.2 A method to identify suitable matching approaches

As long as decisions rely on single criterion that serves as the basis for comparison of alternatives or the scales of the different criteria are consistent and numeric measures accurately capture expected performance, summary statistics or, in some cases, just acting on the human instinct may be sufficient for the decision making process. However, when the decision depends on multiple criteria and scales are not consistent the process becomes complex and difficult, and the involvement of qualitative as well as quantitative methodologies or tools is indispensable. Consequently, in such cases a multi criteria decision making process is required, otherwise known as a *Multi Criteria Decision Analysis (MCDA)*, which is a procedure that aims to support decision makers whose problems are concerned with numerous and conflicting criteria. Such methods developed for better model decision scenarios vary in their mathematical rigor, validity, and design [Grandzol, 2005]. One of such methods, a methodology for supporting a decision making process called *Analytic Hierarchy Process (AHP)* takes into account the considerations of Hahn [Hahn, 2002] regarding the need for a structured results-based approach for decision making that allows trade-offs into the systematic method, including all perspectives and considerations.

The AHP is a systematic approach developed to structure the expectance, intuition, and heuristics based decision making into a well-defined methodology on the basis of sound mathematical principles [Bhushan and Rai, 2004].

AHP provides a mathematically rigorous application and proven process for prioritization and decision-making which helps setting priorities and to making the best decision when both qualitative and quantitative aspects of a decision need to be considered [Saatly, 1990]. By reducing complex decisions to a series of pair-wise comparisons and then synthesizing the results, decision-makers arrive at the best decision based on a clear rationale. It is generally accepted, that AHP constitutes one of the best options to aid multi-criteria decision making[1]. It compares the relative importance that each criterion has with respect to the others, while enabling the relative weight of the criteria to be calculated. Finally it normalizes the weights in order to obtain the measures for the existing alternatives. The AHP-method consists of:

1. **defining the problem or the project objectives**: e.g. buying a car;

2. **building a hierarchy of decision:** AHP provides a means to break down the problem into a hierarchy of subproblems (hierarchy of goal, criteria, sub-criteria and alternatives) which can more easily be comprehended and subjectively evaluated [Bhushan and Rai, 2004]. At the root of the hierarchy is the goal (e.g. suitable car) or objectives of the problem in question, the leaf nodes are the alternatives (e.g. Mercedes, VW) which are to be compared and between these two levels are various criteria (c) and sub-criteria (sc) (e.g. c-car comfort: sc-air condition, sc-leather seat; c-car security: sc-ABS, sc-airbag and c-car body design).

3. **collecting data**; data is collected from domain experts corresponding to the hierarchy in the pairwise comparison of the alternatives on a qualitative scale. This step assesses the characteristics of each alternative (e.g. Alternative 1 (Mercedes) is much better than Alternative 2 (VW) w.r.t leather seats, airbag and car body design but Alternative 2 (VW) is better than Alternative 1 (Mercedes) considering ABS and air condition).

[1]It does not use the normalized groups of separate numbers which destroy the linear relationship among them [Fenton and Pfleeger, 1996]

4. **building a pairwise comparison:** for each level of criteria (sub-criteria and criteria) a pairwise comparison between the sibling nodes is to be built and organized into square matrix[2] (e.g. car security is much more important than car body design and more important than car comfort while car comfort is only a little bit more important than car body design).

5. **calculating the final result:** the ratings of each alternative (cf. step 3) is multiplied by the weight of the sub-criteria (cf. step 4) and aggregated to get local ratings with respect to each criterion. The local ratings are then multiplied by the weights of the criteria (cf. step 4) and aggregated to the global ratings. The final value is used to make a decision about the problem defined in the step 1.

6.3 Applying analytic hierarchy process to matcher selection

AHP is to be applied to the selection of matching approaches. By reducing complex decisions, i.e. which matching is suitable for a given set of requirements, to a series of pair-wise comparisons (dimensions, factors and attributes) and synthesizing the results (list of possible algorithms) decision-makers arrive at the best decision (the best matching approach) based on a clear rationale [Saatly, 1990].

In the following we give a brief overview of how the AHP steps described in the Section 6.2 can be applied to the process of matcher selection taking into account some tool support for the data collection and calculation of the best alternative.

1. The problem to be solved: "Which matching approach is currently relevant w.r.t the given application requirements?"

2. The hierarchy of decision is built using the taxonomy described in Section 3 whereby the goal is to "find a suitable approach" (level 0) which is connected though three levels of criteria: 1^{st}level - dimensions, 2^{nd} level - factors and 3^{rd} level - attributes with the alternative matching approaches (cf. Figure 6.1).

3. In order to collect data about the different alternatives of matching approaches and to be able to conduct the pairwise comparisons we firstly need the relevant information about the particular alternatives. For this reason we have developed (following the hierarchy of the matching characteristic) an online questionnaire (filled out by the domain and matching experts) that allows the addition and rating (by usage of a predefined scale from 0 to 8) of matching alternatives. When a matcher is added via the questionnaire into the collection of the alternatives, all available alternatives in the system are automatically weighted against the new approach. Given two matcher alternatives m_1, m_2 and criteria c as well as the user defined weighings for the single approach $w(c)_{m_1}$ and $w(c)_{m_2}$ the weighings for the pairwise comparisons (between alternatives m_1, m_2) $w(c)_{m_1,m_2}$ and $w(c)_{m_2,m_1}$ are calculated as follows: (i) $w(c)_{m_1,m_2} = w(c)_{m_1} - w(c)_{m_2}$; (ii) $w(c)_{m_2,m_1} = w(c)_{m_2} - w(c)_{m_1}$.

4. Since the users of the AHP-tool have defined the requirements of their application w.r.t the suitable matching approach, they are able to weight the criteria (dimensions, factors and attributes) in the pairwise comparison on the scale from 0 - equal (two criteria have the same importance) to 8 - extremely important (one criteria is much more relevant than the

[2]For details see [Saatly, 1990]

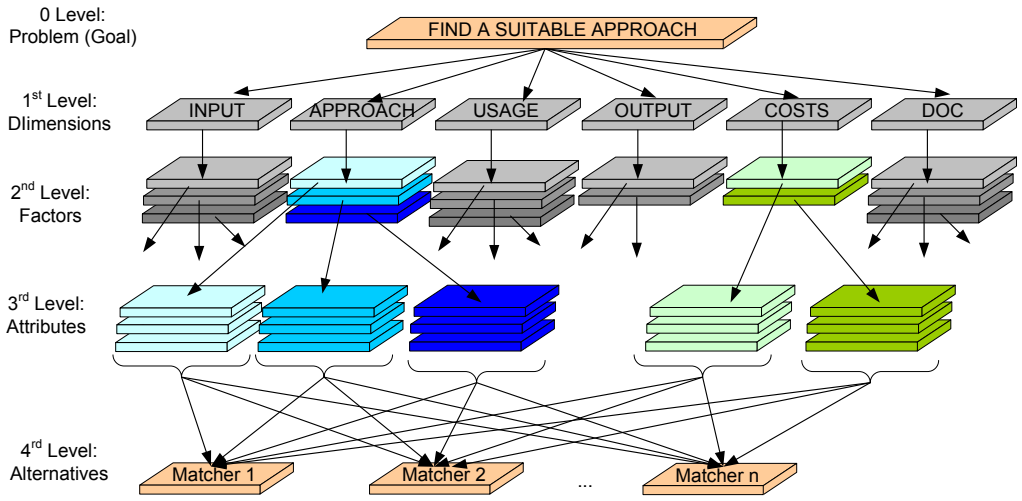


Figure 6.1: AHP hierarchy (Detection of the suitable matching approach)

other) concerning their system specification. This means, that for each level of criteria the users build a pairwise comparison between the sibling nodes: they weight the attributes against attributes, factors against factors and dimensions against dimensions (e.g. within the factor *formality level* the attribute *formal* (ontologies) is more important than *informal*, cf. Figure 6.3). If there is more then one level of criteria (as in our case), for each sub-criteria on the lowest level an "aggregated judgement" must be intended. The aggregated judgement in the hierarchy of the decision problem along all possible paths are multiplied with another from the highest to the lowest level.

Since the weighings of the criteria are the crucial part in the AHP approach they need to be deeply considered and demand the detailed analysis of the application requirements. Even if the complexity of the weighings increase with each level of criteria and it is especially challenging to rate the general dimensions, in our opinion the approach gives a relative easy to use method to find the suitable matcher by defining and rating the application requirements.

5. The decision regarding the determination of the suitable matching approach defined in the step 1 is based on the ranking $r(goal)$ of a matcher alternative m. The ranking is based on the overall priority for each alternative that can be better understood if we think of the priority for each criterion as a weight that reflects its importance. The overall priority is found by multiplying the products of the criterion priority with the priority of its decision alternative. This means that the ranking reflects the global importance of the approach according to the alternative weightings performed in step 3 as well as criteria weightings from step 4 and is calculated as follows:

$$c_{crit} = \{n | n \text{ child of } crit\}$$

$$r(crit) = \begin{cases} |\text{getWeight}(m, crit)|, & \text{if crit is at lowest hierarchy level} \\ \sum_{n \in c_{crit}} r(n) \cdot |\text{getWeight}(m, n)|, & \text{otherwise} \end{cases}$$

The higher a matcher alterative m is weighted for various criteria, with each criteria weighted with respect to the users requirements, the higher the priority of the particular approach in the entire ranking. Following this weighting process the AHP tool supports the creation of a ranking of the alternatives in depending upon the multilevel hierarchy matcher characteristics, weightings of these characteristics as well as weightings of the alternatives that shows the priority of each alternative for the defined goal.

6.4 Implementation

In step 3, PHPSurveyor[3] has been used for providing the online questionnaire[4]. The collected data regarding the matcher alternatives from the questionnaire is stored in a questionnaire database (MySQL) while an additional database (AHP database) stores the weighting results of the pairwise comparisons (cf. Figure 6.2).

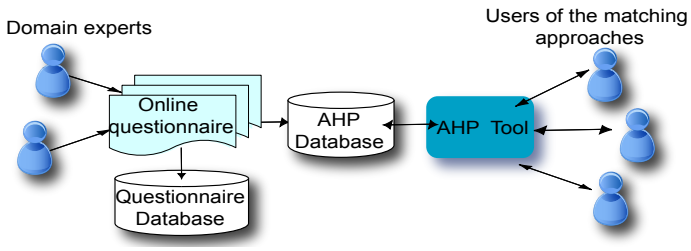


Figure 6.2: AHP Tool with online questionnaire

In step 4, to enable a user-friendly pairwise comparison of the criteria from the multilevel hierarchy matcher characteristic we developed a tool which supports the processing of the AHP method[5].

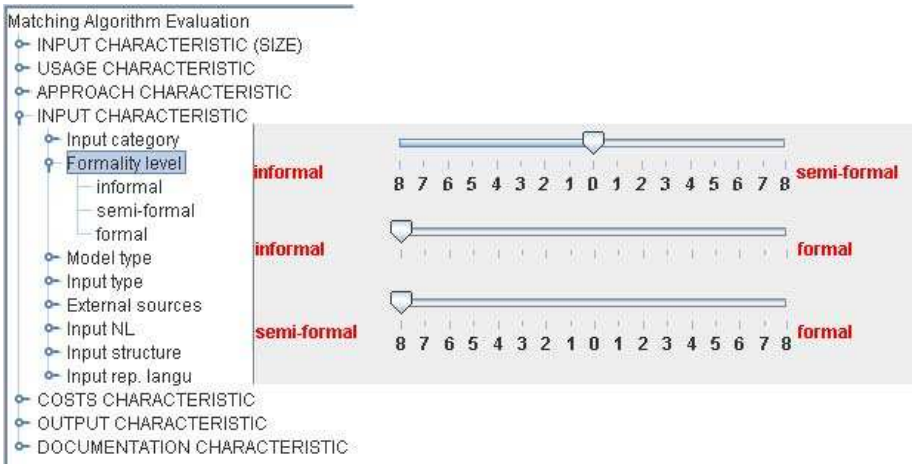


Figure 6.3: AHP tool: weighed attributes

[3] http://www.phpsurveyor.org
[4] http://matching.ag-nbi.de/
[5] AHP tool is a modification of the Java AHP tool JAHP; [6]

6.5 Summary

We have presented two approaches for matching matchers to applications. The first approach is very simple but only requires few information like the one provided in Chapter 2) to provide results. The second approach if more elaborate but requires much more information, especially in ranking the characteristics.

The proposed adaption of the Analytic Hierarchy Process (AHP) to the detection of a suitable matching approach has been based on *multilevel characteristics for matching approaches* and supported by the AHP tool. We hope that by implementing tools for helping domain experts with poor expertise in ontology matching field to use the AHP, we will contribute help accurate description of the application requirements and finding appropriate approach for matching.

We will find the results in Chapter 7.

Chapter 7

Recommendations

Recommending a particular system for some application seems now possible. We apply below the techniques that have been proposed before. As mentioned, these techniques will provide results at different levels of granularity. They will thus be applied in an increasing precision order. The first results are very general since they apply to classes of applications (§7.1.1). We will thus apply them to Knowledge web use cases in order to apply more suited systems to use cases (§7.1.2). The second results use the Analytic Hierarchy Process to decide which matcher is the best for Knowledge web use case 1: Recruitments.

7.1 General view of systems and use cases

This section is a straightforward application of the techniques provided above to the results of systems evaluated during the OAEI campaigns. The results provided here should be taken as an illustration of the techniques rather than a definitive guide to matching systems. In particular, the results are only based on the system participating in OAEI, they do only take precision, recall and automatic into account and finally the weights of the measures are arbitrarily fixed.

We proceed in two steps: first we select systems for applications depending on application requirements and system measured performances. Then we apply the findings to Knowledge web use cases depending on their identified class of application.

7.1.1 Characterisation of system applicability

For each kind of application as considered in §6.1, we aim at evaluating matchers with regard to application requirements. Unfortunately, we have only few parameters that can be taken into account: automaticity requirement which is always satisfied by all the tested systems, precision and recall. We weight these three criteria as suggested in Example 1 in §6.1, i.e., $high = 5$, $medium = 3$, $low = 1$ and normalise them so that they sum to 1. Then, weighted harmonic means is computed from the performance results obtained through evaluation.

The result is provided in Table 7.1.

Unfortunately, either systems are very homogeneous and non specialised or these three criteria are not sufficient to introduce variability in choice. Indeed, the best system with regard to the measures is always the same: RiMOM.

Application	automatic	precision	recall	System
Ontology evolution	.09	.45	.45	RiMOM (.91), Falcon, Coma (.88)
Schema integration	.09	.45	.45	RiMOM (.91), Falcon, Coma (.88)
Catalog integration	.09	.45	.45	RiMOM (.91), Falcon, Coma (.88)
Data integration	.09	.45	.45	RiMOM (.91), Falcon, Coma (.88)
P2P information sharing	.14	.43	.43	RiMOM (.91), Coma (.88), Falcon (.87)
Web service composition	.45	.45	.09	RiMOM (.87), Falcon (.83), Coma (.82)
Multi agent communication	.38	.38	.23	RiMOM (.88), Falcon, Coma (.84)
Context match in ambient computing	.38	.38	.23	RiMOM (.88), Falcon, Coma (.84)
Query answering	.27	.27	.45	RiMOM (.90), Falcon, Coma (.87)
Semantic web browsing	.33	.56	.11	RiMOM (.88), Falcon (.84), Coma (.83)

Table 7.1: Advised systems given application requirements.

Use case	Type	System
Recruitments	Data integration	RiMOM (.91), Falcon, Coma (.88)
B2C marketplace for tourism	Query answering	RiMOM (.88), Falcon (.84), Coma (.83)
News aggregation	Semantic web browsing	RiMOM (.88), Falcon (.84), Coma (.83)
Real estate management	P2P information sharing	RiMOM (.91), Coma (.88), Falcon (.87)
Integrated access to biological data	Data integration	RiMOM (.91), Falcon, Coma (.88)

Table 7.2: Knowledge web use cases requiring matching systems and the advised systems according to Table 7.1

7.1.2 Application to use cases

These results can be applied directly to the Knowledge web use cases (Table 7.2). yielding first advice to system designers. Since our results are very similar, this analysis is not particularly specific. It certainly could be better if we had some measure of the speed at which these systems return results.

7.2 Application to use case 1

In this section we briefly describe how we have applied the AHP-based methodology for matcher selection described in Sec.6.3 to use case 1 situated in the human resource domain.

Since the goal of the AHP-based methodology has already been defined in §6.3 (goal: finding a suitable matching approach), the decision hierarchy regarding matcher suitability has been built (§3) and the information considering the matcher approaches has been collected during the OAEI 2006 Campaign (§5.2), the missing link in the process of matcher selection is the weightings of requirements regarding the human resources use case.

7.2.1 Pairwise comparison of the criteria

To identify the requirements of the human resources application we have analyzed the the given human resource ontology along with the restrictions regarding the suitable matching approaches[1] . In order to compare the application requirements, we first need to "translate" the relevant parts of the use case description (in the text above we have marked in bold important information) into the corresponding terms from the developed multilevel characteristic of matching approaches (cf. §3). The translation of some of the information is shown in Table 7.3

Description from human resources use case - free text	**MCMA-based description**
General information	
ontology	input characteristic: input category: ontology
domain relevant terms	input characteristic: input model type: domain ontology
domain specific knowledge	input characteristic: external sources: domain specific resources
highly formal	input characteristic: input formality level: formal ontology
OWL	input characteristic: input formality level: formal ontology
over 8.000 concepts	input characteristic: input size: number of concept: extra large input characteristic: input size: size of input: extra large
less than 100 properties	input characteristic: input size: number of relation: medium input characteristic: input size: number of relation: extra large
different properties	input characteristic: input structure: heterogeneous relations
does not contain any axioms	input characteristic: input size: number of axioms: no axioms
job postings with candidate profiles	input characteristic: input size: number of ontologies: two
data describing job applications and job offers	input characteristic: input size: number of instances: large input characteristic: input size: number of instances: extra large
automatically compare	approach characteristics: kind of similarity relation: black box paradigm
particular country	input characteristic: natural language: one natural language
job portal	usage characteristics: usage goal: internet-based use
particular applicant profile against a multitude of job openings	output characteristics: matching cardinality: 1:m

Table 7.3: HR-use case in free text and MCMA description

To define the requirements of the human resources application within context of the AHP-based methodology concerning the specification of the potential suitable matching approaches we must weight the particular properties (from the same level in the hierarchy) in the pairwise comparison on a scale from 0 to 8 (cf. Table 7.4).

Figure 7.1 shows the weighting of the attributes `local use`, `network use` and `internet-based use` within the factor `usage goal` which belongs to the dimension `usage characteristic`.

7.2.2 Results of the application of the AHP-based methodology

After weighing all the relevant (regarding the information extracted from the human resources test case description) dimensions, factors and attributes the AHP tool (cf. Sec 6.4) calculates the results and delivers the ranking of the most suitable matching approaches. The approaches which

[1] Unfortunately, the corresponding ontologies cannot be exemplified for confidentiality reasons

Importance	Definition	Explanation
0	equal importance	two criteria (c_1 & c_2) have the same importance
2	moderate importance	c_1 is weakly more relevant than c_2
4	strong importance	c_1 is strongly more relevant than c_2
6	very strong (demonstrated) importance	c_1 is very strongly more relevant than c_2
8	absolute importance	c_1 is absolutely more relevant than c_2
1,3,5,7	intermediate between values (for compromise between the values mentioned above)	to interpolate a compromise judgment because there is not good word to describe it

Table 7.4: AHP scale

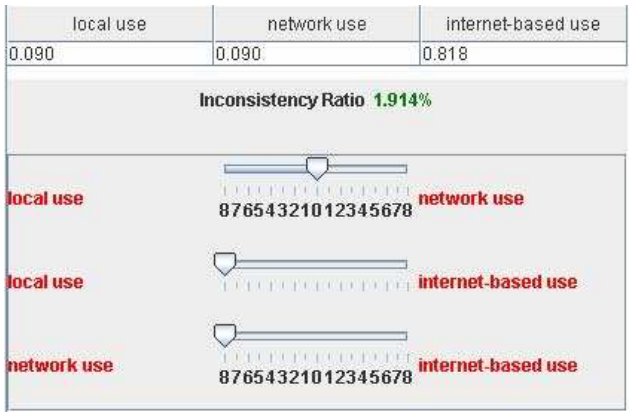


Figure 7.1: Weighting of the attributes

gained top ratings in the human resources test case are PRIOR, HMatch and Falcon, ranked 1^{st}, 2^{nd} and 3^{rd} respectively (cf. Figure 7.2).

Name	Priority	URL	Organization
PRIOR	0,166	http://www.sis.pitt.edu/~mingmao/om06/	university of pittsburgh
HMatch	0,147	http://islab.dico.unimi.it/hmatch/	Information System and Knowledg...
Falcon-AO	0,108	http://xobjects.seu.edu.cn/project/falcon/m...	Southeast University, China
RiMOM (Risk Min...	0,095	http://keg.cs.tsinghua.edu.cn/project/RiMOM	Knowledge Engineering Group, De...
AUTOMS	0,079	http://www.icsd.aegean.gr/kkot/AUTOMS/d...	University of the Aegean, Dept. of In...
AOAS - Anatomic...	0,078	n/a	* U.S. National Library of Medicine, ...
MAOM-QA Multi-A...	0,044		Open University, Knowledge Media ...
OWL-CtxMatch	0,035	unavailable	OWL-CtxMatch: S?awomir Niedha?

Figure 7.2: Results achieved by AHP-based mythology for the human resources use case

If we consider the terms defined in Table 7.3 and then compare the results obtained by the AHP-based methodology with the information on the diagrams from §5.2 it becomes obvious why just these approaches achieved such high rankings. For example with regard to the `input-related questions` concerning the dealing with different types of sources (cf. Figure 5.3) all three matchers are capable of handling, very well to extremely well, the ontological input type, and in addition, PRIOR is the best candidate to serve the extra large ontologies. Apart from that, concerning the `approach-related questions`, PRIOR, HMatch and Falcon have been developed to conduct automatic matching process based on the black-box paradigm (cf. Figure 5.5) while HMatch is also the best approach if internet-based usage of the approach is required.

7.3 Summary

In summary, the two evaluation procedures provide very different results. Indeed, these procedures manipulate totally different data in radically different ways. The first approach has only considered a limited set of parameters that can be relatively objectively evaluated, but used somewhat arbitrary ranking of these parameters. The second approach has used well-proved techniques for decision making on a wide range of parameters, but only questionnaire data on a limited set of systems as entry.

So, while the first approach is certainly more accurate in terms of the comparability of the data it provides and also its validity, the second approach is broader in the number of parameters it takes into account. This would certainly be improved by using the data of the first approach in the process of the second one.

Moreover, this shows that it remains difficult to evaluate the suitability of systems to application and application classes. Hence a really valuable evaluation has to go through application specific evaluation.

Chapter 8

Conclusions

Selecting an ontology matching system given a particular application is a difficult problem. Even when the application needs are very precise, there are many criteria that can be used for choosing the adequate matchers and all criteria cannot be assessed in the same way. It is also difficult to obtain all the information about each matcher (the information is most often fragmentary).

This deliverable can be seen as a first attempt at finding a way to solve these problems and providing the invaluable data for choosing a matcher. We summarise what has been done (§8.1) before providing a methodological guide for taking advantage of the findings (§8.2) and finally consider comparable work (§8.3).

8.1 Summary

In order to help identifying matching solutions to particular applications, we have performed the following steps:

- Identify relevant criteria for general categories of applications;
- Identify relevant characteristics of matchers;
- Explain how to benchmark matching systems in order to assess some of these characteristics;
- Provide profiles from a number of existing matching systems depending on the identified characteristics;
- Provide a method and present a tool for identifying suitable matching systems for an application;
- Finally, apply these techniques to Knowledge web use cases (and especially one of these).

8.2 Methodological guide

Given some application to develop, there can be several ways to use this deliverable. We characterise them as superficial, deep and involved. They are as follows:

superficial The superficial method would use the application analysis table (Table 6.1)for identifying the profile of the application and would use this either to select a subset of matchers on which to perform a deep analysis or to select the best suited matcher according to Chapter 7.

deep The deep analysis would use the tool presented in §6.2, input the detailed criterion preferences in order to find the matching system the closest to the requirements.

involved The involved approach would instrument the application in order to carry out application specific evaluation as presented in Chapter 4. This is a very costly approach however.

In the current state of the art, choosing a matching system cannot be done on a catalogue. This requires a precise analysis of the application requirement and a comparison with precise matcher characteristics. Both resources are expensive to produce.

Future work will be dedicated to the collection of further matcher alternatives (with help of the online questionnaire) and the application of the AHP tool into the various Semantic Web scenarios connected with the evaluation of the entire framework.

8.3 Related work

The closest work that we are aware of has been carried out within the INTEROP network of excellence [Huza *et al.*, 2006]. It aims at developing a system, OntoMas[1], for helping and teaching how to carry out matching. Since the tool aims at helping user to find an adequate matcher for some task, it has developed a classification of tools and a characterization of tasks. The tool classification based on [Shvaiko and Euzenat, 2005b] is comparable to the one presented here and the positioning of tools within this classification is assessed through questionnaires as well. The description of the task is more rudimentary (in particular, it is based only on the syntactic aspects of the input ontologies). This is certainly because the tool must be usable by novice users who do not know the task in depth. Contrary to the work presented here, the multicriteria decision is based on ad hoc rules. Some of these rules are filters that eliminate some unsuitable systems; some others increase or decrease the score of the considered methods (usually by one point). So this scheme is again equivalent to an equi-weighted linear aggregation.

As far as we can tell, no other work has considered proved multicriteria decision techniques for matching matchers to matching tasks.

[1] http://www.polytech.univ-nantes.fr/ontomas/

Bibliography

[An *et al.*, 2005a] Yuan An, Alexander Borgida, and John Mylopoulos. Constructing complex semantic mappings between XML data and ontologies. In *Proc. 4th International Semantic Web Conference (ISWC)*, volume 3729 of *Lecture notes in computer science*, pages 6–20, Galway (IE), 2005.

[An *et al.*, 2005b] Yuan An, Alexander Borgida, and John Mylopoulos. Inferring complex semantic mappings between relational tables and ontologies from simple correspondences. In *Proc. 4th International Conference on Ontologies, Databases and Applications of Semantics (ODBASE)*, volume 3761 of *Lecture notes in computer science*, pages 1152–1169, Agia Napa (CY), 2005.

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

[Berlin and Motro, 2002] Jacob Berlin and Amihai Motro. Database schema matching using machine learning with feature selection. In *Proc. 14th International Conference on Advanced Information Systems Engineering (CAiSE)*, volume 2348 of *Lecture notes in computer science*, pages 452–466, Toronto (CA), 2002.

[Bhushan and Rai, 2004] N. Bhushan and K. Rai, editors. *Strategic Decision Making: Applying the Analytic Hierarchy Process*. Springer, 2004.

[Bilke and Naumann, 2005] Alexander Bilke and Felix Naumann. Schema matching using duplicates. In *Proc. 21st International Conference on Data Engineering (ICDE)*, pages 69–80, Tokyo (JP), 2005.

[Bizer *et al.*, 2005] C. Bizer, R. Heese, M. Mochol, R. Oldakowski, R. Tolksdorf, and R. Eckstein. The Impact of Semantic Web Technologies on Job Recruitment Processes. In *Proc. of the 7th Internationale Tagung Wirtschaftsinformatik 2005*, pages 1367–1383, 2005.

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

[Bouquet *et al.*, 2003b] Paolo Bouquet, Luciano Serafini, and Stefano Zanobini. Semantic coordination: A new approach and an application. In *Proc. 2nd International Semantic Web Con-*

ference (ISWC), volume 2870 of *Lecture notes in computer science*, pages 130–145, Sanibel Island (FL US), October 2003.

[Bouquet *et al.*, 2006] Paolo Bouquet, Luciano Serafini, Stefano Zanobini, and Simone Sceffer. Bootstrapping semantics on the web: meaning elicitation from schemas. In *Proc. 15th International World Wide Web Conference (WWW)*, pages 505–512, Edinburgh (UK), 2006.

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

[Castano *et al.*, 2004] S. Castano, A. Ferrara, and S. Montanelli. Methods and Techniques for Ontology-based Semantic Interoperability in Networked Enterprise Contexts. In *Proc. of the 1st CAiSE INTEROP Workshop On Enterprise Modelling and Ontologies for Interoperability (EMOI - INTEROP 2004)*, pages 261–264, June 2004.

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

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

[Chang *et al.*, 2005] Kevin Chang, Bin He, and Zhen Zhang. Toward large scale integration: Building a metaquerier over databases on the web. In *Proc. 2nd Biennial Conference on Innovative Data Systems Research (CIDR)*, pages 44–55, Asilomar (CA US), 2005.

[Clements *et al.*, 2002] P. Clements, F. Bachman, L. Bass, D. Garlan, J. Ivers, R. Little, R. Nord, and J. Stafford, editors. *Documenting Software Architectures: Views and Beyond*. Addison-Wesley Professional, 2002.

[Clifton *et al.*, 1997] Chris Clifton, Ed Hausman, and Arnon Rosenthal. Experience with a combined approach to attribute matching across heterogeneous databases. In *Proc. 7th IFIP Conference on Database Semantics*, pages 428–453, Leysin (CH), 1997.

[Dhamankar *et al.*, 2004] Robin Dhamankar, Yoonkyong Lee, An-Hai Doan, Alon Halevy, and Pedro Domingos. iMAP: Discovering complex semantic matches between database schemas. In *Proc. 23rd International Conference on Management of Data (SIGMOD)*, pages 383–394, Paris (FR), 2004.

[Do and Rahm, 2002a] H. H. Do and E. Rahm. COMA—a system for flexible combination of schema matching approaches. In *Proc. of the 28th VLDB Conference*, 2002.

[Do and Rahm, 2002b] Hong-Hai Do and Erhard Rahm. COMA – a system for flexible combination of schema matching approaches. In *Proc. 28th International Conference on Very Large Data Bases (VLDB)*, pages 610–621, Hong Kong (CN), 2002.

[Do *et al.*, 2002] H. H. Do, S. Melnik, and E Rahm. Comparison of Schema Matching Evaluations. In *Proc. of GI-Workshop "Web and Databases"*, 2002.

[Doan *et al.*, 2001a] A. Doan, P. Domingos, and A. Halevy. Reconciling Schemas of disparate Data sources: A Machine Learning Approach. In *Proc. of the the SIGMOD01*, 2001.

[Doan *et al.*, 2001b] An-Hai Doan, Pedro Domingos, and Alon Halevy. Reconciling schemas of disparate data sources: A machine-learning approach. In *Proc. 20th International Conference on Management of Data (SIGMOD)*, pages 509–520, Santa Barbara (CA US), 2001.

[Doan *et al.*, 2004a] A. Doan, J. Madhavan, P. Domingos, and A. Halevy. Ontology Matching: A Machine Learning Approach. *Handbook on Ontologies*, pages 385–516, 2004.

[Doan *et al.*, 2004b] An-Hai Doan, Jayant Madhavan, Pedro Domingos, and Alon Halevy. Ontology matching: a machine learning approach. In Steffen Staab and Rudi Studer, editors, *Handbook of ontologies*, chapter 18, pages 385–404. Springer Verlag, Berlin (DE), 2004.

[Dogac *et al.*, 2002] A. Dogac, G. Laleci, Y. Kabak, and I. Cingil. Exploitingweb service semantics: Taxonomies vs. ontologies. *IEEE DATA ENGINEERING BULLETIN*, 4, 2002.

[Dou *et al.*, 2003] D. Dou, D. McDermott, and P. Qi. Ontology Translation on the Semantic Web. In *Proc. of the AInt'l Conf. on Ontologies, Databases and Applications of Semantics (ODBASE2003)*, pages 952–969, 2003.

[Dou *et al.*, 2005] Dejing Dou, Drew McDermott, and Peishen Qi. Ontology translation on the semantic web. *Journal on Data Semantics (JoDS)*, II:35–57, 2005.

[Ehrig and Staab, 2004] Marc Ehrig and Steffen Staab. QOM – quick ontology mapping. In *Proc. 3rd International Semantic Web Conference (ISWC)*, volume 3298 of *Lecture notes in computer science*, pages 683–697, Hiroshima (JP), 2004.

[Ehrig and Sure, 2004] Marc Ehrig and York Sure. Ontology mapping – an integrated approach. In *Proc. 1st European Semantic Web Symposium (ESWS)*, volume 3053 of *Lecture notes in computer science*, pages 76–91, Hersounisous (GR), May 2004.

[Ehrig *et al.*, 2005] Marc Ehrig, Steffen Staab, and York Sure. Bootstrapping ontology alignment methods with APFEL. In *Proc. 4th International Semantic Web Conference (ISWC)*, volume 3729 of *Lecture notes in computer science*, pages 186–200, Galway (IE), 2005.

[Ehrig, 2006] Marc Ehrig. *Ontology alignment: bridging the semantic gap*. PhD thesis, Universität Fridericiana zu Karlsruhe, Karlsruhe (DE), 2006.

[Embley *et al.*, 2004] David Embley, Li Xu, and Yihong Ding. Automatic direct and indirect schema mapping: Experiences and lessons learned. *ACM SIGMOD Record*, 33(4):14–19, 2004.

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

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

[Euzenat *et al.*, 2004a] Jérôme Euzenat, Thanh Le Bach, Jesús Barrasa, Paolo Bouquet, Jan De Bo, Rose Dieng-Kuntz, Marc Ehrig, Manfred Hauswirth, Mustafa Jarrar, Rubén Lara, Diana Maynard, Amedeo Napoli, Giorgos Stamou, Heiner Stuckenschmidt, Pavel Shvaiko, Sergio Tessaris, Sven Van Acker, and Ilya Zaihrayeu. State of the art on ontology alignment. Deliverable D2.2.3, Knowledge web NoE, 2004.

[Euzenat *et al.*, 2004b] Jérôme Euzenat, Marc Ehrig, and Raúl García Castro. Specification of a benchmarking methodology for alignment techniques. Deliverable D2.2.2, Knowledge web NoE, 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*, pages 69–87, Columbus (OH US), 1994.

[Fenton and Pfleeger, 1996] N. Fenton and L. Pfleeger, editors. *Software Metrics, A Rigorous & Practical Approach*. International Thomson Cmputer Press, 1996.

[Fürst and Trichet, 2005] F. Fürst and F. Trichet. Axiom-based ontology matching. In *Proc. of the 3rd international conference on Knowledge capture (K-CAP'05)*, pages 195–196, New York, NY, USA, 2005. ACM Press.

[Garbers *et al.*, 2006] J. Garbers, M. Niemann, and M. Mochol. A personalized hotel selection engine. In *Proc. of the Poster Session of 3rd ESWC 2006*, 2006.

[Giuchiglia and Shvaiko, 2004] F. Giuchiglia and P. Shvaiko. Semantic Matching. *Knowledge Web Review Journal*, pages 265–280, 2004.

[Giunchiglia and Shvaiko, 2003] Fausto Giunchiglia and Pavel Shvaiko. Semantic matching. *The Knowledge Engineering Review (KER)*, 18(3):265–280, 2003.

[Grandzol, 2005] J. R. Grandzol. Improving the faculty selection process in highe education: A case for the analytic hierarchy process. *Using Advenced Tools, Tehciques, and Methodlogies. Association for Institutional Research*, 6, 2005.

[Gruber, 1995] T. R. Gruber. Toward principles for the design of ontologies used for knowledge sharing. *Int. J. Hum.-Comput. Stud.*, 43(5-6):907–928, 1995.

[Guarino, 1998] N. Guarino. Formal ontology and information systems. In *Proc. of the 1st International Conference on Formal Ontologies in Information Systems (FOIS'98)*, pages 3–15, 1998.

[Haas *et al.*, 2005] Laura Haas, Mauricio Hernández, Howard Ho, Lucian Popa, and Mary Roth. Clio grows up: from research prototype to industrial tool. In *Proc. 24th International Conference on Management of Data (SIGMOD)*, pages 805–810, Baltimore (MD US), 2005.

[Hahn, 2002] E. D. Hahn. Better decisions come from a results-based approach. *Marketing News*, 36(36):22–24, 2002.

[He *et al.*, 2004] Hai He, Weiyi Meng, Clement Yu, and Zonghuan Wu. Automatic integration of web search interfaces with WISE-Integrator. *The VLDB Journal*, 13(3):256–273, 2004.

[He *et al.*, 2005] Hai He, Weiyi Meng, Clement Yu, and Zonghuan Wu. WISE-Integrator: A system for extracting and integrating complex web search interfaces of the deep web. In *Proc. 31st International Conference on Very Large Data Bases (VLDB)*, pages 1314–1317, Trondheim (NO), 2005.

[Hovy, 1998] Eduard Hovy. Combining and standardizing large-scale, practical ontologies for machine translation and other uses. In *Proc. 1st International Conference on Language Resources and Evaluation (LREC)*, pages 535–542, Granada (ES), 1998.

[Hu *et al.*, 2005] Wei Hu, Ningsheng Jian, Yuzhong Qu, and Qanbing Wang. GMO: A graph matching for ontologies. In *Proc. K-CAP Workshop on Integrating Ontologies*, pages 43–50, Banff (CA), 2005.

[Humphrey, 1999] W. S. Humphrey, editor. *Introduction to the Team Software Process*. Addison-Wesley Professional, 1999.

[Huza *et al.*, 2006] Mirella Huza, Mounira Harzallah, and Francky Trichet. Ontomas: a tutoring system dedicated to ontology matching. In *Proc. ISWC workshop on ontology matching*, pages 228–323, Athens (GA US), 2006.

[Ichise *et al.*, 2004] Ryutaro Ichise, Masahiro Hamasaki, and Hideaki Takeda. Discovering relationships among catalogs. In *Proc. 7th International Conference on Discovery Science*, volume 3245 of *Lecture notes in computer science*, pages 371–379, Padova (IT), 2004.

[Jian *et al.*, 2005] Ningsheng Jian, Wei Hu, Gong Cheng, and Yuzhong Qu. Falcon-AO: Aligning ontologies with falcon. In *Proc. K-CAP Workshop on Integrating Ontologies*, pages 87–93, Banff (CA), 2005.

[Kalfoglou and Schorlemmer, 2003a] Yannis Kalfoglou and Marco Schorlemmer. IF-Map: an ontology mapping method based on information flow theory. *Journal on Data Semantics (JoDS)*, I:98–127, 2003.

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

[Kang and Naughton, 2003] Jaewoo Kang and Jeffrey Naughton. On schema matching with opaque column names and data values. In *Proc. 22nd International Conference on Management of Data (SIGMOD)*, pages 205–216, San Diego (CA US), 2003.

[Kim *et al.*, 2005] Jaehong Kim, Minsk Jang, Young-Guk Ha, Joo-Chan Sohn, and Sang-Jo Lee. MoA: OWL ontology merging and alignment tool for the semantic web. In *Proc. 18th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems (IEA/AIE)*, volume 3533 of *Lecture notes in computer science*, pages 722–731, Bari (IT), 2005.

[Kotis and Vouros, 2004] Konstantinos Kotis and George Vouros. HCONE approach to ontology merging. In *Proc. 1st European Semantic Web Symposium (ESWS)*, volume 3053 of *Lecture notes in computer science*, pages 137–151, Hersounisous (GR), 2004.

[Kotis *et al.*, 2006] Konstantinos Kotis, George Vouros, and Konstantinos Stergiou. Towards automatic merging of domain ontologies: The HCONE-merge approach. *Journal of Web Semantics (JWS)*, 4(1):60–79, 2006.

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

[Lee *et al.*, 2002] Mong Li Lee, Liang Huai Yang, Wynne Hsu, and Xia Yang. XClust: clustering XML schemas for effective integration. In *Proc. 11th International Conference on Information and Knowledge Management (CIKM)*, pages 292–299, McLean (VA US), 2002.

[Léger *et al.*, 2005] Alain Léger, Lyndon Nixon, cois Paulus Fran Laurent Roquet, Malgorzata Mochol, Yannis Kompatsiaris, Vasileios Papastathis, Stamatia Drosopoulou, Mustafa Jarrar, Roberta Cuel, and Matteo Bonifacio. System and knowledge technology components for prototypical applications and business cases. Deliverable D1.1.4, Knowledge web NoE, 2005.

[Lerner, 2000] Barbara Staudt Lerner. A model for compound type changes encountered in schema evolution. *ACM Transactions on Database Systems (TODS)*, 25(1):83–127, 2000.

[Li and Clifton, 1994] Wen-Syan Li and Chris Clifton. Semantic integration in heterogeneous databases using neural networks. In *Proc. 10th International Conference on Very Large Data Bases (VLDB)*, pages 1–12, Santiago (CL), 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 and Knowledge Engineering (DKE)*, 33(1):49–84, 2000.

[Li *et al.*, 2005] L. Li, B. Wu, and Y. Yang. Agent-based Ontology Integration for Ontology-based Applications. In *Proc. of the Australasian Ontology Workshop (AOW 2005)*, volume 58, pages 53–59, 2005.

[Lozano Tello and Gomez Perez, 2004] A. Lozano Tello and A. Gomez Perez. ONTOMETRIC: A Method to Choose the Appropriate Ontology. *Journal of Database Management*, 15:1–18, 2004.

[Madhavan *et al.*, 2001a] J. Madhavan, P. A. Bernstein, and E. Rham. Generic Schema Matching with Cupid. In *Proc. of the 27th VLDB Conference*, 2001.

[Madhavan *et al.*, 2001b] Jayant Madhavan, Philip Bernstein, and Erhard Rahm. Generic schema matching using Cupid. In *Proc. 27th International Conference on Very Large Data Bases (VLDB)*, pages 48–58, Roma (IT), 2001.

[Madhavan *et al.*, 2005] Jayant Madhavan, Philip Bernstein, An-Hai Doan, and Alon Halevy. Corpus-based schema matching. In *Proc. 21st International Conference on Data Engineering (ICDE)*, pages 57–68, Tokyo (JP), 2005.

[Melnik *et al.*, 2002a] S. Melnik, H. Garcia-Molina, and E. Rahm. Similarity Flooding: A Versatile Graph Matching Algorithm and Its Application to Schema Matching. In *Proc. of the 18th International Conference on Data Engineering (ICDE02)*, 2002.

[Melnik *et al.*, 2002b] Sergey Melnik, Hector Garcia-Molina, and Erhard Rahm. Similarity flooding: a versatile graph matching algorithm. In *Proc. 18th International Conference on Data Engineering (ICDE)*, pages 117–128, San Jose (CA US), 2002.

[Miller *et al.*, 2000] Renée Miller, Laura Haas, and Mauricio Hernández. Schema mapping as query discovery. In *Proc. 26th International Conference on Very Large Data Bases (VLDB)*, pages 77–88, Cairo (EG), 2000.

[Milo and Zohar, 1998] Tova Milo and Sagit Zohar. Using schema matching to simplify heterogeneous data translation. In *Proc. 24th International Conference on Very Large Data Bases (VLDB)*, pages 122–133, New York (NY US), 1998.

[Mitra *et al.*, 1999] Prasenjit Mitra, Gio Wiederhold, and Jan Jannink. Semi-automatic integration of knowledge sources. In *Proc. 2nd International Conference on Information Fusion*, pages 572–581, Sunnyvale (CA US), 1999.

[Mitra *et al.*, 2005] Prasenjit Mitra, Natalya Noy, and Anuj Jaiswal. Ontology mapping discovery with uncertainty. In *Proc. 4th International Semantic Web Conference (ISWC)*, volume 3729 of *Lecture notes in computer science*, pages 537–547, Galway (IE), 2005.

[Mochol and Paslaru Bontas Simperl, 2006] Malgorzata Mochol and Elena Paslaru Bontas Simperl. Practical guidelines for building semantic erecruitment applications. In *Proc. of the International Conference on Knowledge Management (iKnow'06), Special Track: Advanced Semantic Technologies*, 2006.

[Modica *et al.*, 2001] Giovanni Modica, Avigdor Gal, and Hasan Jamil. The use of machine-generated ontologies in dynamic information seeking. In *Proc. 9th International Conference on Cooperative Information Systems (CoopIS)*, volume 2172 of *Lecture notes in computer science*, pages 433–448, Trento (IT), 2001.

[Nottelmann and Straccia, 2005] Henrik Nottelmann and Umberto Straccia. sPLMap: A probabilistic approach to schema matching. In *Proc. 27th European Conference on Information Retrieval Research (ECIR)*, pages 81–95, Santiago de Compostela (ES), 2005.

[Noy and Musen, 2000] Natalya Noy and Mark Musen. PROMPT: Algorithm and tool for automated ontology merging and alignment. In *Proc. 17th National Conference of Artificial Intelligence (AAAI)*, pages 450–455, Austin (TX US), 2000.

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

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

[Noy and Musen, 2003] Natalya Noy and Marc Musen. The PROMPT suite: interactive tools for ontology merging and mapping. *International Journal of Human-Computer Studies (IJHCS)*, 59(6):983–1024, 2003.

[Noy, 2004] Natalya Noy. Tools for mapping and merging ontologies. In Steffen Staab and Rudi Studer, editors, *Handbook of ontologies*, chapter 18, pages 365–384. Springer Verlag, Berlin (DE), 2004.

[Palopoli *et al.*, 2003] Luigi Palopoli, Giorgio Terracina, and Domenico Ursino. DIKE: a system supporting the semi-automatic construction of cooperative information systems from heterogeneous databases. *Software–Practice and Experience (SPE)*, 33(9):847–884, 2003.

[Pan *et al.*, 2005] Rong Pan, Zhongli Ding, Yang Yu, and Yun Peng. A bayesian network approach to ontology mapping. In *Proc. 3rd International Semantic Web Conference (ISWC)*, volume 3298 of *Lecture notes in computer science*, pages 563–577, Hiroshima (JP), 2005.

[Paslaru Bontas and Mochol, 2005] Elena Paslaru Bontas and Malgorzata Mochol. Towards a reuse-oriented methodology for ontology engineering. In *Proc. of 7th International Conference on Terminology and Knowledge Engineering (TKE 2005)*, 2005.

[Paslaru Bontas *et al.*, 2005] Elena Paslaru Bontas, Malgorzata Mochol, and Robert Tolksdorf. Case Studies on Ontology Reuse. In *Proc. of the 5th International Conference on Knowledge Management*, 2005.

[Qu *et al.*, 2006] Yuzhong Qu, Wei Hu, and Gong Chen. Constructing virtual documents for ontology matching. In *Proc. 15th International World Wide Web Conference (WWW)*, pages 23–31, Edinburgh (UK), 2006.

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

[Rham and Bernstein, 2001] E. Rham and P. A. Bernstein. A survey of approaches to automatic schema matching. *Journal of Very Large Data Bases*, 2001.

[Saatly, 1990] T. L. Saatly. How to Make a Decision: The Analytic Hierarchy Process. *European Journal of Operational Research*, (48):9–26, 1990.

[Sayyadian *et al.*, 2005] Mayssam Sayyadian, Yoonkyong Lee, An-Hai Doan, and Arnon Rosenthal. Tuning schema matching software using synthetic scenarios. In *Proc. 31st International Conference on Very Large Data Bases (VLDB)*, pages 994–1005, Trondheim (NO), 2005.

[Shvaiko and Euzenat, 2005a] P. Shvaiko and J. Euzenat. A Survey of Schema-Based Matching Approaches. *Journal on Data Semantics*, 4:146–171, 2005.

[Shvaiko and Euzenat, 2005b] Pavel Shvaiko and Jérôme Euzenat. A survey of schema-based matching approaches. *Journal on Data Semantics (JoDS)*, IV:146–171, 2005.

[Shvaiko *et al.*, 2006] Pavel Shvaiko, Jérôme Euzenat, Natalya Noy, Heiner Stuckenschmidt, Richard Benjamins, and Michael Uschold, editors. *Proc. 1st ESWC 2006 international workshop on ontology matching, Athens (GA US)*, 2006.

[Shvaiko, 2004a] P. Shvaiko. A Classification of Schema-Based Matching Approaches. Technical Report DIT-04-09, University of Trento, `http://eprints.biblio.unitn.it/archive/00000654/01/093.pdf`, December 2004.

[Shvaiko, 2004b] P. Shvaiko. Iterative schema-besed semantic matching. Technical Report DIT-04-020, University of Trento, `http://eprints.biblio.unitn.it/archive/00000550/01/020.pdf`, June 2004.

[Straccia and Troncy, 2005] Umberto Straccia and Raphaël Troncy. oMAP: Combining classifiers for aligning automatically OWL ontologies. In *Proc. 6th International Conference on Web Information Systems Engineering (WISE)*, pages 133–147, New York (NY US), 2005.

[Straccia and Troncy, 2006] Umberto Straccia and Raphaël Troncy. Towards distributed information retrieval in the semantic web: Query reformulation using the oMAP framework. In *Proc. 3rd European Semantic Web Conference (ESWC)*, volume 4011 of *Lecture notes in computer science*, pages 378–392, Budva (ME), 2006.

[Stumme and Mädche, 2001a] Gerd Stumme and Alexander Mädche. FCA-MERGE: Bottom-up merging of ontologies. In *Proc. of the 17th IJCAI 2001*, pages 225–230, 2001.

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

[Uschold and Jasper, 1999] M. Uschold and R. Jasper. A Framework for Understanding and Classifying Ontology Applications, 1999.

[van Rijsbergen, 1979] C.J. van Rijsbergen, editor. *Information retrieval, 2nd edition*. ButterworthsLondon, 1979.

[Velegrakis *et al.*, 2004] Yannis Velegrakis, Renée Miller, and Lucian Popa. Preserving mapping consistency under schema changes. *The VLDB Journal*, 13(3):274–293, 2004.

[Vouros and Kotis, 2005] George Vouros and Konstantinos Kotis. Extending HCONE-merge by approximating the intended interpretations of concepts iteratively. In *Proc. 2nd European Semantic Web Conference (ESWC)*, volume 3532 of *Lecture notes in computer science*, pages 198–210, Hersounisous (GR), May 2005.

[Wang *et al.*, 2004] Jiying Wang, Ji-Rong Wen, Frederick Lochovsky, and Wei-Ying Ma. Instance-based schema matching for web databases by domain-specific query probing. In *Proc. 30th International Conference on Very Large Data Bases (VLDB)*, pages 408–419, Toronto (CA), 2004.

[Xu and Embley, 2003] Li Xu and David Embley. Discovering direct and indirect matches for schema elements. In *Proc. 8th International Conference on Database Systems for Advanced Applications (DASFAA)*, pages 39–46, Kyoto (JP), 2003.

Related deliverables

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

Project	Number	**Title** and relationship
KW	D1.1.4	**System and knowledge technology components for prototypical applications and business cases** introduces the use case that has been more specifically considered here.
KW	D1.2.4	**Architecture of the semantic web framework** describes the semantic web framework and in particular the components for ontology matching that this deliverable is supposed to help choosing.
KW	D2.2.1	**Specification of a common framework for characterizing alignment** provided the framework for us to define the benchmarking actions.
KW	D2.2.2	**Specification of a benchmarking methodology for alignment techniques** defines the methodology that has been used in order to evaluate the methods considered here.
KW	D2.2.3	**State of the art on ontology alignment** provides a panorama of many of the techniques that have been taken into account here. It also provided a begining of classification that has been further developed elsewhere.
KW	D2.2.4	**Description of alignment implementation and benchmarking results** describes the application of the benchmarking techniques to the available systems that has been used here.