UNIVERSITÀ DELLA CALABRIA

Dipartimento di Elettronica,
Informatica e Sistemistica

Dottorato di Ricerca in
Ingegneria dei Sistemi e Informatica
XXI ciclo

Tesi di Dottorato

Ontologies and Semantic Interoperability in Distributed Systems

Giuseppe Pirrò

UNIVERSITÀ DELLA CALABRIA

Dottorato di Ricerca in
Ingegneria dei Sistemi e Informatica

XXI ciclo

Tesi di Dottorato

Ontologies and Semantic Interoperability in Distributed Systems

Giuseppe Pirrò

[illegible]

Coordinatore
Prof. Domenico Talia

[illegible]

Supervisore
Prof. Domenico Talia

[illegible]

DEIS

DEIS- DIPARTIMENTO DI ELETTRONICA, INFORMATICA E SISTEMISTICA
Novembre 2008

Settore Scientifico Disciplinare: ING-INF/05

Acknowledgements

I wish to thank my PhD advisor prof. Domenico Talia for shedding light on my way toward research and helping me in the walk of life.

I wish to thank ”my attorney”, Mr. Steve Valentia for the bunch of ”legal advices” he keeps giving me. A special thank goes to all ”my friends” for their support. Last but not least to the 60’s and 70’s that allow me to do time warp once in a while.

To Hunter S. Thompson for reasons that do not need to be explained here ...

To Bob Dylan for Mr. Tambourine Man ...

To my beloved for ”keeping me down-to-earth” ...

Preface

The Semantic Web (SW) aims at improving today's Web by meaningfully describing online available resources and interconnecting them in a "new" space of *semantic links*. Ontologies are key enablers toward this "new" Web of semantically rich resources. They are meant to provide a vocabulary describing a domain of interest and a *machine-understandable* specification of the meaning of the terms used in the vocabulary. Ontologies are viewed as cutting edge technologies in many applications including Peer-to-Peer (P2P), Knowledge Management (KM), Grid computing and Semantic Web Services (SWS). The objective of this dissertation is twofold.

On one side, we address an important problem related to the use of distributed ontologies, that is, *ontology mapping*. This issue arises since different people or organizations can give different representations of the same or overlapping knowledge domain. We investigate ontology mapping techniques that have been proposed as a bright solution to this problem. These aim at discovering correspondences (aka mappings) between entities belonging to different ontologies. Mappings can be viewed as *semantic links* enabling the knowledge represented in the mapped ontologies to semantically interoperate and are useful for various tasks including semantic search, ontology merging and semantic-driven query routing. In particular, our study addresses the ontology mapping problem from three perspectives:

- *Offline perspective*: here can be placed mapping systems that do not feature any particular facet apart from mapping discovery. We will present a new matcher which exploits linguistic information encoded in ontology entities.
- *User perspective*: current research efforts to ontology mapping focus on only one of the relevant ontology mapping aspects (e.g., the discovering of mappings), while end users rarely focus on a single task. In this context, we discuss a new approach to design mapping systems that on one side supports the user by providing him/her with a user friendly mapping environment and on the other side it "suggests" the most appropriate mapping strategy by scrutinizing the affinities between the two ontologies to be mapped.
- *Online perspective*: this particular instance of the ontology mapping problem is required in dynamic scenarios such as P2P networks. Here, a multitude of

independent peer ontologies autonomously made available by each node joining the system have to be mapped. Mappings must be quickly discovered and only on the parts of ontologies ”contextual” to a specific interaction in which the peers are involved. Besides, complex mapping strategies (e.g., structural mapping based on graph matching) cannot be exploited since peers are not aware of one another’s ontologies. We will present a mapping system to address these issues.

On the other side, we investigate applications of Semantic Web technologies and in particular ontologies in distributed systems. In particular we focus on:

- Distributed and Ontology based Knowledge Management: we devised an ontology framework to represent *organizational* knowledge at different level: (i) organizational level, to define the organizational knowledge background; (ii) group level, to define the knowledge shared by a group of people (e.g., people working on the same project); (iii) personal level, to represent the knowledge owned by a single. This framework has been exploited in a semantic virtual office model called *K-link+*. The requirements of this system have emerged in the context of an Italian research project named KMS-Plus.
- Semantic Service Discovery on the Grid: we defined a semantic service discovery model exploiting ontologies and combining Distributed Hash Tables (DHTs) and Semantic Overlay Networks (SONs). One innovative aspect of this solution is a service matchmaker based on semantic similarity. The requirements of this approach have emerged in the context of the CoreGRID Network of Excellence.

Rende (CS), Italy, *Giuseppe Pirró*
November 2008

Contents

List of Figures

List of Tables

Part I

The Semantic Web: Background and Uptake

1 Introduction and Overview

”Scientia Potentia Est”[1]. Now as never before, this quote took from Bacon’s *Meditationes Sacrae (1597)* seems to apply. We live in the knowledge era where the global economy is shifting away from the production of physical goods toward the manipulation of knowledge. *”The basic economic resource- the means of production- is no longer capital, not natural resource, nor labor. It is and will be knowledge”* Peter Drucker stated [43]. The same author coined the term *”knowledge worker”* to denote someone who is employed because of his/her knowledge of a certain subject, rather than his/her ability to perform manual labor. As a matter of fact, today knowledge is the core asset for organizations and enterprises and therefore its efficient management is crucial to keep current and innovate.

The modern world is glutted by knowledge, formal and informal, partial and comprehensive, out of context and with interpretation. Knowledge in information systems exists in different forms, e.g., databases, documents, e-mails, blogs. Moreover, its amount constantly increases mainly due to two factors: on one side, we have the cheaper and cheaper price of computational resources; today’s average storage space per personal computer is on the order of 200 GBs and the performance in terms of MIPS and FLOPS constantly increases [2]. On the other side, thanks to the Internet, disparate and scattered knowledge sources can get connected to each other thus transforming the whole worldwide network in a distributed knowledge repository. Indeed, this scenario has pros and cons. In the pros we can list the huge amount of knowledge to which one *potentially* has access. People create knowledge, personalize and deepen it according to their specific point of views and needs. However, since there is no way to automatically interpret the meaning of a piece of knowledge, its retrieval can only be performed on syntactic basis (e.g., by Information Retrieval techniques). Besides, it is difficult to interconnect disparate knowledge sources in a meaningful way. Hence, users have to struggle in a labyrinth of hyperlinks and results in order to find out a decent answer to an information need.

[1] *”Knowledge is power”*

[2] Refer to `http://en.wikipedia.org/wiki/Instructions_per_second` for a comparative study of computer performance

To overcome this limitation, Tim Berners-Lee, the inventor of the Web, pictured a new Web where the online information can be meaningfully linked and machines can ”understand” its meaning. This is the Semantic Web [9]. To realize this vision, information has to be endowed with a semantic description. Ontologies [67] have been seen as a promising support to this task. Ontologies are meant at modeling and explicitly representing the knowledge related to a particular domain in terms of concepts, relations between concepts and axioms. This knowledge is used to semantically tag information in a way that it can also be interpreted by machines.

However, this new scenario brings in new problems. As ontologies are inherently distributed and created to fulfill the specific goals of organizations or individuals, there is the need to link them to enable the *semantic information flow*. Hence, discovering (semantic) links between ontologies is one of the next challenges toward the realization of the Semantic Web. More specifically, this problem in the literature is referred to as the ontology mapping problem and concerns the discovery of correspondences (aka mappings) between ontology entities. Mappings allow ontology-based applications to get interconnected in a new space of semantic links. One key contribution of this thesis is a systematic investigation of the ontology mapping problem from different perspectives. For each one of these, we will present innovative ideas and provide implementations and evaluations.

Moreover, by keeping in mind an Anton Chekhov's quote: *”Knowledge is of no value unless you put it into practice”*, in this thesis we will discuss practical applications of ontologies and ontology mapping in the specific context of distributed systems which is one of the keywords of the title of this thesis. It has been widely recognized that distributed applications, and in particular Peer-to-Peer (P2P) networks and the Grid, offer more advantages as compared to their centralized counterparts. Among these we recall fault-tolerance and scalability. These benefits can be further enhanced if we make use of Semantic Web technologies and in particular ontologies. For instance, as for knowledge management, it has been recognized [14] that the best way to manage knowledge is not to adopt ”brain-washing” approaches that clean diversity, but to enable distributed (e.g., P2P) and semantic based knowledge management. In this thesis we will report on a concrete example of the use of P2P and ontologies for organizational knowledge management.

A further application will be shown in the context of the Grid. In this respect we will present an innovative architecture combining Distributed Hash Tables (DHTs) and Semantic Overlay Networks (SONs) to perform semantic-based service discovery.

1.1 Main Contributions

The goal of this thesis is twofold. First, it is meant to systematically investigate the ontology mapping problem from three different perspectives:offline, user-oriented and online. As we will show these perspectives can be identified depending on which is the final aim of the mapping solution. We present innovative methods and respective implementations. Second, it aims at providing a practical outcome of our findings through some applications of ontologies and ontology mapping in distributed systems. In this respect, we will analyze two different use cases emerging from research projects. The first concerns the design of a Peer-to-Peer knowledge management architecture based on ontologies to enable cooperative work. We will describe and evaluate a fully-fledged system i.e., K-link+. The second use case is in the context of the Grid and concerns scalable and semantic-based service discovery. Even in this case an innovative solution, combining Distributed Hash Tables (DHTs), Semantic Overlay Networks (SONs) and semantic similarity will be presented.

The following figure summarizes the context of this thesis. The starting points are ontologies and distributed systems.

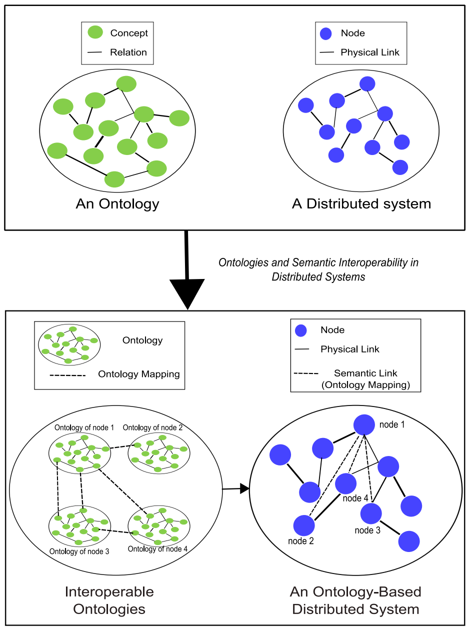


Fig. 1.1. Contribution of this thesis

As can be noted, an ontology includes concepts and relations while a distributed system is composed by a set of nodes that are interconnected through physical links. This work aims at discovering ontology mappings between different and distributed ontologies to enable semantic interoperability. In particular, as shown in the right bottom part of Fig. 1.1 mappings can be seen as semantic links (in general different from physical links) between nodes of a distributed system. We will investigate how ontologies and ontology mappings (semantic links) can be exploited in distributed systems.

1.2 Problem Description

The context of this thesis is specified in more detail by discussing some usage scenarios of ontologies and ontology mapping. These represent the cornerstones of this work and will guide us in designing methods to discover ontology mappings and exploiting ontologies in distributed systems.

- A company has to manage the acquisition of another company. The need to integrate the two knowledge representations arises. This can be done by discovering ontology mappings, that is, correspondences between their knowledge representations (in this case ontologies). However, since this task may not be performed by a knowledge engineer it is mandatory to assist the potential user with a user-friendly mapping environment. We will provide different ontology mapping techniques and a fully-fledged system supporting the user.
- In an open environment (e.g., a P2P network) different peers may use different semantic artifacts (e.g., ontologies, taxonomies) to model their views about the world. These representations can be, for instance, exploited to annotate personal documents, e-mails and so forth. In this scenario, the use of semantic artifacts to annotate content is per se an improvement as compared to current approaches (e.g., file sharing systems enabling keyword search) since peers can pose semantic queries by picking concepts from their own knowledge representations. However, as different knowledge representations are allowed, discovering mappings among them is crucial. Mappings can be exploited, for instance, to rewrite queries according to the local view of a peer. Note that, differently from the previous scenario, here mappings have to be discovered on the fly and should regard the specific parts of ontologies ”contextual” to a request. Moreover, specific mapping techniques have to be devised as peers are typically not aware of one another’s knowledge representations. We therefore differentiate the requirements of a mapping system that has to work in an *online* context. In this respect, we will present specific techniques.
- Another benefit deriving from using Semantic Web technologies in open environments is semantic query routing. As an example, assuming a common artifact (e.g., a taxonomy) to be used by peers to annotate their content, we can compute the semantic similarity between a query and a peer by computing the semantic similarity between the concepts in the query and those to which a peer has annotated its content. This way we can choose the most semantically-similar neighbor

to forward the query to. The similarity is computed by referring to the common semantic structure and allows to contact only relevant peers. We will present a new semantic similarity metric which can be exploited in this and other scenarios.

- Today, an increasing number of knowledge workers works outside of the traditional office for many hours a day. They need to cooperate, share documents, concurrently work on the same piece of information. In a nutshell, knowledge workers require to have their ”desk” available at any time from everywhere. Moreover, in the new economy collaboration between knowledge workers is moving from intra organization to inter organization, that is, across organizational boundaries. Since centralized knowledge management systems have a corporate-based infrastructure and/or proprietary networks to operate, they do not support ad-hoc and volatile collaborations. To cope with these issues, the usage of P2P technologies is a profitable solution. Peer groups, bringing together different people from different organizations, can be formed and dissolved dynamically. We will present a fully-fledge P2P system based on an ontology framework to support semantic-based knowledge management and cooperative work.
- As another example of how Semantic Web technologies can be exploited in open environments let’s consider the Grid. This paradigm has recently moved toward a service oriented architecture. This means that Grid resources and services are exposed in the form of Grid services. As the number of services increases it becomes harder and harder to discover the most appropriate ones fulfilling an information need. Even more complicate becomes to coordinate them e.g, by composing several services to fulfill a unique goal. Semantic Web technologies, and in particular ontologies, can be exploited to semantically describe services. Moreover, P2P architecture are a prominent solution to perform efficient and scalable service discovery. We will describe a P2P architecture making use of Semantic Web technologies to perform efficient semantic-based service discovery and ranking.

This thesis is concerned to investigate ontology mapping and applications of ontologies in distributed systems in a principled way. However, its practical outcome has to be found in the exploiting of this research to real world problems that emerged from two research projects. Therefore, here a trade off can be recognized between a theoretical investigation of ontology mapping and practical applications of ontology mappings (here also referred to as *semantic links*) and Semantic Web technologies in distributed systems.

1.3 Outlook

This section provides the reader with an overview on the content of this thesis. Moreover a chapter dependency schema is sketched which allows to follow the path that motivated each individual chapter and understand how chapters are connected to one another.

1.3.1 Thesis' Structure

Part I comprises two chapters (i.e., Chapter 1 and 2) Chapter 1 introduces and motivates the work presented in the other chapters. Moreover, here the goal of the thesis is stated and the difference w.r.t existing work is outlined. Finally, the structure of the thesis along with a reader's guide is presented. Chapter 2 lays out the basis for the subsequent work. In particular, here will be presented in a high level of detail some applications of Semantic Web technologies and the prominent role of ontologies will be underlined. Special emphasis will be given to the role of ontologies in distributed environments such as P2P networks and the Grid. This approach is helpful since it allows to better differentiate the present thesis from related work. At the end of this chapter, a set of requirements will be identified. In particular the problem of *ontology mapping* will be recognized as a central issue in distributed ontology-based applications. Overall, the aim of this chapter is to grip the reader's interest and create a well-founded motivation for the work done in later chapters.

Part II comprises one chapter (i.e., Chapter 3) which lays out the core problem addressed by this thesis, that is, ontology mapping. In particular, here formal definitions for ontology and ontology mapping will be provided. Moreover, the context in which this problem arises, that is, the Semantic Web will be introduced. Finally, in order to motivate and differentiate the work done in the next part of this thesis (i.e, Part III) a detailed overview of current ontology mapping techniques and systems will be provided. In doing this we blaze a new trail as we classify mapping systems according to three new perspectives: *offline*, *user oriented* and *online*. In the last part of this chapter pointers will be provided to the original contribution presented in Part III.

Part III represents the core of research contribution in the field of ontology mapping. This part comprises four chapters (i.e., chapters from 4 to 7). Chapter 4 tackles the ontology mapping problem from the *offline* perspective. Here, a new approach based on Information Retrieval techniques, that is, the Lucene Ontology Matcher (LOM) will be described and evaluated. Chapter 5 faces the mapping problem from the *user-oriented* perspective. Here, the User Friendly Ontology mapping environment (UFOme) will be presented. Moreover, one of its striking features, that is, the strategy prediction capability will be extensively evaluated. Chapter 6 discusses the SEmantiC COordinator (SECCO) algorithm, which addresses the ontology mapping problem from the third of the identified perspectives, that is, the *online* perspective. Even this approach will be extensively evaluated and compared w.r.t the state of the art. Finally, Chapter 7 discusses another important contribution of this thesis, that is, the design and evaluation of a new semantic similarity metric. This metric can also be seen in a context wider than ontology mapping. As our results will show it yields results above the state of the art. Note that for each of the research contributions presented, corresponding implementations and evaluations have been provided.

Part IV discusses two case studies concerning the application of ontologies in distributed environments. The first case study covering chapters from 8 to 11 has been motivated by an Italian research project named KMS-Plus. In this case study, we investigate the role of ontologies in designing a distributed organizational knowledge management system. In particular, in Chapter 8 we provide a generic framework architecture for organizational knowledge management based on ontologies. Here, a multi-layer ontology framework will be outlined and the application of ontologies for retrieving organizational *knowledge entities* (e.g., documents) will be discussed. In Chapter 9 we face a problem that arises in distributed and collaborative environments, that of content consistency and peer synchronization. Here a hybrid protocol and how it works in different usage scenarios (e.g., creating, updating a new shared object) will be described. Chapter 10 presents an implementation of the architectures described in chapters 8 and 9 in the K-link+ system. K-link+ will be evaluated in terms of semantic search and shared object management. Finally, the last chapter of Part IV, that is, Chapter 11 investigates the application of Semantic Web technologies to enable semantic based service discovery on the Grid. This work has been realized in the context the CoreGRID Network of Excellence jointly with the University of Manchester's Information Management Group. As this is an ongoing work, here we describe the architecture of the system and discuss some research contributions related to the new combination of Distributed Hash Tables (DHTs) and Semantic Overlay Network (SONs). Besides, we describe a new service discovery and ranking mechanism based on semantic similarity.

Finally, Part V sketches final conclusions and discuss future trends in the field. Here the contribution of the present thesis will be once more outlined w.r.t. the motivations and requirements identified at the beginning.

1.3.2 Reader's Guide

The present thesis has been written following a logical path interconnecting the various research contributions. However, it is possible to recognize in this work two main threads. The first is that related to ontology mapping and comprises Part II and III. The second is related to the two case studies in the context of research projects and concerns Part IV.

As for the first thread, a reader interested in this specific problem can focus on these two parts even if the introductory chapter and part of the second one is in any case a must. Part III has been logically divided in three sub-parts each of which treats one of the perspectives we identified in ontology mapping systems. Therefore, the reader could chose only to focus on one of these. In Part III, Chapter 7 can also be place in a context wider than that of ontology mapping. The interested reader can only focus on it as we provide all the needed background information for avoiding him/her to "get lost".

The two case studies presented in Part IV are independent from each other even if chapters from 8 to 10, covering the first case study, should be considered as a single piece of work. The second case study is independent from the rest of the work even if it recalls the work presented in Chapter 7. However, even in this case we provide the reader with the necessary knowledge background.

The following figure summarizes chapters organization and provides links between the work presented in the different parts and chapters in order to allow the reader to choose the parts on which s/he is interested. In particular, three kinds of dependencies between parts and three between chapters are depicted. The relation *concludes* between Part I and Part V indicates that Chapter 12 analyzes the claims presented in Part I on the basis of the research discussed in the various parts. The relations *motivates* from Part II and III and *tackles* the other way around indicates that the content of Part II is exploited to find out missing requirements in current mapping solutions that are tackled in Part III. The relation *exploits* indicates that the contribution introduced in a chapter has been totally or in part used in the work presented in the chapter with the outgoing edge. The relation *implements* indicates that the abstract architecture discussed in a chapter has been implemented in the chapter that has the outgoing edge. Finally, the relation *supports* states that the contribution of a chapter has been devised to support the work introduced in the chapter with the incoming edge.

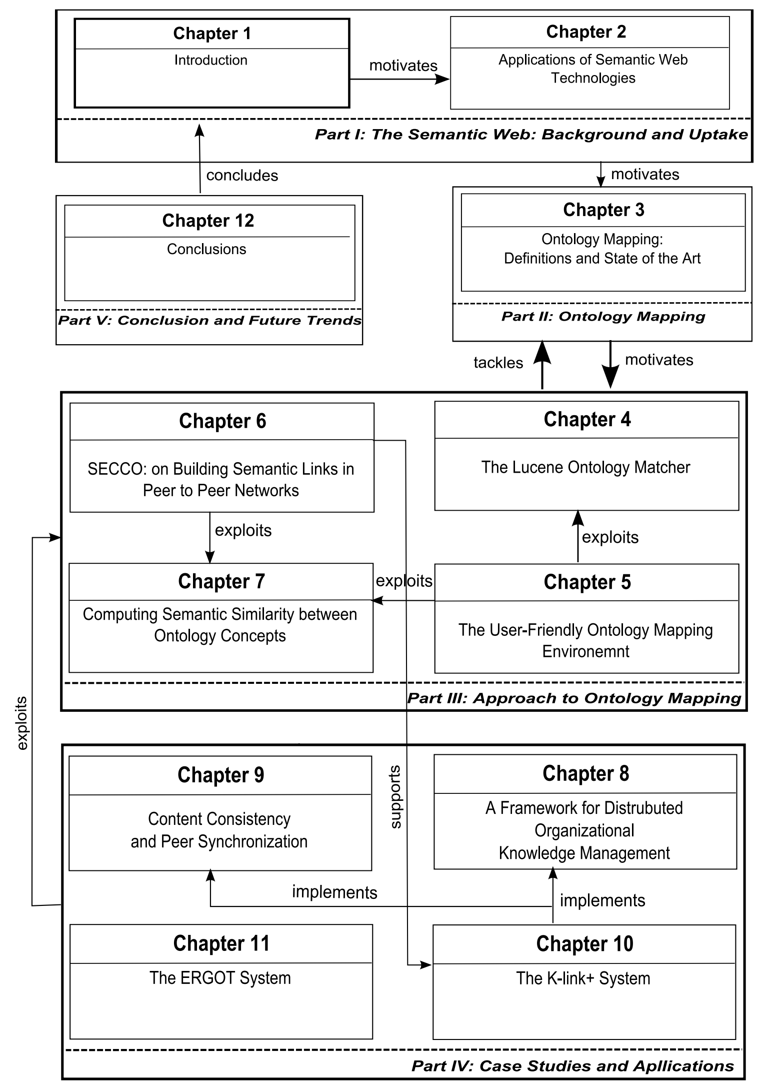

Chapter 1
Introduction
motivates
Chapter 2
Applications of Semantic Web Technologies
Part I: The Semantic Web: Background and Uptake
concludes
motivates
Chapter 12
Conclusions
Part V: Conclusion and Future Trends
Chapter 3
Ontology Mapping: Definitions and State of the Art
Part II: Ontology Mapping
tackles
motivates
Chapter 6
SECCO: on Building Semantic Links in Peer to Peer Networks
Chapter 4
The Lucene Ontology Matcher
exploits
exploits
Chapter 7
Computing Semantic Similarity between Ontology Concepts
exploits
Chapter 5
The User-Friendly Ontology Mapping Environemnt
Part III: Approach to Ontology Mapping
exploits
supports
Chapter 9
Content Consistency and Peer Synchronization
Chapter 8
A Framework for Distrubuted Organizational Knowledge Management
implements
implements
Chapter 11
The ERGOT System
Chapter 10
The K-link+ System
Part IV: Case Studies and Apllications

Fig. 1.2. Structure of the thesis and chapter dependencies

1.3.3 Publications

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

Journals

- G. Pirró, D. Talia, ”LOM: a Linguistic Ontology Matcher Based on Information Retrieval”. *Journal of Information Science,* Sage Publishing, 2008.
- G. Pirró, M. Ruffolo, D. Talia, ”SECCO: On Building Semantic Links in Peer-to-Peer Networks”. *Journal on Data Semantics*, vol. XII, Springer-Verlag, 2008. To appear.
- G. Pirró, M. Ruffolo, D. Talia, ”Leveraging Peer-to-Peer and Ontologies for the Extended Enterprise”. *International Journal of Business Process Integration and Management. Special Issue in Innovations in the Extended Enterprise*, vol. III, Issue III, Inderscience, 2008. To appear.

Book Chapters

- C. Mastroianni, G. Pirró, D. Talia, ”K-Link+: A P2P Semantic Virtual Office for Organizational Knowledge Management”. *In: Semantic Knowledge Management: An Ontology-Based Framework, A. Zilli, E. Damiani, P. Ceravolo, A. Corallo, G. Elia (Editors),* IGI Publishing, 2008. ISBN: 978-1-60566-034-9.

Conferences and Workshops

- G. Pirró, N. Seco, ”Design, Implementation and Evaluation of a New Similarity Metric Combining Feature and Intrinsic Information Content”. *In Proceedings of the International Conference on Ontologies, DataBases, and Applications of Semantics (ODBASE)*, 2008, LNCS, Springer Verlag, 2008.
- C. Mastroianni, G. Pirró, D. Talia, ”A Hybrid Architecture for Content Consistency and Peer Synchronization in Cooperative P2P Environments”. *In Proceedings of the International ICST Conference on Scalable Information Systems (InfoScale)*, 2008.
- G. Pirró, M. Ruffolo, D. Talia, ”An Algorithm for Discovering Ontology Mappings in P2P Systems”. *In Proceedings of the International Conference on Knowledge-Based and Intelligent Information and Engineering Systems (KES) - Session on Advanced-Knowledge-based Systems*, LNCS, Springer-Verlag, 2008.
- E. Le Coche, C. Mastroianni, G. Pirró, M. Ruffolo, D. Talia, ”A Peer-to-Peer Virtual Office for Organizational Knowledge Management”. *In Proceedings of the International Conference on Practical Aspects of Knowledge Management (PAKM)*, LNCS, Springer-Verlag, 2006.
- G. Pirró, M. Ruffolo, D. Talia, ”Advanced Semantic Search and Retrieval in a Collaborative Peer-to-Peer System”. *In Proceedings of the Third Workshop on Use of P2P, GRID and Agents for the Development of Content Networks (UPGRADE-CN) at the High Performance and Distributed Computing Conference*, ACM Press, 2008.
- C. Mastroianni, G. Pirró, D. Talia, ”Data Consistency in a P2P Knowledge Management Platform”. *In Proceedings of the Second Workshop on Use of P2P, GRID and Agents for the Development of Content Networks (UPGRADE-CN) at the High Performance and Distributed Computing Conference*, ACM Press, 2007.

- G. Pirró, D. Talia, ”An approach to Ontology Mapping based on the Lucene search engine library”. *In Proceedings of the International Workshop on Semantic Web Architectures For Enterprises (SWAE) at the International Conference on Database and Expert Systems Applications (DEXA)*, IEEE, 2007.
- G. Pirró, D. Talia, ”UFOme: A User Friendly Ontology Mapping Environment”. *In Proceedings of the Italian Workshop on Semantic Web Applications and Perspectives (SWAP)*, CEUR-WS, 2007.
- G.Pirró, M. Ruffolo, D. Talia, ”K-link: A Peer-to-Peer Solution for Organizational Knowledge Management”. *In Proceedings of the International Workshop on Agents and Peer-to-Peer Computing (AP2PC) at International Conference on Autonomous Agents and MultiAgent Systems (AAMAS 2006)*, LNAI, Springer-Verlag, 2008.

2

Applications of Semantic Web Technologies

The Semantic Web is a promising solution to the limitations of today's Web. In particular, it aims at enabling machine understandable descriptions of available information through ontologies [9]. In the rest of this chapter we will analyze some scenarios where the use of Semantic Web technologies and in particular ontologies may result in a major improvement in the description, management and retrieval of knowledge. Special emphasis will be given to the application of these technologies in distributed systems. The analyzed scenarios have been chosen to depict a wide range of possible applications. Our investigation will consider several facets thus making it possible to take into account different user perspectives. It is worth noting that in real world applications it is more common to make use of a combination of different facets rather than a single one.

2.1 Ontology Based Data Access

The problem of accessing and integrating disparate sources of information has been studied since several years in the field of databases. In particular, given a set of independently developed schemas the final aim is to construct a global view [139]. The problem occurs because schemas modeling world domain are developed by different people in different real-world contexts and then often have different structure and terminology. This issue is relevant to a number of applications including enterprise information integration, medical information management, geographical information systems, and e-Commerce applications.

With the advent of the Semantic Web, the problem of assigning a precise semantic meaning to available information is alleviated. In particular, existing data can be accessed through ontologies. Through ontologies logical transparency in access to data can be achieved [135]. In particular it is possible to:

- Hide to the user where and how data are stored
- Present to the user a conceptual view of data
- Use a semantically rich formalism for the conceptual view.

Ontologies in data access can be used in one of the three following ways [174]:

- *Single ontology approach*: all source schemas are directly related to a shared global ontology that provides a uniform interface to the user. However, this approach requires that all sources have nearly the same view on a domain, with the same level of granularity.
- *Multiple ontology approach*: each data source is described by its own (local) ontology separately. Instead of using a common ontology, local ontologies are mapped to each other. For this purpose, an additional representation formalism is necessary for defining the inter-ontology mappings.
- *Hybrid ontology approach*: a combination of the two preceding approaches is used. First, a local ontology is built for each source schema, which, however, is not mapped to other local ontologies, but to a global shared ontology. New sources can be easily added with no need for modifying existing mappings.

From this analysis emerges that ontologies have a prominent role to support both integration and data access. In some cases to enable ontology-based data access discovering mappings between different ontologies becomes a prerequisite.

2.2 Knowledge Management

The benefits of Semantic Web technologies have attracted also the knowledge management community. Knowledge workers, in order to improve organizational productivity should easily share, manage, organize, find and reuse pieces of knowledge of their interest. In today's Web due to the scarcity of metadata describing information, retrieving a relevant piece of information is very difficult. In addition, organizations have intranets where a huge amount of content are weakly structured. Repositories are usually searched by means of keyword-based search engines allowing a user to retrieve information by stating a combination of keywords. However, these mechanisms are not very precise as they are based on the matching of text strings.

Conversely, in a Web where information is semantically marked up through ontologies it will be possible to find out answers on the basis of the semantic interpretation of search keys. Ontologies enable the representation of shared concepts in a domain by specifying a set of terms to facilitate communication among people (collaboration) and applications systems (integration of tools). For instance, when looking for research articles of a given author, it will be possible to semantically specify the author's name, the topic the article should be about and so forth. The identifications of such things will not be performed on a text string basis but, through Uniform Resource Identifiers (URIs). Next generation knowledge management systems will allow to meaningfully find knowledge sources which are relevant for a problem as well as the process of providing knowledge sources, which can be used in resolving some problems. The prominent role of ontologies in knowledge management is discussed in [60] and [105].

More recently, a new way of intending knowledge management has been proposed, that is, Distributed Knowledge Management (DKM) [14]. This new model enables individual knowledge workers to autonomously manage, organize and share knowledge objects according to their personal ”point of view”. According to this new distributed paradigm, organizational cognition is represented as a distributed process that balances the autonomous knowledge management of individual and groups, and the coordination needed in order to exchange knowledge across different autonomous entities. From this perspective, technology is viewed as a way enabling distributed control, differentiation, customization, and redundancy [15]. This is in contrast with traditional corporate knowledge management infrastructures that tend to clean diversity by uniforming knowledge before storing it. In the DKM scenario, ontologies have a prominent role; these can be used to express individual’s point of view, add semantic markups to personal knowledge, pose semantic-based queries or find semantically-similar peers. However, in this scenario several new problems emerge. In particular, as peers have their own ontologies, it is necessary to integrate them by discovering mappings. Mappings allow peer ontologies to get interconnected in a new space of semantic links.

As another example of ontology-based distributed knowledge management platform, in [4] a system supporting the automatic evolution of dynamic ontologies by a Multi-Agent System is presented. This system uses classification and learning techniques to extract information and to discover new concepts from the retrieved content. These general-purpose agents periodically access a user-given ontology (each agent acts on a part of the ontology) and support search functions resulting in the retrieval of documents related to the ontology concepts. Conceptually similar documents will get clustered into categories; information will then be extracted by statistical approaches. Discovery of new knowledge will lead to modifications in the ontology: pruning of irrelevant sections, addition of new branches, refinement of its granularity and/or testing of its consistency. In this case ontology mapping is not required.

2.3 Peer to Peer Information Sharing

Peer-to-peer (P2P) is a distributed computing model in which participants (i.e., peers) can have similar roles in the process of exchanging data with each other or providing services. This computing model became popular thanks to file sharing systems such as Napster, Gnutella, Kazaa and so forth. Files exchanged in these systems are described by simple metadata in which information such as the file name, dimension, musical genre is included. The major limitation of such an approach is that metadata describing content is shallow and static and peers are not allowed to change it, by describing further information, at their will. Therefore, in some sense peers are not totally autonomous in the system as they have to comply with a single and superimposed schema. This has both pros and cons. On one side, in this scenario the semantic heterogeneity problem, which arises when peers are allowed to create their own content description model, e.g., an ontology, does not exist by construction. Hence, it is not necessary to perform mapping between different peer ontologies. On the other side, the superimposed content description model violates the principle of *autonomy* of peers which is at the basis of the definition of P2P itself. In a P2P system, in fact, peers are autonomous in the sense that they can join or leave the network at their will, share content and so forth.

Indeed, to respect the autonomy principle it would be desirable to allow peers to create their own knowledge representations. However, in this case the problem of linking peer knowledge representations arises. In particular, in order to enable meaningful information searching and exchange it becomes mandatory to discover mappings between peer ontologies. Mappings are useful for several purposes such as query answering, query routing or to find peers with similar interests. In a P2P scenario, the problem of discovering mappings between ontologies takes a specific connotation. While in a traditional data integration scenario, mappings can be discovered at design time, in the case of P2P, peer ontologies need to be mapped and coordinated on-the-fly. The process of mapping should be related only on the parts of peer ontologies specific to a request, that is, there is no need to map the whole two peer ontologies. Finally, incomplete and approximate answers could also be taken into account as they can approximate well-enough an information need. In this respect it would be useful to provide result ranking mechanisms.

2.4 Semantic Overlay Networks

Semantic Overlay Networks (SONs) [33] have been proposed as a mechanism enabling peers to get interconnected with other relevant peers. In SONs, nodes with semantically similar content are ”clustered” together. Queries are processed by identifying which SON (or SONs) are better suited to answer them. A SON exploits one or more semantic artifacts (e.g., ontologies) to semantically support the categorization of content, peers and route queries. In the original formulation of SON proposed by Crespo and Garcia Molina [33] both content and peers were annotated to concepts of a share taxonomy (i.e., all Music Guide).

Through annotations, a set of overlay networks can be defined in a way that, when given a request, a small number of overlay networks will be selected and only peers that have a relevant number of interesting content will be contacted. The benefit of this strategy is two fold. First, the nodes to which the request is sent will have many matches, so the request is answered faster; and second, but not less important, the nodes that have few results for this query will not receive it, avoiding wasting resources on that request. As content and peers have a precise semantic meaning defined by taxonomy concepts, it is possible to improve query routing thus saving messages while obtaining significant values of recall (i.e., the number of relevant results retrieved). Fig. 2.1 shows an example of SON. As can be noted, in a SON peers are linked according to a criterion of semantic similarity.

In order to define ”similarity” between peers or between a peer and a query adhoc mechanisms are necessary. In some cases (e.g., [71]) there is a shared artifact (e.g., a taxonomy) on which a similarity metric can be exploited. Hence, computing

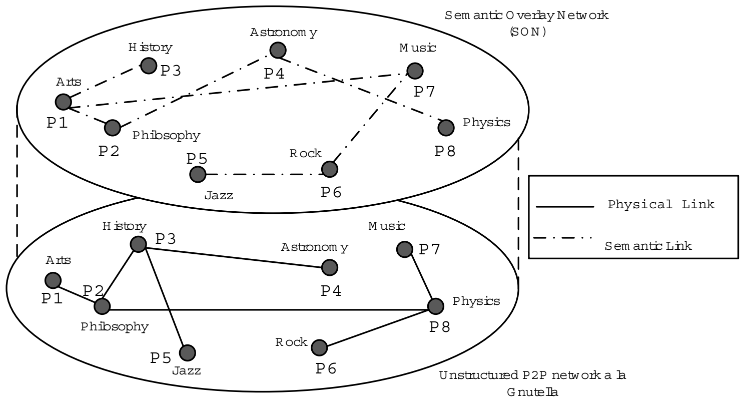


Fig. 2.1. An example of SON

the similarity between two peers requires to compute an overall similarity score between their representative concepts in the shared taxonomy. In other cases, there are no shared artifacts and links between peers are discovered by computing the similarity between the two peer ontologies. As an example, H-Link [118] is a semantic routing approach designed to exploit the results of an ontology matchmaking process for providing a semantic overlay network where peers having similar contexts are recognized and interlinked as semantic neighbors. In particular, H-Link aims at advancing the existing semantic routing protocols by combining ontology-based peer context descriptions and ontology mapping techniques for providing query forwarding on a real semantic basis, in a completely decentralized way. In this case the performance of the system heavily depend on the accuracy of the ontology mapping algorithm.

2.5 Semantic Web Services

Web Services in the last years have gained momentum. Todays, more and more computing platforms are shifting to a service oriented architecture. Also the Grid has evolved from a toolkit centric middleware toward a service oriented architecture in which resources are exposed as services. However, services that have similar functionality but different syntax, are hardly identifiable through traditional approaches. Standards based on XML i.e., the Web Services Description Language (WSDL), can specify the operations available through a Web service and the structure of data sent and received but cannot specify semantic meaning of the data or semantic constraints on the data. This enables to solve only syntactic interoperability.

Semantic Web technologies and in particular ontologies can help to solve ”semantic interoperability” by transforming traditional services into Semantic Web Services (SWs) [19]. SWs enable a precise definition and understanding of the meaning of each input and output parameter as they are built around universal standards for the interchange of semantic data. This makes it easy for programmers (or machines) to combine data from different sources and services without losing meaning. The publishing or advertisement of SWs will allow agents or applications to more precisely discover services based on well-defined goals and capabilities. In this respect, a semantic service registry can be used for registering semantic service profiles. Recently, initiatives such as METEOR-S [129] have been proposed. METEOR-S aims at exploiting existing Web services technologies and combine them with ideas from the Semantic Web to create a better framework for Web service discovery and composition. In more detail, the MWSAF (METEOR-S Web Service Annotation Framework) framework is meant to semi-automatically marking up Web service descriptions with ontologies. This happens through an algorithm to match and annotate WSDL files with relevant ontologies. Domain ontologies are used to categorize Web services into domains.

More formal frameworks for annotating Web services have recently been proposed such as OWL-S and WSMO. As an example, OWL-S (formerly DAML-S) is an OWL ontology with three interrelated subontologies, known as the profile, process model, and grounding (see Fig. 2.2). In brief, the profile is used to express ”what a service does”, for purposes of advertising, constructing service requests, and matchmaking; the process model describes ”how it works”, to enable invocation, enactment, composition, monitoring and recovery; and the grounding maps the constructs of the process model onto detailed specifications of message formats, protocols, and so forth (normally expressed in WSDL). The inputs and outputs of an atomic process are given types from the (class hierarchical, description logic-based) typing system of OWL, which allows for the use of concepts defined and shared as part of the Semantic Web [108].

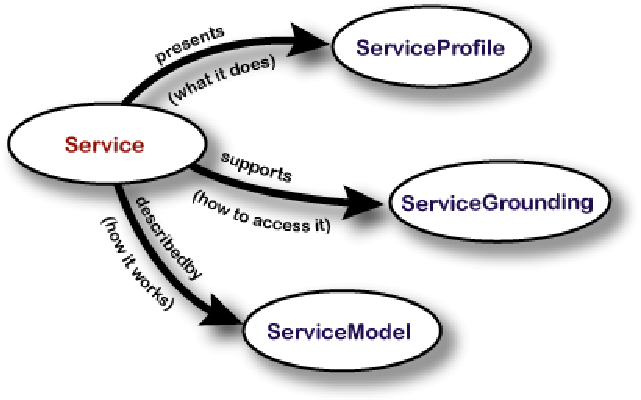


Fig. 2.2. The OWL-S approach to Semantic Web services

Another important task to be addressed in SWs is that of discovery. The discovery of SWs consists in the process of semantic matchmaking between a service request and a service profile. A request can be expressed in terms of service name, inputs, outputs, preconditions and other attributes. The process of matchmaking can be performed in different ways. For instance, it can be performed by reasoning operations to determine semantic relations between concepts in the request and in the profile. For example, an input of type *PhD Student* in the description of a service can be said to match an input of type *Student* in the request as there exist an inheritance relation between *PhD Student* and *Student* defined in an ontology. The matching can also be done at the level of tasks or goals to be achieved, followed by a selection of services which solves the task. The process of service matchmaking can exploit a common ontology as in [12] or entities can define their own ontologies. In this case ontology mappings need to be discovered.

Moreover, as the number of online available services increases, some issues related to scalable service discovery arise. Approaches based on centralized registries such as UDDI can encounter serious problems in managing the incoming rate of service requests. As a central instance, UDDI does not easily scale to a growing number of users and lacks acceptance by the industry. Moreover it does not support semantic-based service discovery. To cope with these issues, P2P architectures combined with Semantic Web technologies have been recently adopted. As an example, the WSPDS system [86] aims at constructing an overlay network of peers (here called servents, that is, server and client at the same time) by comparing their data content (Web service descriptions). In particular, nodes create links by comparing the inputs and the outputs of their services. Besides, the similarity between a query and the peers to whom forward it is computed by an ad-hoc matchmaker. Each WSDL description has associated a WSDL-S [1] exploited to semantic annotate a service. The matching between requests and services is enhanced by annotating both with concepts belonging to a shared ontology.

2.6 Semantic Search

Today's Web is a huge container of information. Unfortunately, the unstructured nature of this information have made it increasingly difficult for users to sift through and find relevant information. Numerous information retrieval techniques have been developed to help deal with this problem. Most used information retrieval techniques are based on the analysis of keywords and exploit the Vector Space Model [149]. These techniques use keyword lists to describe the content of information. However, one problem with such lists is that it is not possible do not say anything about the semantic relationships between keywords.Moreover, the meaning of words and phrases is not took into account.

To cope with these issues and enable semantic-based search ontologies are seen as a bright solution [70]. The main hypothesis is that the inclusion of conceptual knowledge such as ontologies in the process of information retrieval can contribute to the solution of its problems. For instance, using conceptual knowledge can help

[1] `http://www.w3.org/Submission/WSDL-S`

users to precisely formulate their requests. However, the use of ontologies to perform semantic search has a number of challenges. First, it is necessary to establish a criterion of semantic similarity between terms in the underlying ontology in order to quantify the degree of fuzziness of results. Second, it is necessary to recognize semantic information in texts and map of this knowledge into the ontologies in use.

The approach described in [25] proposes a model for the exploitation of ontology-based Knowledge Bases to improve search over large document repositories. This approach includes an ontology-based scheme for the semi-automatic annotation of documents and a retrieval system. The retrieval model is based on an adaptation of the classic vector-space model, including an annotation weighting algorithm, and a ranking algorithm. Semantic search is combined with keyword-based search to achieve tolerance to Knowledge Base incompleteness.

Other approaches (e.g., [97]) make usage of lexical ontologies such as WordNet to ”expand” query terms with neighbors terms as for instance synonyms, hyponyms and so forth. In particular, in [172] is discussed a novel information retrieval approach built on the notion of semantic similarity. This method is capable of detecting similarities between documents containing semantically similar but not necessarily lexicographically similar terms. Here, semantic similarity plays a central role therefore designing an accurate similarity measure is mandatory.

2.7 Summary

We identified a set of requirements that emerged from the previous scenarios that will help to put focus on the content of this thesis:

- *Mapping Discovery*: Discovering ontology mappings has clearly emerged as a main issue to be addressed when dealing with distributed ontologies. We can see that the features that a mapping solution should provide vary depending on the application context. For instance, in a data integration context it would be useful to have a user-friendly system supporting the user in all the phases of the integration process. In this respect the burden to the user (e.g., parameter tuning) has to be as low as possible. Besides, the problem of discovering ontology mappings in open environment should be faced by ad-hoc techniques to ensure fastness as mappings have to be discovered ”on the fly” and only of the relevant parts of the specific interaction in which peers are involved.
- *Application of Ontologies for Knowledge Management*: Ontologies are a cutting edge technology in the management of knowledge as they enable to semantically characterize pieces of knowledge thus enabling their efficient retrieval and reuse. In an organizational context it would be very useful to define an ontology framework to give structure to the different organizational knowledge entities. Also ontology-based knowledge retrieval should be enabled. Finally, as distributed knowledge management has been proposed as a promising alternative to traditional knowledge management, it is mandatory to enable knowledge workers to work in a decentralized fashion by providing them with a cooperative environment for knowledge sharing and exchange.

- *Computing Semantic Similarity*: The problem of computing semantic similarity has emerged as a central issue to be faced in many application scenarios. For instance, semantic similarity can be used to discover mappings between two ontology entities (e.g., classes) whose names are syntactically non similar (e.g., car-automobile). Other applications of semantic similarity have been recognized in the context of SONs and service matchmaking.
- *Semantic Web Service Discovery*: Efficient service discovery is a main issue when the number of available services is large. Traditional approaches based on centralized architecture such as UDDI are poorly scalable and only provide syntactic-based service discovery. Therefore, new approaches more scalable and making use of semantic technologies are required.

In the rest of this thesis we will elaborate more on these requirements providing solutions for each of them.

Part II

The Ontology Mapping Problem

3

Ontology Mapping: Definitions and State of the Art

This chapter lays the foundations in terms of formal definitions of ontology and ontology mapping that will serve as a basis to introduce the Ontology Mapping Problem (OMP). Subsequently, a detailed review of some existing mapping systems is presented. However, since several surveys (e.g., [29, 122]) and books on ontology mapping (e.g., [56, 44]) extensively discuss mapping systems and their comparison, our aim here is merely to show their variety. In order to comply with this objective we proceed as follows. First, we provide a brief overview of some individual strategies that can be exploited to discover mappings and discuss how their individual contributions can be combined toward determining and overall similarity value between two ontology entities. Then, we analyze mapping systems by classifying them under one of the following three perspectives: *offline*, *user-oriented* and *online*. The basic principles of the analyzed systems will be briefly outlined and special emphasis will be given to their respective peculiarities. The content of this chapter will be useful both to identify potential shortcomings of current mapping solutions and motivate the work presented in the next part of this thesis (i.e., chapters from 4 to 6).

3.1 Definitions

In order for the reader to deeply understand the OMP it is necessary to provide him/her with the adequate knowledge background. In this respect, the following sections will provide definitions for ontology, ontology mapping and related terminology. These notions will be accompanied by practical examples that, hopefully, will shed more light on their meaning.

3.1.1 Ontology

Ontologies' history dates from Aristotle with his studies on categories (350 BC). Ontologies are a central branch of metaphysics and investigate what types of things there are in the world and what relations these things bear to one another. In the course of the years, the notion of ontology assumed different facets depending on what point of view it has been considered. In artificial intelligence, one early influential use of the term ontology was by John McCarthy in his 1980 paper on ”circumscription” [109]. In the early 1990’s, an effort to create interoperability standards identified a technology stack called out the *ontology layer* as a standard component of knowledge systems [120]. Some definitions of ontology are also discussed by Nicola Guarino in [69] and Tom Gruber in [68]. The definition provided by the latter, is the most commonly accepted in the computer science community. Gruber defined an ontology as *”an explicit specification of a conceptualization”*. This definitions states that through ontologies the terminology related to a particular knowledge domain is identified and made explicit. In more detail, ontologies can be seen as artifacts designed with the specific purpose to model some knowledge domain. Often the original definition of ontology is complemented with additional constraints thus leading to the following rearranged definition: *”An ontology is an explicit, formal specification of a shared conceptualization of a domain of interest”*. Here, *formal* means that the definition should not be ambiguous and should come with formal constraints defining the coherent use of ontologies. This way, both humans and machines can understand what has been modeled. The word *shared* states that ontologies aim at capturing knowledge typically shared by a group (e.g., an organization). Finally, as in the definition a domain of interest is mentioned, we can say that ontologies are used not to model the whole word but rather specific parts that are of interest to fulfill a goal. For a comprehensive discussion about the term ontology refer to [67].

After a high level presentation of what an ontology is and what it is used for, it is time to provide a more formal definition to which we will refer throughout this thesis. The following definitions adhere to the ontology model described in [162].

Definition 3.1 ***(Ontology Structure)***. *An ontology structure S is a five-tuple of the form:*

$$S := \langle C, \leqslant_C, R, \sigma, \leqslant_R \rangle$$

consisting of:

- *two disjoint sets C and R whose elements are called concept identifiers and relation identifiers (concepts and relations from now on).*
- *a partial order $\leqslant_C$ on C which defines the concept hierarchy and is referred to as the taxonomy.*
- *a signature function $\sigma : R \rightarrow C \times C$ that associates each relation name $r \in R$ with a couple of concepts belonging to C. Given a relation r, the first attribute of the tuple defines the relation domain $dom(r) = \pi_1(\sigma(r))$ and the second attribute the range $range(r) = \pi_2(\sigma(r))$.*
- *a partial order $\leqslant_R$ on R which defines the relation hierarchy.*

In this definition, datatypes (e.g., integers, strings) are treated as special kinds of concepts. Moreover, if $c_1 < c_2$ then c_1 is a *subconcept* of c_2 and c_2 is a superconcept of c_1 with $c_1, c_2 \in C$.

Definition 3.2 ***(Axioms)***. *It is possible to define relations between concepts along with specific constraints needed to model the knowledge domain by using logical* *statements referred to as axioms. The truth of an axiom is taken for granted and serves as a starting point for deducing and inferring other (theory dependent) truths. Therefore, an ontological definition can comprises a set A of axioms that can be expressed through logical languages such as first-order logic or description logics.*

An ontology structure is meant to define how concepts and relations are arranged in the domain. In order to link the domain model with the reality it is necessary to populate the ontology with instances. This yields to the definition of knowledge base.

Definition 3.3 ***(Knowledge Base)***. *A knowledge base is a five-tuple of the form:*

$$KB := \langle C_{KB}, R_{KB}, I, \iota_c, \iota_r \rangle$$

consisting of:

- *two disjoint sets C_{KB} and R_{KB}*
- *a set I whose elements are called instance identifiers, instances from now on.*
- *a function $\iota_c : C_{KB} \rightarrow \mathfrak{I}(I)$ called concept instantiation*
- *a function $\iota_r : R_{KB} \rightarrow \mathfrak{I}(I^2)$ with $\iota_r(r) \subseteq \iota_c(dom(r)) \times \iota_c(range(r))$, for all $r \in R$. This function is called relation instantiation.*

As in our definition we consider datatypes as concepts, concrete values for datatypes are treated as instances. In the rest of this thesis we use the term entity e to generally refer to either a concept $c \in C$, a relation $r \in R$ or an instance $i \in I$.

Besides, concepts, relations and instances have associated names, that represent linguistic symbols to characterize them. Often ontology languages also allow to associate to concepts, relations and instances additional metadata in the form of labels and comments. We refer to this information as the linguistic description of an ontology.

Definition 3.4 ***(Linguistic description)***. *The linguistic description L is a three-tuple of the form $L := \langle L_c, L_r, L_i \rangle$ where the set L_c contains the linguistic description of concepts. Analogously are defined L_r and L_i for relations and instances respectively.*

Overall we can now define an ontology as the union of the components defined previously.

Definition 3.5 ***(Ontology)***. *An ontology O is defined by the following four-tuple:*

$$O := \langle S, A, KB, L \rangle \text{ consisting of:}$$

- *the ontology structure S*
- *the set of axioms A*
- *the knowledge base KB*
- *the linguistic description L*

3.1.2 Ontologies and the Semantic Web

After its original starting in rather restricted environments, typically academic co-operation networks, the Internet has now reached huge dimensions. Moreover, the cheaper and cheaper cost of hardware devices so that each person owns a computer, contributed to increase the amount of information today available. On one side this is a great opportunity as people can search for almost whatever the human brain can conceive. On the other side, such a large and diverse amount of information raises the problem to find the right one, that is, the one that fits with what was originally in the mind of who started the search. If we remain in restrict environments it is possible, even if tough, for a human to handle with the available information for instance by browsing it looking for the sought answer. However, as we move toward global environments this goal becomes impossible to met. Therefore, it would be valuable to have a ”support”, that is, an automatic method to help automatically filtering and connecting only the interesting information. In order to realize such a dream, computers should be able to ”understand” the meaning of information.

This is the vision brought up by Tim Berners Lee when he for the first time coined the term *Semantic Web* [9]. The Semantic Web is an extension of the current Web where machines are able to understand the meaning of content they deal with. In order for content to have a well precise meaning such that machines can interpret it, the exploiting of some artifacts becomes necessary. These artifacts are ontologies. Through ontologies in fact, it is possible to formally define the knowledge embedded in a particular domain and share it. This explicit representation of the semantics of data makes it possible to provide a qualitatively new level of Web services. In this scenario, software agents can profitably interact with one another thus making semantics-driven information sources interconnected in a denser space of semantic links.

As one can wonder, it is necessary to build ontologies in a format such that they can be suitable for the Web. This requirement gave birth to several ontology languages that are becoming more and more expressive. The first step toward the definition of a machine-interpretable modeling information language was the Resource Description Framework (RDF) [1]. RDF, in its original formulation was not an ontology language but just a means to express assertions on the form of *triples* composed by a *subject*, a *predicate* and an *object* where the subject denotes the resource and the predicate denotes traits or aspects of the resource and expresses a relationship between the subject and the object. RDF is an abstract model with several serialization formats (i.e., file formats) and so the particular way in which a resource or triple is encoded varies from format to format. An extension of RDF, that is, RDF-schema allows for the creation of relations among concepts and taxonomies. Since these first attempts several other languages have been proposed. Among these, the most prominent are DAML (DARPA Agent Markup Language) and OIL (Ontology Inference Layer) which gave birth to the current W3C standard ontology language, that is, the Ontology Web Language (OWL)[2]. OWL is available in three flavors: OWL-Lite,

[1] `http://www.w3.org/RDF`

[2] `http://www.w3.org/2004/OWL`

OWL-DL and OWL-FULL which increasing complexity. In particular, OWL-Lite provide constructs to allow simple inference. OWL-DL, which provides most of the constructs available in Description Logics while ensuring decidability of reasoning procedures. Finally, OWL-Full provides a wider range of constructs at the expense of decidability of reasoning operations.

More recently some attempts have been provided which aim at extending ontology languages with rules. In this respect, the Semantic Web Rule Language (SWRL) [76] has been proposed. SWRL combines sublanguages of OWL DL and Lite with those of the Rule Markup Language (Unary/Binary Datalog).

3.1.3 An Ontology Example

In order to clarify the terminology introduced in the previous sections here we present an ontology example. In the representation shown in Fig. 3.1 we use circles to represent concepts, solid rectangles to represent object properties and dotted rectangles to represent datatype properties. Moreover an incoming arrow indicates the domain of a property while the range is indicated by an outgoing arrow. Inheritance relations are represented as solid arrows. Finally, instances of concepts and instantiation of relations are depicted as dotted arrows. The considered ontology is meant to model the domain of music records. As can be observed, a music record can encompass different categories as for instance tape, vinyl, CD. Moreover a record has an artist, a label and a year of issue. Finally an artist is assigned a country.

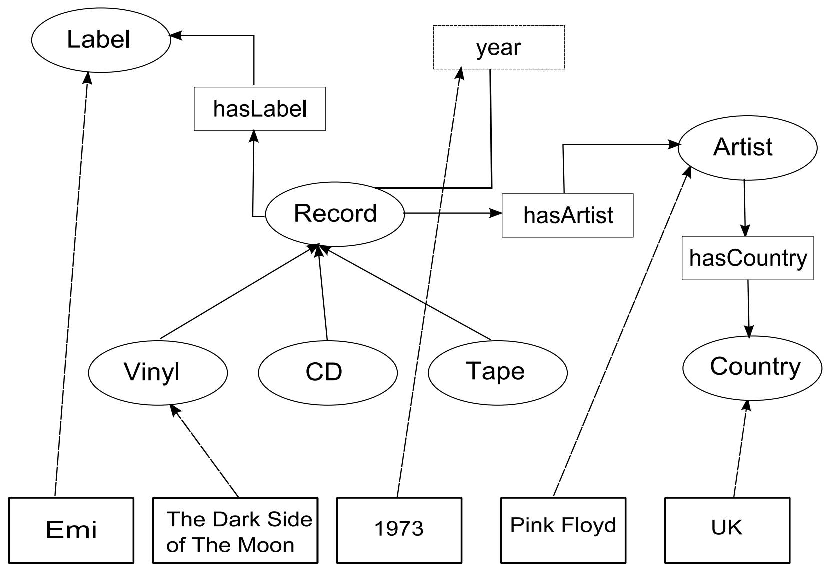


Fig. 3.1. An example of ontology

In the following figure the OWL representation of the ontology represented in Fig. 3.1 is shown.

```
<rdf:RDF
...
xmlns="http://grid.deis.unical.it#"

 <owl:ObjectProperty rdf:about="#has_artist">
        <rdfs:label>artist</rdfs:label>
        <rdfs:comment
            >The artist who authored the record</rdfs:comment>
    </owl:ObjectProperty>
   ...

      <owl:DatatypeProperty rdf:about="#year">
        <rdfs:label>year</rdfs:label>
        <rdfs:comment
            >The year of publication</rdfs:comment>
    </owl:DatatypeProperty>

    <owl:Class rdf:about="#Artist">
        <rdfs:label>Artist</rdfs:label>
    </owl:Class>
 ....
      <owl:Class rdf:about="#Tape">
        <rdfs:label>Tape</rdfs:label>
        <rdfs:subClassOf rdf:resource="#Record"/>
    </owl:Class>
 ....
      <Label rdf:about="#EMI">
        <rdfs:label>EMI</rdfs:label>
    </Label>

    <Artist rdf:about="#PinkFloyd">
        <rdfs:label>PinkFloyd</rdfs:label>
    </Artist>

    <Vinyl rdf:about="#TheDarkSideofTheMoon">
        <rdfs:label
            >TheDarkSideofTheMoon</rdfs:label>
    </Vinyl>

    <Country rdf:about="#UK">
        <rdfs:label>UK</rdfs:label>
    </Country>
```

```
</rdf:RDF>
```

3.2 Ontology Mapping: Definition and Example

In a distributed environment such as the Semantic Web, is not feasible to have a single (and universally accepted) ontology describing the whole universe. As ontologies start to become a commodity for semantics-driven applications, there will be different ontologies, independently designed, describing the same or overlapping knowledge domain. This makes ontologies suffering from several forms of heterogeneity owing to, for instance, the different names used to dub ontology entities (*lexical heterogeneity*) or the degree of detail adopted in modeling the domain (*structural heterogeneity*). As an example, an ontology O_1 can use the word *car* to represent a motor vehicle while an ontology O_2 can express the same information by the word *automobile*. Moreover, O_1 can include information about convertible *car* while O_2 cannot. A detailed survey on the possible forms of ontological heterogeneity is provided in [51].

As a matter of fact, ontology-based applications in order to (semantically) exchange information with one another in the Semantic Web need to ”harmonize” their ontologies. This issue in the literature is referred to as the ontology mapping problem (OMP) and concerns the discovery of correspondences (aka mappings) among entities (e.g., concepts, relations) belonging to different ontologies (i.e., a *source* and a *target* ontology). Ontology mapping is crucial for the success of the Semantic Web since it allows ontology-based applications to get connected in a ”new” space of *semantic links* and meaningfully exchange information. Semantic links can be profitably exploited by several kinds of semantic applications such as search, query rewriting, query routing, Web service composition and so forth. Is is important noting that mappings aim at representing the semantic relations between two ontologies and often they are considered as non bidirectional. Moreover, mapping discovery is tied to a specific application task such as query rewriting, Web service composition and so forth.

Fig. 3.2 shows three excerpts of ontologies. Fig. 3.2 (b) and Fig. 3.2 (c) are alterations of the ontology in Fig. 3.2 (a). In particular, in Fig. 3.2 (b) the structure was altered and names of entities randomized whereas in Fig. 3.2 (c) the names of entities were not altered and the two ontologies contain some overlapping information. If we consider the ontology in Fig. 3.2 (a) as *source ontology* and the ontologies in figures 3.2 (b) and 3.2 (c) as *target ontologies* it is possible to establish some mappings between their entities as shown in Tables 3.1 and 3.2.

As can be noted, mappings between ontologies (a) and (c) are obvious since the mapped entities have identical names. Conversely, mappings between ontologies (a) and (b) are not so immediate. In fact, they can be discovered by scrutinizing the structural similarity between entities. In particular, the dotted areas in Fig. 3.2 can be considered similar if we look at their structure defined by the RDF(S) *rdfs:domain* and *rdfs:range* constructs. Note that in Tables 3.1 and 3.2 the last column represents a value of overlap between the compared entities. Typically, the more this value

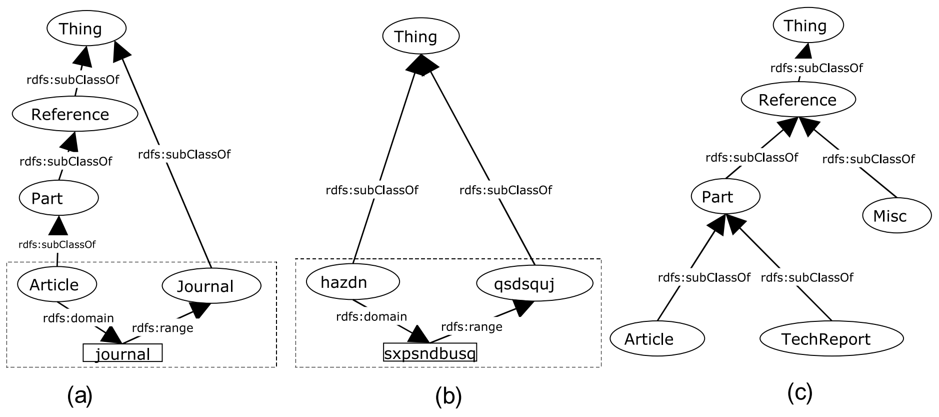


Fig. 3.2. Three excerpts of ontologies

Table 3.1. Examples of ontology mappings between (a) and (b)

Source Ontology: (a)	**Target Ontology: (b)**	
Source Entity	*Target Entity*	*Value of Overlap*
Reference	Reference	1.0
Part	Part	1.0
Article	Article	1.0

Table 3.2. Examples of ontology mappings between (a) and (c)

Source Ontology: (a)	**Target Ontology: (c)**	
Source Entity	*Target Entity*	*Value of Overlap*
Article	hazdn	0.577
Journal	sxpsndbusq	0.577
Journal	Article	0.577

approaches 1 the more the two entities are similar. On the other side, some mapping systems that implement the *semantic-matching* model [17, 65] give as output not a similarity value but a semantic relation (e.g., equality, subsumption). In this thesis the focus is on determining confidence values between ontology entities.

Once introduced the OMP and clarified what mapping discovery stands for, it is time to put emphasis on techniques that allow to discover mappings and corresponding overlap values.

3.2.1 How to Represent Ontology Mappings

Even if the OMP has received a lot of attention from the scientific community, there is still no standardized format for storing ontology mappings. Ontology languages such as OWL provide built-in constructs for representing equivalence between concepts (i.e., *owl:equivalentClass*), relations (i.e., *owl:equivalentProperty*) and instances (i.e., *owl:sameAs*). This approach allows OWL inference engines to automatically interpret the semantics of mappings and perform reasoning across different ontologies. However, a confidence value cannot be interpreted. Therefore, in [49] an alternative mapping representation based on XML is proposed. Here an ontology mapping m can be defined as a four-tuple of the form: $m = \langle e_s, e_t, r, k \rangle$ where:

- e_s represents the entity e_s in the *source ontology* O_s.
- e_t represents the entity e_t in the *target ontology* O_t.
- r is the kind of relation (e.g., exact, narrower, broader, partial overlap) between e_s and e_t.
- k $\rightarrow$ [0,1] represents the degree of partial overlap between e_s and e_t. The closer k is to 1 the more similar e_s and e_t are.

This latter approach is more flexible and obtain the degree of overlap by combining the results provided by the individual mapping strategies. However, the use of either formulation heavily depends on the application domain.

3.2.2 Related Terms

In this thesis the focus will be on devising techniques to discover mappings between ontology entities. As we will show, these technique can apply in different contexts even if in some cases, due to the specific requirements of the application scenario, adhoc techniques should be adopted. Since its starting, research focused on resolving the semantic interoperability problem has generated a large terminology in which terms are often not properly used. In this section, we will try to shed a bit more of light on this terminology. For a more comprehensive discussion about this aspect refer to [44].

- **Integration:** with the term integration is indicated the process that enables one or more ontologies to be used for a new ontology. In particular, through integration the original concepts are not changed but can possibly be extended. This approach results particularly useful when there is need to extend the knowledge covered by a given ontology by complementing it with that encoded in another ontology covering a different domain. In order to discover the correct way of connecting the two ontologies it is necessary to discover their similar parts. Therefore, discovering correspondences between the two ontologies becomes a prerequisite.
- **Merging:** ontology merging has the aim to produce an ontology given two or more ontologies as input and requires some entities belonging to the two ontologies to be replaced by a single entity in the final ontology. Even in this case, similar entities should be identified by discovering mappings.
- **Mediation:** this process concerns the reconciling in an upper-level way the differences between heterogeneous ontologies. This process requires mapping between entities to be discovered. Ontology mediation allows the interoperation between data sources that may exploit ontologies as underlying technology to describe their content. An application of mediation can be query rewriting.

- **Matching:** matching two ontology consists in discovering couples of correspondences between their entities. The kind of relation between these entity is fixed.
- **Alignment:** ontology alignment aims at merely identifying the relation between ontologies and is not related to a specific task such as query rewriting, query routing and so forth.

3.3 Ontology Mapping Techniques: an Overview

In the ontology mapping (OM) community, several mapping algorithms have been proposed [29, 56, 122, 44]. These are based on techniques borrowed from different research areas such as Bayesian Decision theory (e.g., [116, 165]), Graph Similarity (e.g., [179]), Information Retrieval (e.g., [163, 133]), Description Logics [17] just to name a few. Current approaches to OM compute similarities among entities by leveraging the different kinds of information (e.g., linguistic, structural) encoded in ontologies. However, since a single mapping approach cannot grasp all this information, different individual strategies are usually adopted. In the next sections we provide a brief overview of some general mapping techniques. Our aim is not to provide an exhaustive survey on existing strategies (see [56] for a more detailed overview) but to give some insights on how mapping algorithms assess similarity between ontology entities. In this section we also discuss some aspects related to the combinations of results provided by different individual strategies.

3.3.1 Linguistic Techniques

Linguistic mapping techniques aim at exploiting the linguistic information encoded in ontology entities. This information encompasses the entity name, RDF(S) comments, labels and other annotation properties. The most natural approach to compute linguistic similarity between two entities is to compare their respective names and/or descriptions. In this respect, some popular string comparison approaches such as the Jaro Winkler [180], the Levenshtein [98] and the EditDistance metrics are usually adopted. Besides, an ad-hoc string comparison metric (i.e, the I-sub [161] metric) has been devised in the context of OM.

More recent solutions toward linguistic-based ontology mapping (e.g., [133, 137]) rely on the concept of virtual document which represents an ontology entity as a vector of terms in the space of all terms belonging to the two ontologies to be mapped. This approach grounded on Information Retrieval techniques computes the similarity between two entities as the cosine between their representative vectors [149].

Even if these approaches can be successful in many cases, they are not capable to grasp possible semantic relations between words used to dub ontology entities. For instance, two ontology entities dubbed as *car* and *bus* respectively, even if syntactically not similar are semantically similar as they are both means of transportation. In order to overcome this issue, mapping approaches often exploit generic lexical ontologies such as WordNet [112] or more specialized one such as MeSH [3]. As these ontologies contain semantic relations between terms it is possible to compute their semantic similarity by scrutinizing, for instance, their position in the ontology. To date, several semantic similarity metrics have been proposed which can be generally classified as either path based approaches, which mainly exploit the position of the two word in the ontology, or information theoretical approaches which exploit the notion of Information Content originally proposed in [142]. A comprehensive discussion on semantic similarity metrics will be provided in Chapter 7.

3.3.2 Structural Techniques

Structural mapping exploits the arrangement of the concepts in the *source* and *target* ontologies. In particular, a common approach is to consider ontologies as graphs. Hence, the similarity between two entities (nodes) belonging to two different ontologies is computed by scrutinizing their position within the ontologies. The main intuition here is that if two nodes from two ontologies are similar, their neighbors might also be similar to some extent. Several approaches scrutinize leaves, children and relations of graph nodes representing ontology entities [46]. More sophisticated approaches exploit graph matching techniques to gain similarity between ontology entities. Some approaches in this field have been proposed by Wei et al. [179] and Tous et al. [166]. The first approach exploits a bipartite-graph representation of ontologies and the graph matching algorithm proposed in [13]. The second one exploits built-in RDF(S) and OWL properties by representing them in a vector space and compute graph matching by the same algorithm (i.e., [13]). Another pioneer work in structural matching is the Similarity Flooding algorithm [111] originally designed for schema matching. This algorithm has recently been adapted for ontology mapping as discussed in [184].

Note that the abovementioned techniques rely on rather complex strategies such as graph matching and require to scrutinize the whole two ontologies to be mapped. Therefore, in some cases and in particular contexts where ontology mappings need to be quickly and dynamically discovered (e.g., P2P environments) such techniques become prohibitive.

3.3.3 Instance Based Techniques

Another class of mapping techniques aims at exploiting the information encoded in ontology instances. As an example, there are some approaches e.g., [163] that treat instances of the same ontology entity as virtual documents and represent them as vectors of terms in the space of all the terms appearing in the two ontologies. Therefore, each entity will be represented by a vector representative of all its instances. Finally, the similarity between entities is computed as the cosine between their representative vectors. Instance-based matching is also discussed in [80, 89].

[3] `http://www.nlm.nih.gov/mesh`

3.3.4 Combining Results of Different Strategies

As discussed in the previous sections, there are several ”individual” techniques that can leverage the different kinds of information encoded in ontology entities. However, once results provided by the different strategies are available it is necessary to combine them in order to obtain a final indicator of the similarity between entities. This problem has been discussed in Berkovsky et al. [8] and Do and Rahm [40]. We will discuss these aspects in more detail in Chapter 5 where we describe our approach to overcome this issue.

In general, a common approach to combine multiple results consists in adopting a combination strategy relying on a set of parameters (e.g., weights, thresholds) which are manually assigned by the user (e.g., [21, 134]) or automatically determined (e.g., [183, 77]). In particular, the results provided by each individual matcher are weighted and those that do not exceed a give threshold are discarded. Other approaches, try to ”learn” weight assignment by exploiting machine learning techniques [79]. Besides, as the combination of results provided by different strategies includes some degree of uncertainty, some approaches have been proposed (e.g., [119, 176]) which make use of theories such as the Dempster Shafer theory of evidence [156] to take uncertainty into account.

3.4 Ontology Mapping Systems

In this section we report on some initiatives proposed to face the OMP. Our analysis will proceed by classifying the analyzed systems in one of the following three categories: (i) *offline*; (ii) *user oriented* and; (iii) *online*, on the basis of their peculiar features. Of course our classification is not meant to be ”exact” as a system may show characteristics belonging to more than one of the identified categories. Our categorization solely has the aim to help us to identify some missing requirements in current mapping solutions thus motivating the work discussed in the next part of this thesis. For a complete report on existing mapping solutions refer to [56, 29].

3.4.1 *Offline* Systems

In this category, we classify general ontology mapping systems where general means that they neither exhibit specific features nor address specific issues beyond mapping discovery.

OWL-Lite Alignment System (OLA)

OLA [55] is an ontology matching system dedicated to the alignment of ontologies expressed in OWL, with an emphasis on its restricted dialect called OWL-Lite. OLA aims at taking into account and balancing the different contributions of the different ontology components (e.g., classes, constraints, data instances). The process of computing similarity between ontology entities is supported by a graphical representation of the ontology structure (i.e., the OL-Graph) on which, string distance, lexical and structural distance are defined. String distance is computed by the edit distance metric while semantic similarity exploits the WordNet [112] lexical database and in particular a depth-sensible metric. In more detail, the similarity between nodes of the graph considers both their category (e.g., class, property) and specific features such as class properties, superclasses, subclasses and so forth. The system relies on a system of quasi-linear equations to express similarity between nodes. Similarity values are finally computed as the fixed point of an iterative approximation process which starts with the lexical similarity measure and gradually brings in contributions from functions comparing the structures of ontologies. The system also features a GUI to support the visualization of the ontologies and assist the user in the mapping process.

Human Centered ONtology Engineering (HCONE-merge)

HCONE-merge [91] is an ontology merging system. HCONE-merge makes use of the intended informal meaning of concepts by mapping them to WordNet senses through the Latent Semantic Analysis (LSA) method. In particular, authors have explored the level of human involvement required for mapping concepts of two ontologies to their intended meanings. In this respect, a series of methods for ontology mapping (toward merging) with varying degrees of human involvement have been proposed. The system supported by the reasoning services of Description Logics automatically aligns and then merges ontologies. Basically, for a given concept c, the system finds word senses lexicalized by c or its variations by looking up WordNet and expands the word senses by taking into account hyponymy relations. The semantic space for c is represented by a $n \cdot m$ matrix that comprises the n most frequently occurred terms in the vicinity of the m word senses. On the basis of the semantic space corresponding to c, LSA is used to find the best word sense associated with a query string using the terms in the vicinity of c (i.e., *subclasses* and *superclasses*). As the same authors stated, the current implementation of the HCONE-merge can not be considered for use for fine-grained domain ontologies since highly technical terms do not have an entry to WordNet resulting to the poor performance of the method.

ONION

The ONION system [115] is the successor of SKAT (Semantic Knowledge Articulation Tool) [117], a rule-based system that semi-automatically discovers mappings between two ontologies. ONION discovers ontology mappings among multiple ontologies with the aim to provide a uniform query answering mechanism over the considered ontologies. ONION resolves terminological heterogeneity between ontology entities and produces articulation rules for mappings by the articulation generator (ArtGen). A set of mapping heuristics are exploited to assign a similarity score to each pair of terms in the involved ontologies. If the similarity score is above a threshold then ArtGen suggests the articulation rule to the user which can either accept, modify or delete the suggestion. A structure-based matcher investigate for additional mappings to that generated by ArtGen by looking for structural isomorphisms between subgraphs of the ontologies. Moreover, an inference-based matcher generates matches based on the rules available with ontologies or any seed rule provided by experts. Multiple iterations are required for generating semantic matches between ontologies. The system also features a GUI to assist the user in the mapping process.

iMapper

iMapper [163] supports the process of semiautomatic ontology mapping by relying on the idea of semantic enrichment, i.e. using instance information to enrich the original ontologies and calculate similarities between their elements. The system discovers mappings between concepts and between relations and is based on Referent Modeling Language, which is an Extended ER-like (Entity Relationship) graphic language with strong abstraction mechanism and sound formal basis. The mapping process is split in two phases i.e., enrichment phase and mapping phase. The enrichment phase is based on analysis of the extension information the ontologies have. To this aim, authors exploit documents that are associated with the concepts in the ontologies. An automatic document assignment strategy assigns documents to one or more predefined categories based on their content. Authors exploit a linguistic-based classifier CnS (Classification and Search) to associate documents with one or more ontology concepts. Document association is a semi-automatic process, where users need to manually adjust the assignment results to guarantee the correct assignments. The output of this phase are representative vectors (one for each concept) built from the textual content of their associated documents. In the second phase (i.e., mapping phase) similarities between ontology elements are computed as the cosine between their representative vectors. Further refinements are employed to re-rank the results via the use of a semantic relatedness metric defined on WordNet. The computed relatedness will be amplified by a tuning parameter and then will be added to the similarity values computed in the previous step. The changing on similarity values will change the ranks of the involved mappings.

GLUE

GLUE [41] is a matcher that exploits information encoded in instances to match taxonomies. GLUE relies on machine learning techniques and quantifies similarity among concepts of the two taxonomies in terms of the joint probability of distribution of the instances. GLUE uses multiple learning strategies, each of which exploits a different type of information either in the data instances or in the taxonomic structure of the ontologies. In particular, there are three learners: Content Learner, Name Learner and Meta-Learner. The Content Learner exploits the frequencies of words in the textual content of an instance to make mapping predictions. This learner does not handle attributes directly; rather, they and their values are included in the textual content of instances. The Content Learner employs the Naive Bayes learning technique as text classification method. The Name learner is similar to the Content Learner, but makes predictions using the full name of the input instance, instead of its content.

The full name of an instance is the concatenation of concept names leading from the root of the taxonomy to that instance. The Meta-learner assigns to each base learner a learner weight that indicates how much it trusts that learner's predictions. Then it combines the base learners' predictions via a weighted sum.

Quick Ontology Mapping (QOM)

QOM [45] is an evolution of the NOM (Naive Ontology Mapping) system. NOM relies on a parallel composition of individual matching strategies which exploits rules (the system currently provides 17 rules) derived from the analysis of structural and linguistic information. QOM mainly represents an optimization of NOM and tries to tackle computational problems in current mapping solutions. Authors defined a framework on which the computational complexity of different mapping systems is analyzed. The main claim of QOM is that a loss in the accuracy of the mapping process can be tolerate if the elapsed time to compute it is significantly reduced. The system, for instance, avoids to compute pairwise comparison of all the entities in the two ontologies to be mapped in favor of an incomplete strategy only focusing on promising mapping candidates. The results produced by the individual matching strategies (the same as NOM) are elaborated by using a sigmoid function, which emphasizes high similarity values and deemphasizes low one, and then aggregated through a weighted sum. A threshold is finally used to produce final results.

Ontology Mapping ENhancer (OMEN)

OMEN [116] is the implementation of a framework designed to improve existing ontology mappings using a Bayesian Network. The underlying idea is that if a mapping between two concepts from two ontologies is known, then it can be used to infer mappings between related concepts. The Bayesian network uses a set of meta-rules based on the semantics of the ontology relations that express how each mapping affects other related mappings. Nodes in the Bayesian graph are matches between pairs of classes or properties from different ontologies. Hence, the Conditional Probability Tables (CPTs) that represent how a probability distribution in one node in the BN graphs affects the probability distribution in another node downstream from it are constructed. However, if a node for all possible pairings of concepts in two ontologies is created, the number of nodes in the BN-graph grows quadratically with respect to the number of nodes in the source ontologies. To cope with this issue, the BN-graph is pruned. In particular, all possible nodes in the BN graph that are at a maximum distance of k from an evidence node are generated. The value of k is tunable by the expert running the system. However, as discussed in [116] a small value like $k = 1$ or $k = 2$ suffices, since larger values of k make very little difference to the result but increase the size of the Bayesian Net significantly. Another factor that effects the size of the BN-graph is the number of parents (i.e., nodes that influence the match) that each node has. In this respect, authors propose to choose 10 parents by selecting the top 5 parents with the maximum *a priori* probability and the top 5 parents with the minimum *a priori* probability.

3.4.2 *User oriented* Systems

In this category of mapping systems we encompass those which have been designed with particular attention to the end-user. In particular, these can either provide specific user supports such as GUIs or help him/her in some (sub)tasks of the mapping process as for instance by suggesting parameter values.

PROMPT

PROMPT [126] is a Protegé plugin that supports various tasks for managing multiple ontologies, including ontology mapping and merging. In its original formulation PROMPT works as follows. First, it creates an initial list of matches based on class names. Then the following cycle happens: (i) the user triggers an operation by either selecting one of PROMPT's suggestions from the list or by using an ontology-editing environment to specify the desired operation directly; and (ii) PROMPT performs the operation, automatically executes additional changes based on the type of the operation, generates a list of suggestions for the user based on the structure of the ontology around the arguments to the last operation, and determines conflicts that the last operation introduced in the ontology and finds possible solutions for those conflicts. Thereof, PROMPT presents a set of candidate mappings to the user. A user can examine the mappings, create new mappings, and save the correct ones.

CogZ

CogZ [57] is a PROMPT user-interface developed for addressing some of the missing user requirements in PROMPT. In particular, after a first study on cognitive support for OM discussed in [57] authors propose a first version of CogZ. This version attempts to provide user support for reducing the complexity of a mapping process and improve user interaction by allowing visualization of the contexts of ontology entities and improving incremental navigation. In [58] the authors propose a detailed cognitive framework architecture for ontology mapping. In particular, this framework is based on four ”dimensions” for each of which a set of requirements are identified. In order to fulfill these requirements, the original CogZ is extended. Besides, as CogZ works as an extension to PROMPT, it can harness the features of PROMPT and enhance or support them with additional visual components. The CogZ's plugin architecture also allows any algorithm plugin to indirectly benefit from the cognitive support provided by CogZ. Fig. 3.3 shows an example of CogZ's mapping interface. Mappings are shown as edges drawn between the ontology trees. The view also displays a mapping annotation that can be used by users to explain why they chose to map two terms. Temporary mappings are displayed as dashed lines between the source and target terms. The view supports semantic zooming or ”fisheye” selection to highlight the current focus. The semantic zooming also effectively displays cases of multiple inheritance, as shown in the figure.

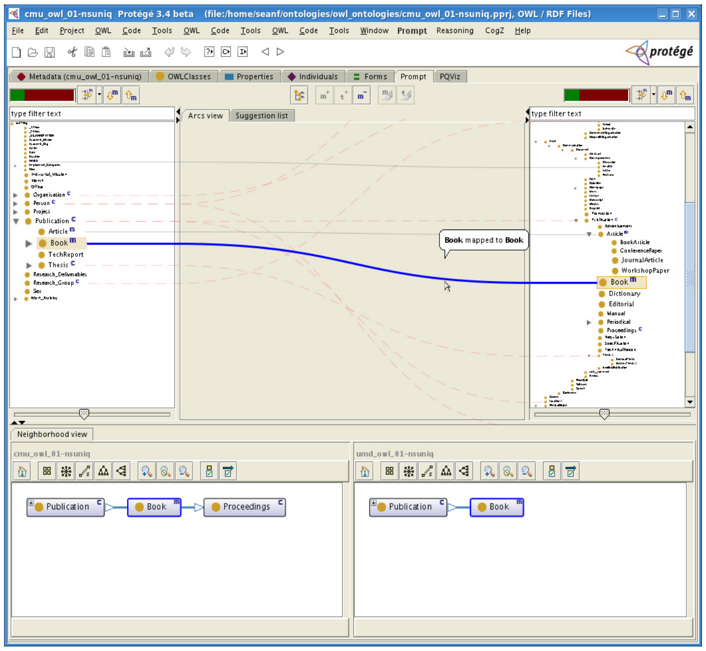


Fig. 3.3. A snapshot of the CogZ GUI

KitAMO

In the context of evaluating mapping strategy, worthy of mention is the work done by Lambrix et al. in [93]. The authors investigate the mapping problem in the biological domain and present an implementation of their framework in the KitAMO system which is related to the SAMBO system proposed by the same authors. KitAMO aims to perform comparative evaluation of ontology alignment strategies and their combinations. It supports the study, evaluation and comparison of alignment strategies and their combinations based on their performance and the quality of their alignments on test cases. It also provides support for the analysis of the evaluation results. The system provides different GUIs allowing the mapping strategies to be chosen and visualize produced mappings. Moreover, it also performs performance evaluation, in terms of elapsed time, of the different strategies.

Falcon-AO

Falcon-AO [83] is an automatic tool for aligning ontologies. There are two matchers integrated in Falcon-AO: one is based on linguistic matching for ontologies, called LMO; the other is a matcher based on graph matching called GMO [179]. In Falcon-AO, GMO takes the alignments generated by LMO as external input and outputs additional alignments. Reliable alignments are gained through LMO as well as GMO according to the concept of reliability which makes Falcon-AO one of the few systems attempting to address the problem of parameter tuning. The reliability is obtained by observing the linguistic comparability and structural comparability of the two ontologies being compared. In more detail, the linguistic comparability (LC) is defined as follows:

$$LC = \frac{M}{\sqrt{\#O_1 \cdot \#O_2}}. \quad (3.1)$$

where M denotes the number of entity pairs with similarity larger than c where c is an experience value; $\#O_1$ and $\#O_2$ represent the number of named entities in O_1 and O_2, respectively.

The structural comparability is determined through comparing the occurrences of built-in properties used in the two ontologies to be mapped. The built-in properties are RDF, RDFS and OWL built-in vocabularies used as properties in triples (e.g. *rdf:type, rdfs:subClassOf and owl:onProperty*). The occurrence of these properties are represented as vectors in the space of all properties. Then, the structural comparability is computed as the cosine of these vectors.

The two comparability coefficients, apriori determined, are exploited a posteriori, that is, after executing the mapping task as they serve to weight the results produced by the linguistic and structural matching strategies.

RiMOM

RiMOM is an ontology mapping system combining multiple techniques. In its original formulation [165] it was mainly devised with the aim to minimize risk in ontology mapping discovery. More recently the algorithm [102] has been extended in different ways. First, it has been endowed with a structural matching strategy which is an adaptation of the similarity flooding algorithm [111]. Second, it addresses the problem of automatic strategy selection, that is, how to choose the strategies to be used in aligning ontologies. In order to do this, RiMOM relies on two affinity coefficients, that is, the structure (F_SS) and label (F_LS) similarity coefficients that are determined by scrutinizing the two ontologies. In more detail, these coefficients are defined as follows:

$$F\_SS = \frac{\#common\_concepts}{\#max(non\_leaf\_c_1, \#non\_leaf\_c_2)}. \quad (3.2)$$

$$F\_LS = \frac{\#same\_label}{max(c_1, c_2)}. \quad (3.3)$$

where $\#nonleaf\_c1$ indicates the number of concepts in O$_1$ that have sub concepts. Likewise for $\#nonleaf\_c2$. The $\#common\_concepts$ coefficient is calculated as follows: if concepts c_1 in O$_1$ and c_2 in O_2 have the same number of sub concepts and they are in the same depth from the concept ”owl:Thing”, $\#common\_concepts$ is incremented. As for F_LS, the value of $\#same\_label$ contains the number of pairs of concepts having the same label.

These coefficients are exploited to determine apriori the mapping components to be involved in the process of mapping discovery.

3.4.3 *Online* Systems

Under the umbrella of *online* mapping systems we classify those systems which are particularly suitable to discover mappings in open and dynamic environments such as P2P networks. In these environments, the OMP assumes a specific connotation as mappings have to be discovered on the fly and mapping discovery should only take into account specific portions of the two (peer) ontologies. Therefore in this context, it is needed to map only the request with the relevant parts of the ontology to which the request is addressed. Moreover, it is not possible for each request (e.g., related to a semantic query), to map two entire ontologies as peers are unaware of one another's ontologies.

To date, few approaches have addressed this particular instance of the OMP. In a recent book on ontology mapping [56], only a few systems with these capabilities are cited. We will discuss in more detail this problem and propose an approach to tackle it in Chapter 6.

H-Match

H-Match [21, 23, 22] is an algorithm to dynamically match concepts in distributed ontologies. It was designed to enable knowledge discovery and sharing within the Helios P2P framework [24]. In order to perform mapping discovery, the algorithm exploits an internal representation of OWL ontologies, that is, the H-Match model. The system computes two kinds of ”affinity” coefficients, that is, lexical and contextual affinity. Moreover it also checks datatype compatibility. In more detail, the lexical affinity is computed by exploiting an ad-hoc thesaurus built on top of WordNet and a similarity metric between concepts in the thesaurus. The system automatically deals with compound terms that are not available from WordNet. In performing contextual affinity H-Match exploits predefined weights assigned to the different types of relations among concepts. These relations include: isa, contains, part-of, and generic. The final similarity values are obtained by weighting the contributions of both the lexical and contextual affinities with more emphasis (i.e., higher weight) on lexical affinity. Finally, results that do not exceed a given threshold are discarded. One striking features of H-Match is that it allows to configure the mapping task according to different matching models (i.e., surface, shallow, deep, and intensive) each of which involves different types of constructs of the ontology. The complexity of the matching models is increasing from the shallow to the intensive model.

CtxMatch and S-Match

CtxMatch [17] is an algorithm aimed at discovering mappings between Hierarchical Categories. This approach is based on the intuition that there is an essential conceptual difference between coordinating generic abstract structures (e.g., arbitrary labeled graphs) and coordinating structures whose labels are taken from the language spoken by the community of their users. The latter type of structures gives the chance of exploiting the complex degree of semantic coordination implicit in the way a community uses the language from which the labels are taken. This system is the first attempt toward semantic matching and has been refined in the S-Match system [65]. CtxMatch serves as semantic support to the KEEx system for Distributed Knowledge Management [14]. CtxMatch heavily relies on WordNet for interpreting the correct sense of concepts in the hierarchies defined by peers on the basis of the context in which they appear. In particular, it translate the mapping problem to a problem of logical satisfiability and gives as output semantic relations between concepts instead of numeric values. The algorithm takes into account different kinds of knowledge, that is, structural, lexical, and domain and encodes concept representations into Description Logics axioms. Hence it computes semantic relations by exploiting logic reasoners such as Pellet [4] or FaCT [5]. In a subsequent work, the system has been extended to support attributes. The main difference w.r.t S-Match is that the latter is able to perform iterative matching.

3.5 Summary

As emerged from the analysis performed in the previous sections, there are a lot of approaches addressing the OMP. Such large number of systems, however, do not cover just the same aspects of the OMP but, conversely, gives rise to several perspectives under which the problem can be considered. As an example, there are systems that mainly focus on discovering mappings while other go beyond it by providing user support or specific techniques suitable for open environments. Surely, such intense activity in the field is a motivating factor that helps a progressive improvement of mapping systems and that, hopefully, will have as final outcome the concrete exploiting of mapping solutions in real world applications.

In Table 3.3 an overall overview of the analyzed system is provided by taking into account the following aspects:

- *Context*: summarizes the objective of the system (e., mapping, merging) and states if the system has been designed to operate in a specific context (e.g., open environments).
- *Information*: reports on which kind of ontological information is exploited by the system.
- *Specific Techniques*: reports on the peculiarities of the system.

From the analysis reported in Table 3.3 several considerations can be done. First, mapping systems mainly rely on the same basic techniques to discover mappings even if each system provides at least one specific technique. This emphasizes how ontology mapping can benefit from well-established techniques devised in other research fields. Besides, the primary sources of knowledge for researchers in ontology mapping are the ”never-ending” findings produced by the database community which has being facing the schema mapping problem since several years.

[4] `http://www.mindswap.ir/pellet`
[5] `http://www.cs.man.ac.uk/~horrocks/FaCT`

Table 3.3. Comparison among mapping approaches

Category	**Mapping system**	**Context**	**Techniques**			**Specific techniques**
			Lexical	Structure	Instance	
OFFLINE	OLA	Mapping	Yes	Yes	No	OWL properties
	HCONE-merge	Merging	Yes	No	No	LSA and WordNet
	ONION	Mapping	Yes	Yes	Yes	Rules
	iMapper	Mapping	Yes	Yes	Yes	Vector Space model
	GLUE	Mapping	Yes	Yes	Yes	Machine Learning
	QOM	Mapping	Yes	Yes	Yes	Low complexity
	OMEN	Mapping	Yes	Yes	No	Bayesian Networks
ONLINE	H-Match	Mapping in open environments	Yes	Yes	Yes	Ad-hoc thesaurus
	CtxMatch	Mapping in open environments	Yes	Yes	Yes	SAT based
USER ORIENTED	PROMPT	Mapping/Merging	Yes	Yes	No	Integrated environment
	CogZ	Mapping/Merging	Yes	Yes	Yes	User oriented
	KitAMO	Merging	Yes	No	No	Evaluation of mapping strategies
	Falcon-AO	Mapping	Yes	Yes	No	Automatic strategy weighting
	RiMOM	Mapping	Yes	Yes	No	Automatic strategy selection

As for the systems described before, we classified them in three categories to put emphasis on their respective peculiar features. In more detail in the *offline* category, systems such OMEN can be viewed as an extension of traditional mapping systems since they exploit similarity values gained by classical techniques to improve the matches or suggest additional matches. HCONE-merge is heavily-tied to the WordNet database and cannot be exploited to merge ontologies with specific knowledge. The QOM system addresses a particular aspect of the OMP, that is, performance. As for the *online* category, we can see that only a few system have been devised to work in open environments. Finally, in the *user oriented* category some initiatives addresses the problems of user support in terms of GUIs or strategy suggestion separately.

3.5.1 Pointer to Next Chapters

In the light of the analysis performed in the previous section we can identify the following aspects to be taken into account in designing a novel and comprehensive mapping system:

- How to exploit in the best way all the information encoded in ontology entities? In some cases mapping solutions underestimate the importance of some sources of information. For instance, most of the existing approaches only exploit entity names to gain similarity. However, further information such as comments, labels can be considered in assessing similarity.
- How to design mapping systems to work in open environments? In this case it is necessary to devise specific techniques taking into account the requirements of quickness and the limited ontological information exploitable to discover mappings.
- How to provide users with off-the-shelf tools in order to enable mapping systems to become a commodity even outside research labs? In this respect it is necessary to develop more user-friendly tools which on one side have to be more charming to be used and on the other side should avoid the user the burden to perform error-prone tasks such as parameter tuning.

In the next part of this thesis (see chapters from 4 to 6) we address these aspects by describing three new approaches to ontology mapping.

Part III

New Approaches to Ontology Mapping

4

The Lucene Ontology Matcher

Current approaches to ontology mapping (OM), as seen in Chapter 3, share common characteristics. They leverage the different types of information encoded in ontologies to find out mappings between entities of a *source* and a *target* ontology. In particular, OM algorithms are composed by a set of individual matchers each of which focuses on a particular kind of ontological information. Typically, a linguistic matcher compares linguistic features of ontology entities to find out an initial set of possible mapping candidates (e.g., [41, 115]). Afterward, this set is passed to a structural matcher that refines results and tries to discover new mappings by exploiting relations among entities. Some OM algorithms also include an instance-based matcher whose aim is to discover mappings by exploiting information encoded in ontology instances [41]. Although structural matching is important, it is basically supported by linguistic matching. Moreover, carefully scrutinizing the results of a recent OM initiative [54] we find out that mappings between ontologies are mostly discovered by linguistic matching techniques.

To date, even if several linguistic matching techniques are available, many of these do not adequately consider all the linguistic information encoded in ontologies. They only leverage string comparison or thesaurus based techniques to compare the local names of ontology entities without taking into account other important sources of linguistic information such as labels, comments, instances and other annotation properties. Thus, the design of accurate techniques for ontology mapping able to exploit all the linguistic information encoded in ontology entities is mandatory. In this chapter we introduce a new linguistic matcher based on the Lucene search engine library called LOM (Lucene Ontology Matcher). The LOM rationale is to gather different kinds of linguistic information of entities of a *source* ontology into Lucene documents stored into an index. Hence, mappings are obtained by exploiting values of the entities of the *target* ontology as search arguments against the index created from the *source* ontology. In particular, similarities between documents into the index and queries (translated into documents) are computed by exploiting the scoring schema implemented in Lucene.

The rest of this chapter is organized as follows. Section 4.1 provides a background on the Lucene search engine library. Section 4.2 thoroughly describes the

LOM architecture. Section 4.3 extensively evaluates the system on the OAEI 2006 benchmark test suite [1] and compare it with other linguistic approaches. Finally, Section 4.4 concludes.

4.1 Background on the Lucene Search Engine Library

Lucene [2]is a high-performance and scalable information retrieval library through which any piece of data converted to a textual format can be indexed and made available for search. Indexing with Lucene includes three main phases:

1. Converting data to text: in this phase Lucene converts into textual format data (e.g., pdf, doc, ppt, xls documents) by exploiting appropriate parsers.
2. Analyzing the text: in this phase stop words are eliminated and words are stemmed.
3. Saving the text into an index. Lucene can create two types of indexes: one maintained in main memory, called RAMDirectory, and the other maintained on the hard disk, called FSDirectory.

Data into the index is stored in the form of documents. A Lucene Document (LD) consists of a collection of fields. Each field, identified by a name, contains a piece of data that is either queried against or retrieved from the index during search. Search in Lucene is performed by specifying one or more keywords and one or more field to search within. Search results (on the form of LDs) are collected within objects called hits. Each LD contained in a hits, has associated a score value indicating its similarity w.r.t the search key.

4.1.1 The Lucene Scoring Schema

Lucene is an information retrieval library founded on the concepts of document and field. Following a classical information retrieval approach [6, 148], when performing a search the corresponding query is translated into a document that will be compared to documents stored into the index to obtain a final score value as output. Lucene scoring schema is based on the Vector Space Model [149] of information retrieval. In general, the idea behind the vector space model is that, the more times a query term appears in a document, relative to the number of times the term appears in all the documents in the collection, the more relevant that document is to the query. In Lucene, the score of a query *q* for a document *d* correlates the cosine-distance or dot-product between document and query vectors in a vector space. A document whose vector is closer to the query vector is scored higher. The score is computed as follows:

$$sim(q,d) = coord(q,d)\cdot queryNorm(q)\cdot \sum_{t\,in\,q}(tf(t\,in\,d)\cdot idf(t))^2\cdot t.getBoost()\cdot norm(t,d). \tag{4.1}$$

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

[2] http://lucene.apache.org

In the following we provide an overall description of the components of this formula: *tf(t in d)* is the term frequency, defined as the number of times a term *t* appears in the current scored document *d*. Documents that have more occurrences of a given term receive a higher score. Lucene by default computes the *tf* value as:

$$tf(t\,in\,d) = \sqrt{frequency}. \tag{4.2}$$

df(t) stands for Inverse Document Frequency. This value correlates *t* to the inverse of *docFreq* (i.e., the number of documents in which the term *t* appears).

$$idf(t) = 1 + log(\frac{numDocs}{docFreq + 1}). \tag{4.3}$$

queryNorm(q) is a normalizing factor used to make scores between queries comparable. This factor does not affect document ranking. By default it is implemented as:

$$queryNorm(q) = \frac{1}{\sqrt{sum\,of\,squared\,weights}}. \tag{4.4}$$

The sum of squared weights (of query terms) for a Boolean query takes the following form:

$$sum\ of\ squared\ weights = q.getBoost()^2 \cdot \sum_{t\,in\,q}(idf(t) \cdot t.getBoost())^2. \tag{4.5}$$

t.getBoost() is a search boost of the term t in the query *q* as specified in the query text. When a document is added to the index, the *norm (t, d)* is computed as follows:

$$norm(t) = doc.setBoost() \cdot lengthNorm(field) \cdot \prod_{field\ f\ dubbed\,as\,t} f.getBoost. \tag{4.6}$$

where: *doc.setBoost()* is used to calculate the boost of the document before adding it to the index and *lenghtNorm(field)* is computed in accordance with the number of tokens of the field in the document, so that shorter fields contribute more to the score.

4.2 The LOM Architecture

Current mapping approaches often underestimate linguistic information of ontology entities as they do not consider all the sources of linguistic information. We face this issue by implementing LOM. Through LOM we aim at collecting and exploiting all the linguistic information (e.g., names, comments, labels) of entities by collecting it into Lucene documents (LDs). Depending on the type of ontology entity two different kinds of LDs are constructed. If e_s is an entity corresponding to a class or a relation, the associated LD has the structure shown in the top part of Fig. 4.1. In this case the LD is composed of three fields:

1. Name: contains the collection of words in the local name of e_s.

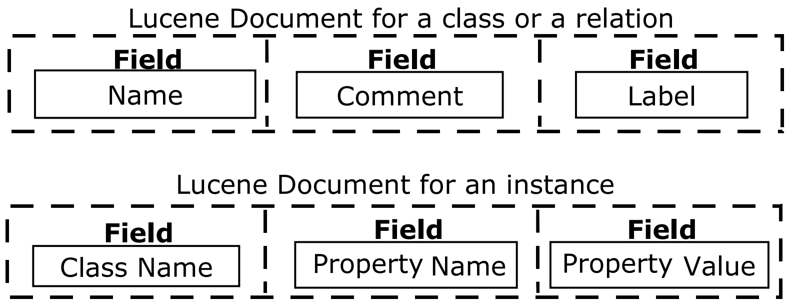


Fig. 4.1. Lucene documents exploited by LOM

2. Comment: contains the collection of words in the *rdfs:comment(s)* of e_s.
3. Label: contains the collection of words in the *rdfs:label(s)* of e_s.

Conversely, if an entity describes an instance, the corresponding LD has the structure shown in the bottom part of Fig. 4.1. In this case the LD is composed of the following three fields:

1. Class Name: contains the local name of the class to which the instance belongs.
2. Property Name: contains the name of an attribute of the instance.
3. Property Value: contains the value of the attribute specified in Attribute Name.

By exploiting these two types of LDs, we build an index of type RAMDirectory that contains a LD for each entity e_s defined in a *source* ontology O_s. We use the RAMDirectory index since in our experimental evaluations we found that this approach is ten time faster than the FSDirectory approach. Ontology mappings are derived by using the values of the entities e_t of a *target* ontology O_t as search arguments against the index built from O_s. The overall architecture of LOM is depicted in Fig. 4.2. The *Matcher M* (dotted area in Fig. 4.2) takes an input of the following form:

$$Ipt = < S, T, Th, w_n, w_l, w_c, w_p >$$

where: S is the set of entities in the *source* ontology O_s, T is the set of entities in the target *ontology* O_t. The parameters Th, w_n, w_l, w_c and w_p are a threshold and a set of weights used to weight the contribution of name, label, comment and property respectively. In order to assess similarity between entities, depending on the type of entity e_t to use as input of a query, two kinds of search can be issued. The first kind, referred to as s_1, aims at assessing similarity by exploiting linguistic information of classes and relations. In this case, for each field (i.e., name, comment, label) that composes the LD of a class or relation a query is constructed which takes as argument the corresponding value of name, comment or label of e_t. These queries are indicated as q_1, q_2 and q_3 respectively. The scores of these queries are weighted according to the parameters w_n, w_l, w_c to obtain an overall similarity score. The second type of search, referred to as s_2, aims at assessing similarity between classes and relations by exploiting the entities that represent instances. For each attribute value of an instance,

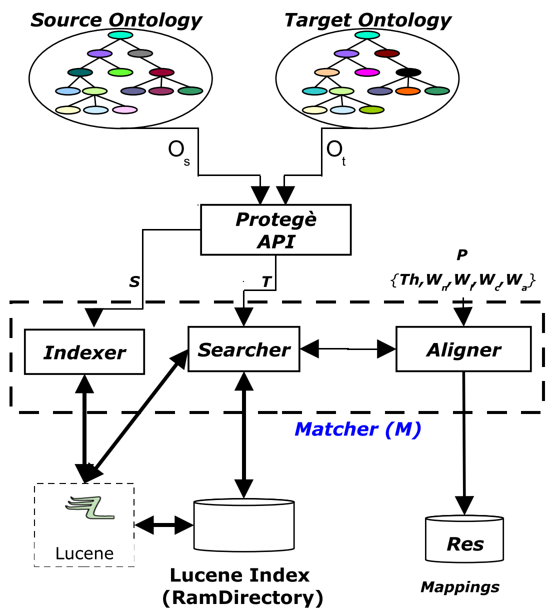


Fig. 4.2. The LOM architecture

LOM searches against the *Property Value* field of the LDs stored in the index. Note that since instances are related to concepts, the similarity values obtained in this way are summed up to these obtained by exploiting the first type of query. In this case the results of s_2 are weighted according to the value of the w_a parameter. Similarity values among elements in *S* and *T* are represented in a similarity matrix. If the overall similarity between two entities e_t and e_s is greater than a fixed threshold (i.e., Th), they are considered mappable and therefore included in the set of results (i.e., *Res* in Fig. 4.2).

Overall, after parsing O_s and O_t by the Protégé API [3] the two sets *S* and *T* are built. The set *S* feeds the *Indexer* module which is responsible for constructing and indexing each e_s as a LD. Entities contained in the set *T* derived from O_t, are used to search against the index through the *Searcher* module. The *Aligner* module collects and weights results according to the parameters in *P* of the specific matching task. Therefore, it filters the results and constructs alignments that will be included in the set *Res*. Alignments in *Res* are created according to the format described in [49].

4.3 Evaluation

In order to show the suitability of LOM, we evaluated it on the popular Ontology Alignment and Evaluation Initiative (OAEI) benchmark test suite [4]. This way we

[3] `http://protege.stanford.edu/plugins/owl/api`
[4] `http://oaei.ontologymatching.org`

Table 4.1. Parameters of the different types of queries

Weight	Field	Type of Search
w_n	Name	s1 (query q1)
w_l	Label	s1 (query q2)
w_c	Comment	s1 (query q3)
w_a	Attribute Value	s2

obtained an evaluation of the strengths and weakness of the algorithm as compared to current linguistic techniques for OM.

4.3.1 The Ontology Alignment and Evaluation Initiative

The OAEI aims at establishing a consensus for evaluating the different methods proposed for ontology mapping. The goal of the OAEI 2006 benchmark series is to identify the areas in which each ontology mapping algorithm is strong or weak. Tests are based on one particular ontology defined in the bibliography domain (referred to as 101 in the rest of this chapter), and some variations of such ontology for which alignments are provided. The benchmark is composed by five groups of tests that are constructed on the basis of different types of alterations.

- Group 1 (G1) (ontologies from 101-104): in this group of tests the reference ontology (i.e., 101) has to be compared with ontologies that have exactly the same classes and relations names or completely different ones (i.e., ontology 102).
- Group 2 (G2) (ontologies from 201-210): in this set of tests the reference ontology has to be compared with ontologies in which linguistic features have been altered. In this case the structure of ontologies has not been altered.
- Group (G3) (ontologies from 221-247): linguistic features have been replaced with randomly generated characters.
- Group (G4) (ontologies from 248-266): in this case both linguistic and structural features have been suppressed. In particular linguistic features have been replaced with randomly generated characters.
- Group (G5) (ontologies from 301-304): ontologies in this group are real-life ontologies.

Traditional Approaches to Linguistic Matching

In this section we present four approaches that we have implemented in order to compare LOM against. We implemented two string-based techniques, a linguistic approach that exploits WordNet and two approaches in which we combine results from LOM with a string-based approach and the linguistic approach respectively.

String based approaches

I-Sub proposed in [161] is a metric for string comparison which computes similarity between two strings by considering their commonalities as well as their differences. In the rest of this section, this approach is referred to as I-Sub. Edit distance string based metric is one of the most commonly used to gain linguistic similarities in ontology matching. We give an implementation based on Levenshtein's edit distance [98]. This approach is referred to as EditDist.

A WordNet based approach

As pointed out in Chapter 3 several approaches exploit WordNet [112] for assessing semantic similarity among entities of different ontologies. Through these approaches the problem of discovering synonymy among terms can be solved. We implemented the Jiang and Conrath metric [84]. In our evaluation this approach is referred to as WN.

Combining different approaches

In order to consider the benefits from combining LOM with other linguistic techniques we propose two combination strategies. The first, referred to as *C1*, aims at combining results of LOM with these of the I-Sub approach. The second, referred to as *C2*, combines result of LOM with these of WN.

Parameter setting

The values of parameters of LOM used in all experiments are shown in Table 4.2. These values are the optimal values deriving from our experimental evaluation.

Table 4.2. Parameter values used in the evaluation

Parameter	**Value**
T_h	0.51
w_n	1
w_l	0.33
w_c	0.8
w_a	0.6

4.3.2 Experimental Evaluation

In the first experiment, we computed the average (per test) execution time. In the adopted test environment (Intel Pentium 4 3.0 GHz processor with 1Gb of RAM and Windows XP), we observed the average execution times (considering the entire set

Table 4.3. Elapsed times

Approach	Average Elapsed Time (s)
I-Sub	1.73
EditDist	1.69
WN	6.8
LOM	1.47
C1	2.11
C2	7.21

of tests) reported in Table 4.3. As can be noted LOM performs very well in terms of execution time as compared to the other approaches. The approaches based on WordNet (i.e., WN and C2) are the most expensive in terms of time since they have to deal with the complex structure of the WordNet database.

Fig. 4.3 shows the results on the five groups of tests in terms of Precision. As

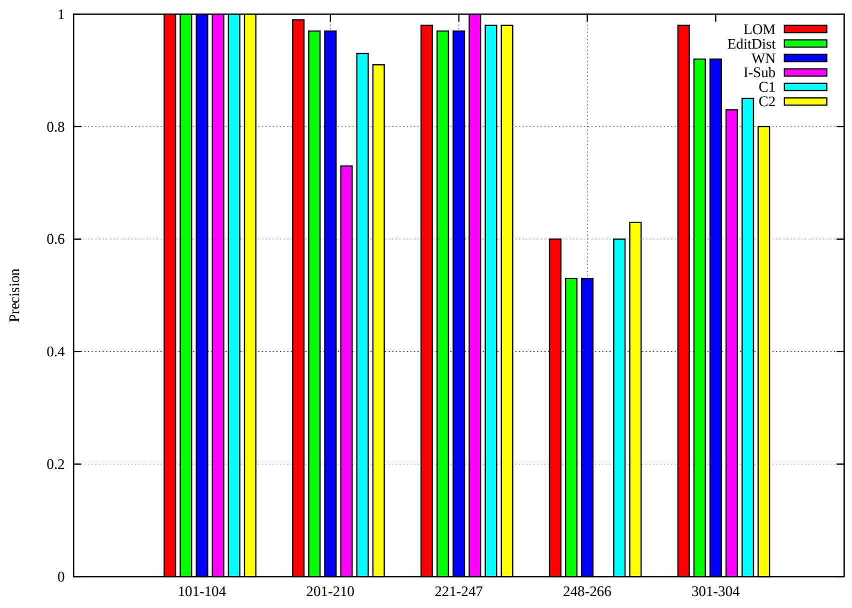


Fig. 4.3. Average Precision values per test group

can be noted, LOM is precise where there are few alterations in linguistic features of ontologies, that is tests 101-104 and tests 221-247. Whereas, it is not the best one in tests 248-266 where the structure of ontologies is altered and linguistic information are mostly suppressed. However, in real case ontologies, that is, tests 301-304, it dominates over the other approaches. Fig. 4.4 shows the Recall results. Also in this case, LOM performs well in tests 101-104 but it is outperformed in tests 201-

210 by C1. In tests 221-247 results obtained by the different approaches (apart from WN) are almost identical. The WN approach does not perform well since in the ontologies included in the tests there are many terms not defined in WordNet. In

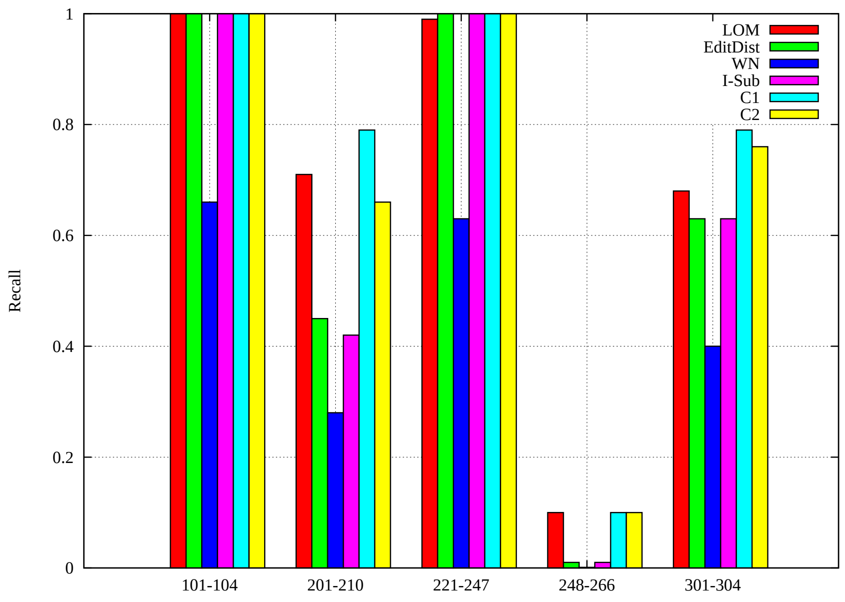


Fig. 4.4. Average Recall values per test group

tests 248-266, WN cannot find any mapping since local names are for the most part altered. Conversely, in the test 205, where ontology terms are substituted by synonyms, WN shows the best results in terms of Recall obtaining 0.40 while I-Sub, EditDist and LOM obtained 0.24, 0.24 and 0.35 respectively. Fig. 4.5 shows results of F-Measure. We can observe that in test 101-104 and 221-247 our approach is very close to string-based approaches and combinations C1 and C2. In test 201-210 our approach dominates over string-based approaches. The combination strategy C1, that includes results of LOM and I-Sub, outperforms the single LOM approach; however the combination C2 is outperformed by LOM. In tests 248-266 that are the most difficult ones since the structure of the reference ontology is modified and linguistic information is altered, our approach is able to find some correct mappings and dominates over string-based approaches. The found mappings are derived thanks to the presence of ontology instances that are exploited by LOM to derive similarity between classes they belong to and properties defined in such classes. The combination strategy C1 obtains the same results while the C2 strategy obtains better results. Finally, in tests 301-304 that include real life ontologies, LOM shows nice values and it performs slightly better than I-Sub. In such group of tests, C1 is the approach that performs better. We also performed our experiments on the overall set of tests as

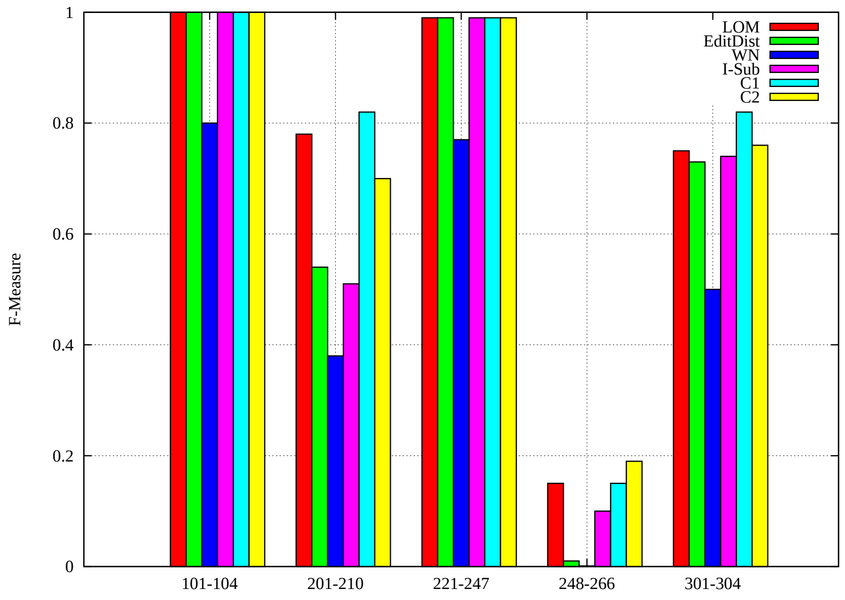


Fig. 4.5. Average F-Measure values per test group

shown in Fig. 4.6. According to these results LOM achieved very good results and in particular it dominates over string based approaches. It performs a bit worse than C1 and C2 in terms of Recall and F-Measure. Note that in our experiments the combination of LOM with other approaches could bring some improvements. However, there are at least two drawbacks:

1. The average execution time increases almost 40% for C1 and almost 500% for C2 as can be seen in Table 4.3.
2. In some experiments, the Precision noticeably decreases as for example in tests 301-304 (see Fig. 4.3) and in the whole set (see Fig. 4.6).

In the light of these observations we can conclude that LOM obtains very good results when there are no alteration in linguistic features of ontologies (names randomized or comments and instances removed) and it outperforms other techniques in terms of execution time. The combination of LOM with other approaches noticeably affects performance in terms of elapsed time.

4.4 Discussion and Lesson Learned

In this work we investigated linguistic techniques for ontology mapping pointing out how these are important to support other mapping techniques. Most of current linguistic approaches mainly compare the local name of ontology entities but underestimate the importance of other sources of linguistic information such as comments,

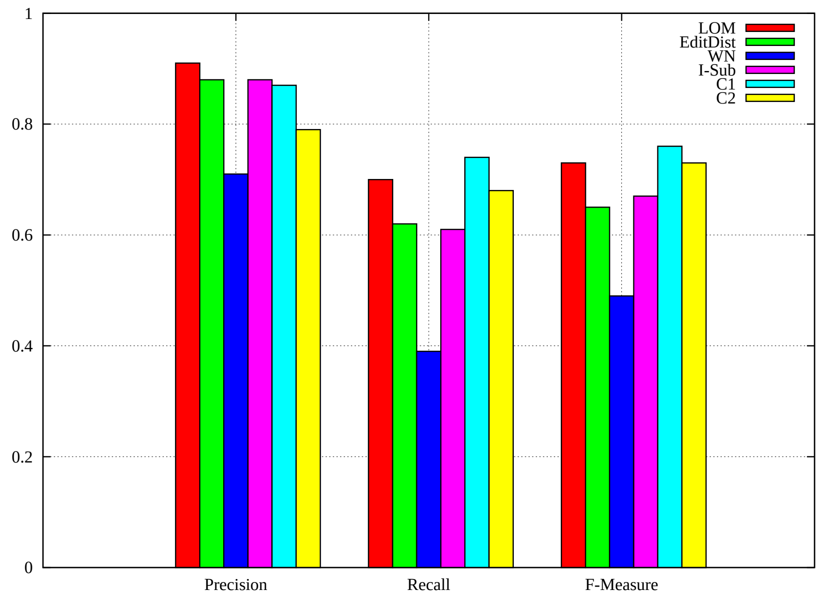


Fig. 4.6. Average Precision, Recall and F-Measure on all tests

labels and instances. In order to overcome this pitfall, we developed an ontology mapping system called LOM exploiting the Lucene search engine library. The underlying idea of LOM is to treat each ontology entity as a Lucene Document (LD) composed of a set of fields. The similarity between entities of different ontologies is computed by constructing an index of LDs derived from a *source* ontology and using values of entities of a *target* ontology as search arguments against that index. An extensive evaluation of LOM has shown its appropriateness as compared to other linguistic approaches.

5

The User-Friendly Ontology Mapping Environment

As emerged from Chapter 3, even if several approaches to tackle the Ontology Mapping Problem (OMP) have been proposed, they underestimate important functional requirements. Firstly, there are no systematic ways to suggest which mapping strategy has to be adopted or how parameter values of the different mapping components have to be set. Parameter tuning is a skill-and-time-intensive process and is only possible in the presence of the ”ground truth” mapping; parameter values that achieve good results in a domain cannot apply in another. Secondly, it is not possible to easily compare and evaluate mapping strategies and their combination in terms of both accuracy and performance. However, such a feature will be useful in investigating how mapping strategies can be exploited in the best way. Thirdly, research efforts are mainly addressed to the development of mapping techniques, underestimating the role of the user. A major problem with the tool landscape at present is that almost all of the available tools focus on only one of the relevant aspects (e.g., mapping discovery) of ontology mapping, while end users rarely focus on a single task. Therefore, an integrated and comprehensive mapping environment will be more appropriate and productive. In addition, recent research on schema matching, to which ontology mapping is related, recognized that the development of fully automatic procedures for discovering matching is still not achievable and therefore ”humans must be in the loop” [10].

In this chapter the focus is on the design and implementation of a comprehensive mapping system. Specifically, the following contributions are made:

- The OMP and related concepts are systematically defined along with the requirements of an ontology mapping (OM) framework.
- An extensible software framework with modules as building blocks has been designed. Modules are (possibly existing) mapping components serving for different purposes (e.g., discovering mappings, combining, comparing and evaluating mapping strategies).
- A mapping strategy is defined in terms of a DAG of mapping modules. The DAG specifies how information flows among modules. The topological sort algorithm is exploited to determine the correct execution order among modules.

- A *Strategy Predictor* module which suggests the most suitable mapping strategy and the related parameter values based on the affinity between ontologies, has been designed. The framework implementation resulted in the UFOme (User Friendly Ontology mapping environment) system. UFOme features have been evaluated on the OAEI 2006 benchmark set of tests with encouraging results.

The remainder of this chapter is organized as follows. Section 5.1 introduces preliminary definitions used throughout the chapter. Section 5.2 describes the requirements a mapping framework should meet; Section 5.3 presents a library of mapping components to cope with the requirements identified in Section 5.2; in particular here the novel *Strategy Predictor* is introduced. The UFOme system is described in Section 5.4 and its strategy prediction capabilities are extensively evaluated in Section 5.5; finally, Section 5.6 concludes.

5.1 A Generic Ontology Mapping System

The aim of this section is to provide preliminary definitions.

Definition 5.1 ***(Ontology Mapping System)***. *An ontology mapping system (OMS) can be defined as a tuple of the form:*

$$OMS := \langle M_s, Ipt \rangle$$

where M_s is a mapping strategy, that is, a combination of interconnected mapping components and Ipt is the input of M_s which takes the following form:

$$Ipt := \langle P, C_m \rangle$$

where P is a set parameter values (e.g., weights, thresholds) associated with the different components and C_m is an initial set of candidate mappings possibly provided by the user. C_m can be exploited as the basis for inferring further mappings as done in [124]. Current ontology mapping systems delegate to the user the responsibility to design the most suitable mapping strategy in terms of mapping components and parameter values. One of the goals of this work is to devise and evaluate a mechanism to automate such skill-intensive task. An OMS aims at discovering correspondences (aka mappings) among entities belonging to a *source* ontology O_s and a *target* ontology O_t and can be considered as a combination of mapping components which can serve for different purposes (e.g., discovering mappings, combining results).

Fig. 5.1 shows an OMS which has *(n+2)* components: *n* individual matchers, one combiner and one selector. The matcher modules produce a set of similarity values for entities in O_s and O_t. The combiner module weights similarity values produced by the different matchers. The matching selector module is used to discard mappings that have similarity values below a given threshold.

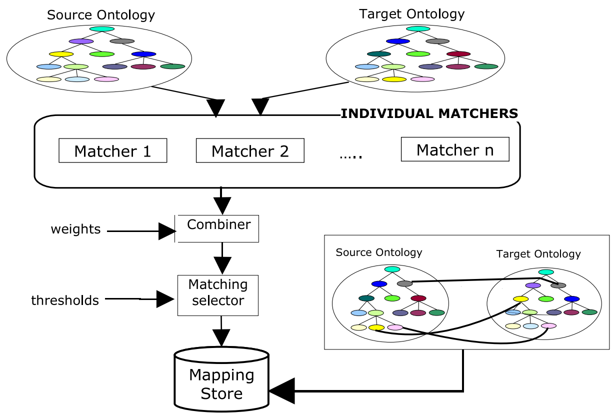


Fig. 5.1. A generic mapping system.

5.2 Requirements of an Ontology Mapping System

To date, several mapping systems are available [29], therefore users have to face the problem of choosing the most suitable one. Noy et al. in [125] provide some useful principles for designing, evaluating and choosing an OM system. In [125] the concept of "best" tool is nicely discussed. Even if the user is tempted to indicate as the best tool the one that reaches results more similar to human judgment, it is important to take into account other factors such as user support. Hence, in designing an OM framework it is not only important to investigate functional requirements, that is, "how thing have to be done", but also the role of the user has not to be underestimated. In fact, easy-to-use tools play a crucial role for the adoption of any new technology and, at the moment, OM is severely undermined by the lack of tools supporting the users. By the investigation of current mapping systems done in Chapter 3 we noticed that:

- OM systems (e.g., [46, 45, 47]) adopt a combination of different individual matching strategies to exploit the different kinds of ontological information.
- Only simple user-system interactions are supported (e.g., visualization of results as a list). There is a lack of comprehensive systems which assist users in the different phases of mapping task execution.
- There is no support for automatically designing ad hoc mapping strategies taking into account a particular instance of mapping problem. Mapping strategies are manually designed and parameter values are manually tuned. However, parameters tuning is a time-and-skill intensive process and heavily depends on the particular knowledge domain.
- Current OM systems do not have a reference architecture. They mainly provide components devoted to discover mappings underestimating other important functional requirements needed in constructing complex mapping strategies.

In the next section a set of requirements an ontology mapping framework should meet will be identified.

5.2.1 What a Mapping System Should Feature ?

This section aims at identifying some main requirements a comprehensive mapping system should meet. These requirements emerged from the analysis done in Chapter 3 and are the basis for the development of the UFOme system, described in Section 5.4. In our investigation, we found that a mapping task can be split in three main phases: (i) *designing*; (ii) *running* and; (iii) *evaluation*. For each phase a set of requirements and features is defined and detailed in Table 5.1. How the UFOme system copes with these requirements will be discussed in the following sections.

5.3 A Library of Mapping Modules

This section presents a library of mapping modules aimed at fulfilling the framework requirements identified in Section 5.2.1. In particular, the functionalities of some of these modules already exist and have just been integrated in this framework. Note that here the novelty does not consist in designing new modules devoted to discovering ontology mappings but in: (i) designing a novel *Strategy Predictor* module which assists the user in developing a mapping strategy with related parameter values; (ii) designing a set of modules to assist the user in all the phases of a mapping task execution. A module is a generic mapping component that can be included in a mapping strategy. It can be represented by the architecture depicted in Fig. 5.2.

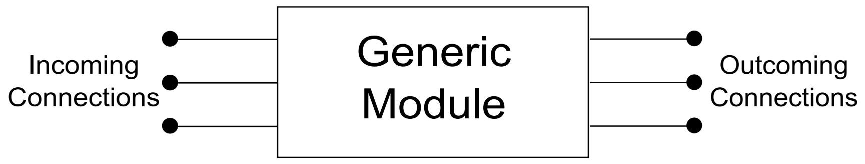


Fig. 5.2. A generic mapping module

A module can be seen as a black box having a set of incoming connections (i.e., the input) and a set of outgoing connections (i.e., the output). A module also exposes one or more configuration parameters. Overall, a module processes the input and, according to its specific aim (e.g., mapping, combination, evaluation), produces the output. We identified different categories of modules, each of which covers a specific aspect of a mapping task execution. These categories will be described in the following.

Table 5.1. Ontology Mapping Framework Requirements

Phase 1: **Designing**
Design/change a mapping strategy: this feature allows a user to design a mapping strategy. According to our definition of mapping strategy (see Section 5.1), a DAG of modules has to be constructed. In this respect, a GUI has to provide an intuitive way to design/change a mapping strategy. However, one can wonder if it would be also possible to automatically suggest a mapping strategy ? As will be shown in Section 5.5.1 the *Strategy Predictor* module is a valuable answer to this question
Adjust parameters: this feature enables a user to experiment with different values of parameters in order to find out their optimal values. Parameter values could also be automatically suggested. Even in this case the *Strategy Predictor* will be shown to be useful for this purpose.
Explore ontologies: this requirement is fundamental in order to allow a user to have an immediate view on the problem s/he is facing. Ontologies should be navigated according to different visualization layouts allowing, for instance, focusing on the context of a concept. It would be also valuable to show detailed information about classes, properties and instances.
Suggest initial candidate mappings: this feature allows a user to establish well-founded mappings (i.e., ground truth) that can be exploited to infer new mappings or also as a reference basis for comparing different mapping strategies.
Phase 2: **Running**
Execute a mapping task: this function allows a user to start, stop, save and resume a mapping task execution.
Phase 3: **Evaluation**
Explore candidate mappings: this feature allows a user to navigate the results of a mapping task execution.
Evaluate quality of results: through this feature, a user can evaluate the quality of the results of a mapping task execution in terms of standard metrics (e.g., Precision, Recall and F-Measure).
Evaluate performance of a mapping strategy: this feature allows a user to investigate the cost (in terms of time) of a mapping strategy. In certain contexts a tradeoff between quality of results and performance is required.
Compare mapping strategies: this feature allows a user to compare different mapping strategies in terms of quality of results and performance.
Construct mappings: by this function, users can create the mappings according to the format defined in [49]. Mappings can be stored and made available to other applications.

5.3.1 Visualization

In this category the *OntoLoader* module was designed which has a twofold purpose. On the one hand, by exploiting the Protégé API [1], it allows constructing the set which contains *source* (i.e., S) or *target* (i.e., T) ontology entities. On the other hand, it is responsible for providing a representation of an ontology as a graph, in which classes consist of nodes and edges consist of relations between classes. The ontology can be loaded (in a GUI) and a user can choose different types of visualizations and layouts. It is also possible to obtain information about classes, properties and instances. Moreover, the source ontology O_s and the target ontology O_t can also be

[1] `http://protege.stanford.edu/plugins/owl/api`

shown in the same GUI thus allowing the user to manually provide an initial set of mappings. This feature allows constructing a reference alignment on which different mapping strategies can be evaluated.

5.3.2 Matching

This category of modules contains the individual matchers. Note that most of these matchers already exist in the literature and have been integrated in the presented framework to make possible the evaluation of the *Strategy Predictor* module. Moreover, the set of matchers included here is not meant to be exhaustive, that is, both new additional strategies and categories can be considered. A matcher takes as input two sets of ontology entities (i.e., *S* and *T*) and returns a similarity matrix which assigns to each source and target entity a similarity score. Each matcher implements its own similarity function. Currently, this category includes four individual matchers:

1. *Lucene Ontology Matcher (LOM)*: this matcher has been extensively discussed in Chapter 4.
2. *String Matcher (SM)*: this matcher implements some of the most common string comparison strategies such as the I-Sub metric (SMi) described in [161]; (ii) the Winkler metric (SMj) described in [180] (iii) a similarity metric (SMe) based on the edit-distance [98].
3. *WordNet Matcher (WM)*: this matcher exploits the WordNet [112] lexical database by implementing the similarity metric defined in [84].
4. *Structural Matcher (SOM)*: this matcher implements the approach defined in [166].

The LOM, SM and WM matchers have been designed to exploit linguistic descriptions of ontology entities and are classified as *Lexical Matching Strategies* (LMS). SOM has been designed to exploit the structural information of ontology entities. Finally, in this work we do not address computational issues due, for example, to the size of ontologies and assume that in case of large ontologies an approach such that proposed in [178] is adopted.

5.3.3 Combination

In this category the *Combiner* module was designed which takes as input a set of similarity assessments (provided by the individual matchers) and a set of configuration parameters and returns results obtained by applying a combination function (e.g., a weighted sum). The *Selector* module is exploited to select results according to a specific strategy (e.g., filtering results on the basis of a threshold).

5.3.4 Strategy Prediction: the *Strategy Predictor* Module

As pointed out in Chapter 3 (see Section 3.3.4), a main disadvantage of current OM frameworks is the lack of support to automatically suggest a mapping strategy and tune parameter values. In this section, to cope with this drawback, the novel *Strategy Predictor* module is presented. This module has been extensively evaluated and results of this evaluation are discussed in Section 5.5. The *Strategy Predictor* takes as input two sets of ontology entities and suggests the individual matchers to be adopted and the values of parameters that allow to combine (by adopting a *Combiner* module) and select (by adopting a *Selector* module) results. The *Strategy Predictor* scrutinizes O_s and O_t and assesses the coefficients of lexical and structural affinity. The lexical affinity (L_a) is defined as follows:

$$L_a(O_s, O_t) = \frac{\#common\,entities}{min(|S|\,, |T|)}. \tag{5.1}$$

where *#common entities* is incremented each time the *linguistic description* (LD) of a source entity(e_s) is very similar (e.g., 0.8 in a scale from 0 to 1) to the LD of a target entity(e_t). The LD of an entity contains linguistic information in terms of name, labels, comments and so forth. The rationale of L_a is that the more two ontologies use the same linguistic terminology to name their entities the more they are linguistically comparable. A high L_a indicates that a lexical mapping strategy should confidently discover mappings and the *Strategy Predictor* will suggest to use it.

In order to assess structural affinity S_a the notion of *intrinsic* Information Content (IC) [155] exploited in the WordNet Matcher (WM) is extended. The IC of an entity e is defined as the quantity of information it provides. In particular, in this context IC values are extracted for entities by considering their position in the hierarchy they belong to. Thus, the Structural Information Content (ICs) for an entity e is obtained by adapting the formulation provided in [155].

$$Ic_s(e) = 1 - \frac{log(Sub(e) + 1)}{log(|E|)}. \tag{5.2}$$

where $Sub(e)$ indicates the number of sub-entities of a given entity e in its hierarchical structure and $|E|$ is the total number of entities in the hierarchy. Note that IC_s values are monotonically decreasing as we go up to in a hierarchy. Overall, S_a is defined as:

$$S_a(O_s, O_t) = \frac{\#common\,entities}{min(|S|\,, |T|)}. \tag{5.3}$$

where *#common_entities* is incremented each time two entities have a very similar (e.g., 0.8 in a scale from 0 to 1) ICs and the same depth w.r.t the root of the hierarchical structure they belong to. L_a and S_a are exploited by the *Strategy Predictor* module to suggest the following parameters:

- *Lexical weight* (w_l): weight assigned to the mappings discovered by LMS.
- *Structural weight* (w_s): weight assigned to the mappings discovered by the SOM matcher.
- *Threshold* (Th): the cutoff threshold values for LMS (i.e., Th_l) and SOM (i.e., Th_s).

Moreover, the *Strategy Predictor* can also suggest the category of matchers to be included in a mapping strategy. In particular, if the value of an affinity coefficient (e.g., S_a) does not exceed a given threshold, the corresponding mapping technique (in this case SOM) is not adopted.

On determining the optimal threshold values

The basic idea in suggesting the threshold values is that the higher are L_a and S_a the more results can be safely assessed. Therefore, as the affinity coefficients increase the thresholds values have to decrease. The exponential decreasing function is appropriate to reach this goal. The *Strategy Predictor* determines the optimal thresholds values as:

$$Th_l = e^{-\alpha L_a}. \tag{5.4}$$

$$Th_s = e^{-\beta L_s}. \tag{5.5}$$

where α <1 and β <1 are constants quantifying the monotonic decreasing rate.

On determining optimal weight values

The *Strategy Predictor* in order to suggest the optimal weights values, starts from a common-sense consideration: if an affinity coefficient is high then the mappings discovered by the corresponding category of matchers have to be weighted high. Therefore, the assignment of weights has to be governed by an increasing function. To model this behavior the following two functions were adopted:

$$w_l = \frac{e^{\eta L_a} - e^{-\eta L_a}}{e^{\eta L_a} + e^{-\eta L_a}}. \tag{5.6}$$

$$w_s = \frac{e^{\gamma L_s} - e^{-\gamma L_s}}{e^{\gamma L_s} + e^{-\gamma L_s}}. \tag{5.7}$$

where η and γ >0 are smoothing factors. The above formulas are adaptations of Shepard's law [158] used in psychological science.

5.3.5 Evaluation

This category includes the modules exploited to perform different kinds of evaluation on a mapping strategy. In particular three modules were designed. The *Evaluator* module allows evaluating the results of a mapping strategy in terms of Precision (P), Recall (R) and F-measure (F-m). These metrics [41] are based on the comparison of an expected result with that returned by the system. In the context of OM, a set of mappings obtained by a mapping task w.r.t a reference alignment are compared. Note that generally a reference alignment is not available. However, a basic reference alignment can easily be constructed by a user which can graphically scrutinize the ontologies to be mapped through the *OntoLoader* module. The *Comparer* module allows the comparison of two mapping strategies in terms of P, R and F-m. The *Performance Evaluator* module allows performance to be evaluated (in terms of time elapsed) of the different modules or of a mapping strategy for the whole.

5.4 UFOme: a Comprehensive Ontology Mapping System

The requirements of the designed OM framework were defined in Section 5.2.1 and Section 5.3 described how these requirements can be fulfilled. This section presents an overall view of the framework architecture and its implementation through the User Friendly Ontology mapping environment (UFOme) system. Moreover, through an example, the support facility that UFOme provides to users, in terms of GUIs, is also shown.

5.4.1 A Reference Ontology Mapping Framework Architecture

Fig. 5.3 depicts an overall view of the designed mapping framework. This architecture brings together discussions done in the previous sections. According to the

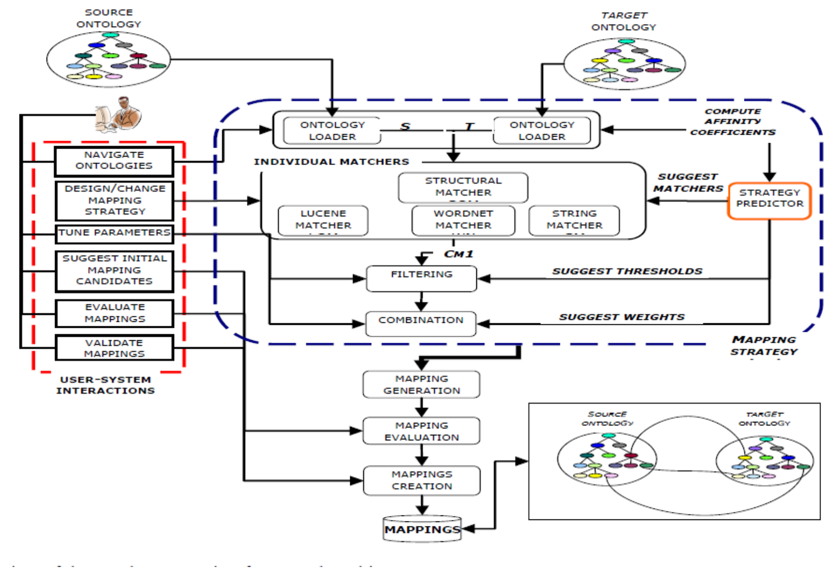


Fig. 5.3. An overall view of the ontology mapping framework architecture

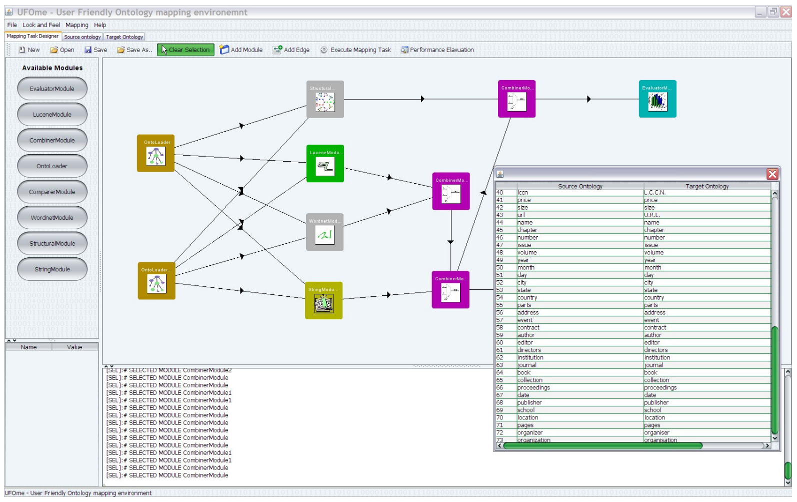
processing flow depicted in this figure, given the source and target ontologies O_s and O_t, the two sets S and T containing entities of O_s and O_t are constructed by exploiting two *Ontology Loader* modules. The individual matchers produce a set of candidate mappings Cm_1 that is passed to other modules responsible to refine results by considering the set of parameters P. The individual matchers to be included in the mapping strategy and the parameter values can be suggested by the *Strategy Predictor* module. A user can interact with the system to provide an initial set of candidate mappings C_m, chose or change the mapping strategy, navigate ontologies, validate and then produce mappings.

The UFOme System

The UFOme system has been implemented in Java. In the remainder of this section how UFOme supports users in the different phases of a mapping task execution will be shown.

Phase 1: Designing

In this phase, a user can choose the various modules to be included in the mapping task. Fig. 5.4 shows a snapshot of the UFOme main GUI. On the left hand side of

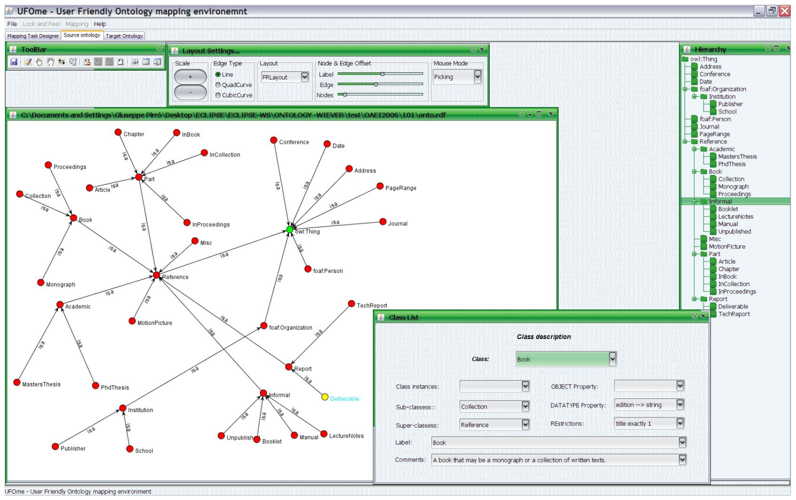


	Source Ontology	Target Ontology
40	lccn	L.C.C.N.
41	price	price
42	size	size
43	url	U.R.L.
44	name	name
45	chapter	chapter
46	number	number
47	issue	issue
48	volume	volume
49	year	year
50	month	month
51	day	day
52	city	city
53	state	state
54	country	country
55	parts	parts
56	address	address
57	event	event
58	contract	contract
59	author	author
60	editor	editor
61	directors	directors
62	institution	institution
63	journal	journal
64	book	book
65	collection	collection
66	proceedings	proceedings
67	date	date
68	publisher	publisher
69	school	school
70	location	location
71	pages	pages
72	organizer	organiser
73	organization	organisation

```
[SEL]:# SELECTED MODULE CombinerModule2
[SEL]:# SELECTED MODULE CombinerModule
[SEL]:# SELECTED MODULE CombinerModule
[SEL]:# SELECTED MODULE CombinerModule1
[SEL]:# SELECTED MODULE CombinerModule1
[SEL]:# SELECTED MODULE CombinerModule
[SEL]:# SELECTED MODULE CombinerModule
[SEL]:# SELECTED MODULE CombinerModule
[SEL]:# SELECTED MODULE CombinerModule
[SEL]:# SELECTED MODULE CombinerModule
[SEL]:# SELECTED MODULE CombinerModule
[SEL]:# SELECTED MODULE CombinerModule1
[SEL]:# SELECTED MODULE CombinerModule1
[SEL]:# SELECTED MODULE CombinerModule
[SEL]:# SELECTED MODULE CombinerModule
```

Fig. 5.4. A snapshot of the UFOme main GUI

the GUI there are the available modules. The user can place them into the mapping task composer (shown in Fig. 5.4) and construct the DAG of the mapping strategy. Parameters of each module are assigned by exploiting the table shown in the left hand side of Fig. 5.4. In the mapping strategy depicted in Fig. 5.4, the results produced by the two *OntoLoader* modules are passed to four individual matchers. Note that the direction of the connections will be exploited to run the topological sort algorithm. The user can choose to visualize the ontologies to be mapped. Fig. 5.5 shows a graph representation of the ontology and the ontology taxonomy. Fig. 5.5 also shows a toolbar allowing to change the visualization layout. Moreover, other information such as: instances, properties, and so forth are also shown. It is also possible to show the two ontologies to be mapped in the same interface thus allowing the user to discover and suggest initial candidate mappings and/or construct a reference alignment on which to evaluate different mapping strategies. Moreover, concepts can be seen in their context thus allowing the user, when validating mappings, to be more confident. For instance, if two concepts have been suggested as similar by the mapping strategy, but their contexts are different the user can choose not to validate this mapping.



UFOme - User Friendly Ontology mapping environemnt

FRLayout

Picking

C:\Documents and Settings\Giuseppe Pirrò\Desktop\ECLIPSE\ECLIPSE-WS\ONTOLOGY-WIEVER\test\OAEI2006\101\onto.rdf

Class description

Class: Book

Class instances:

OBJECT Property:

Sub-classess:: Collection

DATATYPE Property: edition --> string

Super-classess: Reference

REstrictions: title exactly 1

Label: Book

Comments: A book that may be a monograph or a collection of written texts.

UFOme - User Friendly Ontology mapping environment

Fig. 5.5. The UFOme ontology navigation perspective

Phase 2: Running

In this phase, the mapping task is executed according to the order defined by the topological sort algorithm. Results produced by each module are both stored in the module, for allowing individual analysis of the results, and passed to the modules to which it is connected. It is also possible to store a mapping task, stop and restart its execution.

Phase 3: Evaluation

In this phase, the user can explore results of the task and possibly rerun it by choosing a different mapping strategy (i.e., a different combination of modules and/or parameters). In current OM systems, designing different techniques means coding ad-hoc programs conversely in UFOme it corresponds to graphically (re)connect a set of modules. That valuable support makes the system highly usable even by non-expert users. In Fig. 5.4, the *Evaluator* module takes as input the result of the combination (obtained by the *Combiner* module) of the mappings discovered by both LOM and WM. A mapping task can be evaluated in terms of P, R and F-m. In Fig. 5.4 the combination of different strategies have been evaluated. The *Performance Evaluator* module shown in Fig. 5.4 allows evaluating performance of a strategy.

Finally, note that the user-friendship of UFOme consists in two main features: (i) it provides a user-friendly environment in terms of graphical interface; (ii) it, through the *Strategy Predictor*, suggests the mapping strategy with related parameter values thus decreasing the burden to a user.

5.5 Evaluation

This section focuses on evaluating functional aspects of UFOme and in particular discusses in detail the results of the evaluation of the *Strategy Predictor* module which aims at automatically determining the most suitable mapping strategy in terms of modules and parameter values. All experiments were performed on an Intel Core 2 running at 2.0 GHz with 2 GB of memory. UFOme was evaluated on the OAEI benchmark test suite [30] (already discussed in Section 4.3) in terms of Precision (P), Recall (R) and F-Measure (F-m). In Table 5.2 the dataset used is briefly summarized.

Table 5.2. The OAEI benchmark dataset

Group	Ontologies
G1	Ontologies from 101-104
G2	Ontologies from 201-210
G3	Ontologies from 221-247
G4	Ontologies from 248-266
G5	Ontologies from 301-304

5.5.1 Evaluation of the *Strategy Predictor* Module

In UFOme, as well as in all multi-strategy based mapping approaches, weights and thresholds have to be correctly set in order to avoid worsening the quality of results. As discussed in the previous sections, in UFOme results depend on different factors: (i) thresholds: used to cut off results; (ii) weights: used to define the contribution of each individual matching strategy to the final similarity score; (iii) matching module selection: exploited to select the individual matchers to be included in a mapping strategy. The *Strategy Predictor* was evaluated on these three aspects.

5.5.2 Effect of the Threshold on Linguistic Mapping Strategies

In order to see how a manually set threshold value can affect results of LMS, ontologies in G2 were considered that are linguistic alterations of the reference ontology (i.e., ontology 101). The average P, R and F-m were computed as a function of the linguistic threshold Th_l. The I-Sub metric was adopted and its results were combined with these provided by LOM and WM. In these experiments, WM outperformed the others in ontology 204 where the names of entities are replaced with synonyms. LOM performed better than the other techniques when the names of the entities are altered (e.g., ontology 206) since it can assess the correct similarity by exploiting other source of linguistic information such as comments and labels. The combination of these three strategies produced better results than the use of a single strategy as discussed in Chapter 4. In this case, using more than one linguistic strategy improved results. Fig. 5.6 shows the P, R and F-m as a function of Th_l. The best value of P is reached when the Th_l values is 1.0 while R and F-m are better when Th_l is set at 0.12. Note that the Th_l value, if not correctly set, can worsen performance by

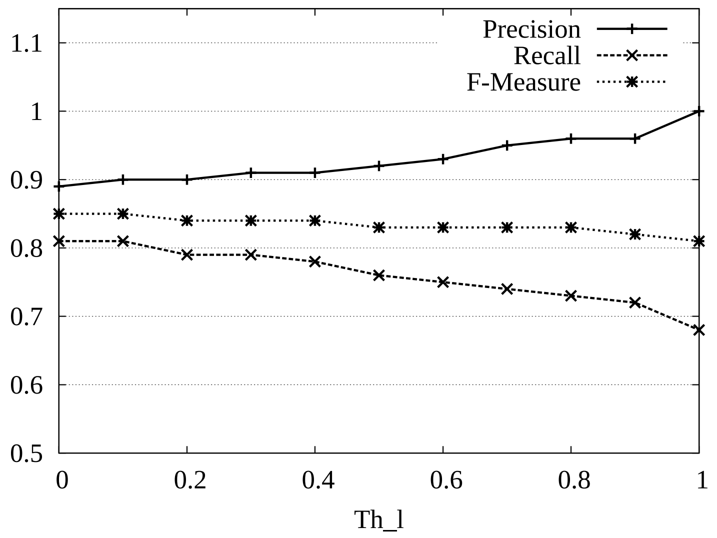


Fig. 5.6. Average values of P, R and F-m as a function of Th_l

about 15% in terms of P, 20% in terms of R and 6% in terms of F-m. In order to evaluate the *Strategy Predictor*, the values of P, R, and F-m were compared when Th_l is manually assigned, with those in which Th_l is suggested by the *Strategy Predictor* on ontologies of G2. Now, a question arises: how to assign the proper value to Th_l? Of course, on trying all the possible values between 0 and 1 the optimal threshold value would be obtained. Indeed, this is a very time expensive process and can only be performed if there is the ground truth mapping on which to evaluate the outcome obtained. Therefore, a Th_l value of 0.51 is assigned, which can be considered a fair value between 0 and 1. Table 5.3 reports the results of this evaluation. As can be noticed the results of P, R and F-m are clearly better when the *Strategy Predictor* is adopted. In particular, on the overall set of tests the P value slightly increases while R and F-m values increase by 25% and 23% respectively.

5.5.3 Effect of the Threshold on Structural Ontology Mapping

The same evaluation was performed for the SOM matcher on G3 ontologies. Fig. 5.7 shows the effects of the structural threshold Th_s on average P, R and F-m. In this case, the optimal Th_s value is about 0.1 since the F-m reaches its maximum value. As can be noted, when Th_s increases the value of P improves while R and F-m dramatically decrease. Even in this case, results were compared when assigning

Table 5.3. Manual Th_l vs. automatic Th_l for *LMS*

Ontology	Manual Th_l			Automatic Th_l		
	Precision	**Recall**	**F-measure**	**Precision**	**Recall**	**F-measure**
201	1.00	0.57	0.72	1.00	0.91	0.95
202	1.00	0.01	0.02	1.00	0.14	0.26
203	1.00	1.00	1.00	1.00	1.00	1.00
204	0.99	0.92	0.95	0.99	1.00	0.99
205	0.77	0.67	0.72	0.83	0.96	0.89
206	1.00	0.61	0.76	1.00	0.92	0.96
207	1.00	0.61	0.76	1.00	0.92	0.96
208	0.99	0.98	0.98	0.99	0.98	0.98
209	0.70	0.46	0.56	0.70	0.46	0.56
210	1.00	0.45	0.62	1.00	0.45	0.62
Overall(201-210)	0.94	0.63	0.71	0.96	0.79	0.87

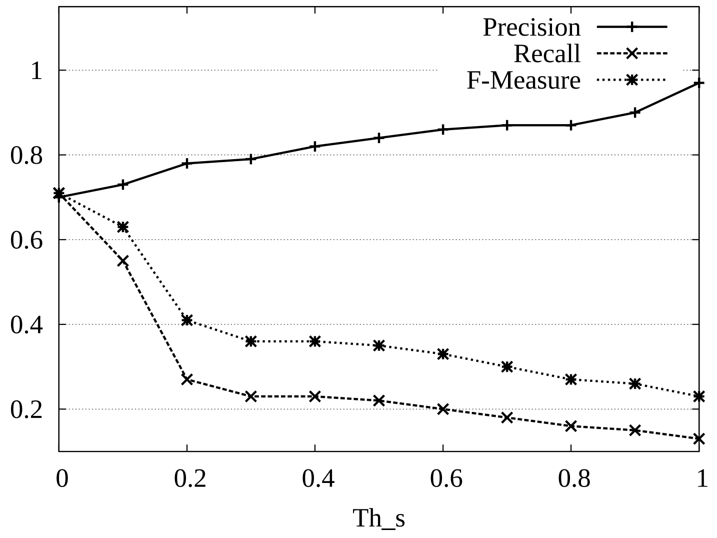


Fig. 5.7. Average values of P, R and F-m as a function of Th_s

a Th_s of 0.51 with those in which the Th_s is determined by the *Strategy Predictor* module. Ten ontologies belonging to the OAEI group including ontologies from 221 to 247 were considered. Table 5.4 shows the results of this evaluation. Even in this case, the values of Th_s suggested by the *Strategy Predictor* slightly improve results in terms of R and F-m while P remains the same. In some cases, i.e. ontology 247, no mappings are discovered with a manually determined Th_s.

Table 5.4. Manual Th_s vs. automatic Th_s for *SOM*

Ontology	**Manual Th_s**			**Automatic Th_s**		
	Precision	**Recall**	**F-measure**	**Precision**	**Recall**	**F-measure**
221	0.89	0.86	0.88	0.89	0.86	0.88
222	0.69	0.69	0.69	0.69	0.69	0.69
223	0.73	0.73	0.73	0.74	0.74	0.74
224	1.00	1.00	1.00	1.00	1.00	1.00
225	0.92	0.92	0.92	0.92	0.92	0.92
228	0.33	0.30	0.32	0.33	0.33	0.33
232	0.86	0.84	0.85	0.86	0.84	0.85
237	0.61	0.61	0.61	0.62	0.62	0.62
246	0.40	0.34	0.37	0.41	0.35	0.39
247	0.00	0.00	0.00	0.27	0.27	0.27
Overall(221-247)	0.67	0.65	0.66	0.67	0.67	0.67

5.5.4 Evaluating Automatic Weights Assignment

The effects of manual weights assignment were evaluated on ontologies from 248 to 266. The evaluation reported in Table 5.5 shows that the F-m reaches the best value when more emphasis is given to w_l whereas if the same is done with w_s, F-m dramatically decreases. In this experiment, both LMS and SOM were adopted. However, it is

Table 5.5. Manual weights assignment

Ontology Group G4		
w_l	w_s	**F-m**
0.1	0.9	0.12
0.2	0.8	0.26
0.3	0.7	0.37
0.4	0.6	0.48
0.5	0.5	0.54
0.6	0.4	0.59
0.7	0.3	0.61
0.8	0.2	0.67
0.9	0.1	0.73

worth noting that LMS are more effective than SOM as the F-m improves when giving w_l a higher value than w_s. Therefore, in this case, adopting a multiple-mapping strategy worsens the F-m value.

In order to evaluate the *Strategy Predictor*, 0.5 was manually assigned to both w_l and w_s and the results compared with those in which w_l and w_s are suggested by the *Strategy Predictor* for ontologies from 301 to 304. The Th_l and Th_s for manual assignment are set to 0.51. Table 5.6 shows the results of this evaluation. As can be noted, the automatic weight assignment worsens P by about 18% while R increases by about 13%. In this case, the cost of a higher R is paid in terms of P. The most notable improvement in terms of R is in ontology 304 where R increases by about 30%, however, P decreases by about 30%. Note that in the overall evaluation the value of R obtained with automatic weights assignment is 0.83 while the maximum reached with manually assigned weights in our evaluation is 0.73. These different values are due to the threshold values that were set to 0.51 in the manual evaluation, while they are determined by the *Strategy Predictor* in automatic weight assignment. This underlines how the *Strategy Predictor* is effective.

Table 5.6. Manual vs. automatic weights assignment on real life ontologies

	Manual w_l and w_s		**Automatic** w_l and w_s	
Ontology	**Precision**	**Recall**	**Precision**	**Recall**
301	0.68	0.79	0.58	0.82
302	0.47	0.60	0.45	0.66
303	0.62	0.80	0.50	0.87
304	0.85	0.75	0.65	0.97
Overall(301-304)	0.65	0.73	0.55	0.83

Evaluating the Effects of Mapping Modules Selection

In this experiment, the effect of including *SOM* in a mapping strategy was evaluated. Table 5.7 shows the results on all the groups of tests. The UFOme column indicates that in the fully-fledged UFOme, if an affinity coefficient (e.g., S_a) is below a given threshold (experimentally set to 0.40) the corresponding matching modules (in this case SOM) will not be adopted. As one would expect, in ontologies where the structure is similar but lexical information has been altered (i.e., G2), the values of P and R improve when adopting SOM. However, in ontologies belonging to G5, which have similar linguistic features, results worsen if SOM is included. In the fully-fledged UFOme, results on G5 ontologies are only derived from LMS since the *Strategy Predictor* suggests not adopting the SOM matcher. In the overall evaluation, the fully-fledged UFOme, obtains a better P (increases by about 4%) while R improves by about 40%.

Comparing UFOme with Other Systems

In this experiment, UFOme was compared with other ontology mapping algorithms. For those systems, results shown in Table 5.8 are those reported by the organizers of OAEI 2006 [54]. In particular, the evaluation was focused on OAEI real life ontologies (i.e., G5). As can be noted, UFOme is among the top four mapping systems. In particular even if it is less precise than other algorithms, it reaches the best value in terms of R and F-m. It is worth noting that in this evaluation, the parameter values

Table 5.7. Effects of strategy selection on all tests

Test Group	**LMS**		**SOM**		**UFOme**	
	Precision	**Recall**	**Precision**	**Recall**	**Precision**	**Recall**
G1(101-104)	1.00	1.00	1.00	1.00	1.00	1.00
G2(201-210)	0.96	0.78	0.98	0.86	0.98	0.94
G3(221-247)	0.99	1.00	0.56	0.50	0.99	1.00
G4(248-266)	0.61	0.16	0.35	0.31	0.68	0.47
G5(301-304)	0.84	0.83	0.20	0.15	0.84	0.83
Overall(101-304)	0.78	0.63	0.56	0.51	0.81	0.89

and the mapping strategy have been automatically determined by the *Strategy Predictor*. It can be concluded that the *Strategy Predictor* is a valuable tool to suggest a mapping strategy and parameter values. In particular, the L_a and S_a coefficients correctly grasp the information needed to design a mapping strategy and set parameter values.

Table 5.8. Comparison of UFOme with other systems

Mapping System	**Ontologies in G5**		
	Precision	**Recall**	**F-Measure**
RiMOM	0.83	0.82	0.83
Falcon	0.89	0.78	0.83
H-Match	0.78	0.57	0.66
Automs	0.91	0.70	0.79
Coma	0.84	0.69	0.76
Jhuapl	0.18	0.50	0.26
Edna	0.94	0.61	0.74
DSSim	0.90	0.78	0.83
Prior	0.85	0.80	0.82
OCM	0.89	0.51	0.65
UFOme	0.84	0.83	0.83

5.6 Discussion and Lesson Learned

This chapter discussed an ontology mapping framework tackling ontology mapping in a systematic way both from functional and user viewpoint. The framework requirements are fulfilled by building block components dubbed as modules. It has been shown in the evaluation section that wrong values assigned to parameters can significantly worsen results. Thus, the new *Strategy Predictor* module was designed, which obtains significant results while avoiding manual parameter tuning. Based on this framework, an implementation of the UFOme framework was performed. On the one hand, UFOme automates the design of a mapping strategy by scrutinizing the affinities between the ontologies to be mapped; on the other hand, it provides a comprehensive and user friendly ontology mapping environment

6

SECCO: on Building Semantic Links in P2P Networks

In a recent interview [171], Tim Berners-Lee, the inventor of the Semantic Web, stated that:

> ”The Semantic Web is designed to smoothly interconnect personal information management, enterprise application integration, and the global sharing of commercial, scientific and cultural data...”

From this interview emerges that semantic-based data sharing is expected to begin in controlled environments smaller than the World Wide Web as for instance: enterprise networks and small-medium Peer-to-Peer (P2P) networks. Moreover, the Semantic Web is expected to follow the same path of the Internet, which started in bounded environments. As discussed in details in Chapter 3, in distributed environments it is not feasible to have a single (and universally accepted) ontology describing a knowledge domain, but there will be several (possibly overlapping) ontologies created w.r.t ”the point of view” of their designers. This problem becomes more challenging in P2P networks for several reasons: (i) the number of overlapping ontologies can dramatically increase, in theory each peer will have its own ontology that reflects peer’s needs and interests; (ii) mappings among peer ontologies must be quickly discovered and only on the parts of ontologies ”contextual” to a specific interaction in which peers are involved.

Thus, ontology mapping algorithms for P2P networks should ensure a trade-off between fastness (not achievable by adopting complex mapping strategies) and accuracy (i.e., quality of results). As discussed in Chapter 3, several approaches to solve the OMP have been proposed. However, these approaches underestimate the following aspects:

- They do not adequately consider the Ontology Mapping Problem (OMP) in open environments such as P2P networks. There is a lack of mapping systems devoted to solve the OMP in open environments.
- They do not take into account the need for ”on the fly” mappings crucial in P2P networks. In such networks, a complete mapping between peer ontologies is not a requirement for interactions among peers; they only need to quickly map the parts

of their ontologies related to the specific interaction in which they are involved. Moreover, since peers are often unaware of one another's ontologies the amount of ontological information exploitable to discover mappings is quite limited.
- They do not adequately interpret the semantic meaning of ontology concepts to be compared. In addition, the context in which concepts appear is not carefully scrutinized from a semantic point of view.

In this chapter we address *online* ontology mapping by defining a new ontology mapping algorithm called SEmantiC COordinator (SECCO). We claim that the OMP in P2P environments is an important issue since P2P applications seem to be a class of applications that will take advantage of Semantic Web technologies in a near future. SECCO is composed by three individual matchers: syntactic, lexical and contextual, each of which tackles the OMP from a different perspective. In particular, the syntactic matcher exploits the LOM matcher presented in Chapter 4, the lexical matcher assesses semantic relatedness, even among syntactically unrelated concepts (e.g., car and automobile), by combining two approaches exploiting WordNet [112] as background knowledge. The contextual matcher implements a new similarity strategy called ”how it fits”. This strategy complies with the contextual theory of meaning [113] and is founded on the idea that two concepts are related if they fit well in each other's context. This approach allows comparing the structures of two concepts both in terms of their position in the ontological taxonomy and constituent properties. This is achieved at an affordable computational cost since it is not required to take into account the whole structure of ontologies. Specifically, the main contributions of this work are:

- We exploit the idea of concept mapping with the aim to gather similarity information between concepts belonging to different peer ontologies. Concept mappings allow building semantic links among peers that can be exploited in several classes of semantic P2P applications (e.g., semantic search, semantic query routing, and community formation).
- We designed and extensively evaluated SECCO with particular emphasis on the contextual matching strategy.

Our approach, differently from other semantic P2P applications (e.g., Piazza, GridVine) that assume the preexistence of mappings for achieving semantic interoperability, focuses on the problem of finding mappings. Extensive experimental results, aimed at comparing SECCO w.r.t the state of the art, show that the combination of the proposed mapping strategies provides an adequate trade-off between accuracy (in terms of quality of mappings) and fastness (in terms of elapsed time for discovering mappings). Moreover, we want to emphasize that SECCO, if coupled with a P2P platform, paves the way toward a comprehensive semantic P2P solution for content sharing and retrieval, semantic query answering and semantic routing. We report on the advantages of integrating SECCO within the K-link+ system that will be presented in Chapter 10.

The remainder of this chapter is organized as follows. Section 6.1, after introducing the terminology adopted in the rest of this chapter, presents the SECCO ontology mapping algorithm. Section 6.2 describes the individual matchers of SECCO with particular emphasis on the contextual matcher. In this section, we also motivate the design of SECCO. Section 6.3 presents a detailed evaluation of the system in two different settings. In particular, here we compare SECCO with H-Match [22] and performs a sensitivity analysis of the different parameters of the algorithm. We also evaluated the algorithm on four real-life ontologies of the OAEI 2006 benchmark test suite by comparing it with other mapping algorithms not explicitly designed for facing the OMP in P2P networks.

6.1 The SEmantiC COordinator (SECCO)

We designed the SECCO ontology mapping algorithm for addressing the OMP in P2P networks, since semantic P2P applications, built by interconnecting knowledge managed at personal level, seem to be the applications taking advantage of Semantic Web technologies in a near future [171]. We argue that most of the existing mapping algorithms are not suitable for P2P networks since they, to work properly, need to deal with the whole two ontologies to be mapped. For instance, top ontology mapping algorithms, i.e., Falcon [83], and RiMOM [183] have structural mapping strategies built upon graph matching techniques. These techniques are well suited to work *offline* while in P2P environments where the OMP has to be faced *online* specific techniques are required. In this section, after introducing the terminology adopted in the rest of the paper we describe the ontology model exploited by SECCO to discover mappings. Moreover, we also present a scenario of usage of SECCO and provides the pseudo-code of the algorithm.

6.1.1 Preliminary Definitions

We consider a P2P network in which each peer owns an ontology (i.e., peer ontology) that represents the point of view of the peer on a particular knowledge domain. Each (seeker) peer can request to other (providers) peers a concept mapping whose aim is to provide information of similarity among a concept belonging to the seeker peer ontology and concepts belonging to ontologies of provider peers. The aim of the request depends on the application class (e.g., semantic search). In this scenario, we define both seeker and provider peers as semantic peers since they manage, share and exchange knowledge by exploiting ontologies. In this chapter, we adopt the ontology model discussed in Chapter 3. However, in order to provide the reader with more details, we discuss how this model is built in Section 6.1.2.

Definition 6.1 ***(Seeker peer)***. *A seeker peer is a semantic peer that sends a semantic request over the P2P network to provider peers and receives a set of concept mappings.*

Definition 6.2 ***(Provider peer)***. *A provider peer is a semantic peer that receives a request from a seeker peer and returns a concept mapping obtained by exploiting SECCO.*

Definition 6.3 ***(Request)***. *Let O be an ontology, a request is a two-tuple of the form $RQ = \langle c, ctx(c)\rangle$ where $c \in C$ is a concept belonging to the seeker peer ontology and ctx(c) is the context of c.*

Definition 6.4 ***(Concept Context)***. *Let O be an ontology, the set of strings ctx(c) is the context of the concept $c \in C$. This set contains names of concepts related to c by relations in R that correspond to OWL objectype properties and the names of relations in R that correspond to OWL datatype properties. More formally, $ctx(c) = C_{range} \bigcup C_{dom} \bigcup R_{dt}$ where: (i) C_{range} and C_{dom} are the sets of concepts names for which respectively hold the following conditions: $c \in dom(r) \wedge c_{range} \in range(r)$ and $c_{dom} \in dom(r) \wedge c \in range(r)$ with $r \in R$ and range(r) and dom(r) both corresponding to user defined classes; (ii) R_{dt} is the set of relation names for which either range(r) or dom(r) is defined on a data type. Datatype property names, present in the original OWL ontology, are included in ctx(c) as described in Section 6.1.2.*

The concept mapping between a seeker peer concept and the set of concepts belonging to a provider peer ontology is defined as follows:

Definition 6.5 ***(Concept Mapping)***. *Given the seeker peer request $RQ = \langle s, ctx(s)\rangle$ and the ontology O belonging to a provider peer, a concept mapping M between each provider concept $p \in C$ and the seeker peer concept s is a set of 3-tuples of the form $M = \langle s, p, \sigma\rangle$ where $\sigma \in [0, 1]$ is the similarity value between the couple of concepts s and $p \in C_p$.*

Similarity values between couples of concepts are obtained by adopting the similarity measure defined as follows:

Definition 6.6 ***(Similarity Measure)***. *Given two ontologies O_s and O_p belonging to a seeker and a provider peer respectively, a request $RQ = \langle s, ctx(s)\rangle$ and the set CTX_p composed by two-tuples of the form $\langle c_j, ctx(c_j)\rangle$ where $\forall c_j \in C_p$ $ctx(c_j)$ is the context of c_j; the similarity between the couples of concepts $c_s \in C_s$ and $c_p \in C_p$ is computed by the following function:*

$$sim(c_s, c_p) : f(sim_{syn}(c_s, c_p), sim_{lex}(c_s, c_p), sim_{ctx}(c_s, c_p)) \mapsto [0, 1] \tag{6.1}$$

where $sim_{syn}(c_s, c_p)$ is the syntactic similarity, $sim_{lex}(c_s, c_p)$ is the lexical similarity and $sim_{ctx}(c_s, c_p)$ is the contextual similarity. These three similarity measures are symmetric and reflexive i.e., $\forall c_s \in C_s$ and $c_p \in C_p$:

$$sim(c_s, c_s) = 1 \qquad (reflexivity)$$

$$sim(c_s, c_p) = sim(c_p, c_s) \qquad (symmetry)$$

How to Represent Mappings in SECCO

Even if the OMP has received a lot of attention from the scientific community, a standardized format for storing ontology mappings does not exist. In order to overcome this problem, there are two possible ways:

1. Exploiting features of ontology languages. For instance, OWL provides built-in constructs for representing equivalence between concepts (i.e.,*owl:equivalentClass*), relations (i.e., *owl:equivalentProperty*) and instances (i.e., *owl:sameAs*). This approach allows OWL inference engines to automatically interpret the semantics of mappings and perform reasoning across different ontologies. However, by adopting this approach, a confidence value cannot be interpreted.
2. Adopting the approach described in [49]. This mapping representation exploits RDF/XML to formalize ontology mappings. Each individual mapping is represented in cells and each cell has the following attributes: *entity 1* (i.e., the concept in the source ontology), *entity 2* (i.e., the concept in the target ontology), measure (i.e., the confidence value), *type of mapping* (usually equivalence). Due to its different parameters, this representation can easily be exploited by several kinds of applications.

In SECCO, we adapt the second type of representation to the context of a P2P ontology mapping system. The adopted mapping representation is depicted in Fig. 6.1. This representation allows a seeker peer (i.e., the *seeker_peer* tag), for a given seeker concept (i.e., the *seeker_concept* tag), to maintain both the URIs of provider concepts (i.e., *provider_concept* tag) and values of similarity (i.e., the similarity tag) grouped on the basis of provider peers (i.e., the *provider_peer* tag) that answered to the seeker request.

```
<mapping>
   <seeker_peer name= ID>
   <seeker_concept=URI>
   <provider_peer name = ID>
            <provider_concept ID=URI>
                 <similarity>σ</similarity>
            </provider_concept>
            <provider_concept ID=URI>
                 <similarity> σ</similarity>
            </provider_concept>
             …
   </provider_peer name>
     ...
 </mapping>
```

Fig. 6.1. Representation of mappings in SECCO

6.1.2 The SECCO Ontology Model Construction

The SECCO ontology model is built by exploring ontology class definitions contained in peer ontologies. To explain how the SECCO ontology model is constructed, let us consider the fragment of the Ka ontology [1] shown in the following.

[1] `http://www.cs.man.ac.uk/~horrocks/OWL/Ontologies/ka.owl`

```
<owl:Class rdf:about="file:F:/..ka.daml#Publication">
 <rdfs:subClassOf>
    <owl:Restriction>
        <owl:onProperty rdf:resource="file:F:/..ka.daml#title"/>
        <owl:allValuesFrom rdf:resource="http:..XMLSchema#string"/>
    </owl:Restriction>
 </rdfs:subClassOf>
 <rdfs:subClassOf>
   <owl:Restriction>
   <owl:onProperty rdf:resource="file:F:/..ka.daml#describesProject"/>
       <owl:allValuesFrom>
           <owl:Class rdf:about="file:F:/..ka.daml#Project"/>
       </owl:allValuesFrom>
   </owl:Restriction>
 </rdfs:subClassOf>
....
 <rdfs:subClassOf>
   <owl:Restriction>
       <owl:onProperty rdf:resource="file:F:/..ka.daml#year"/>
       <owl:allValuesFrom rdf:resource="http:../XMLSchema#integer"/>
   </owl:Restriction>
 </rdfs:subClassOf>
....
<owl:Class rdf:about="file:F:/...ka.daml#Book">
      <rdfs:subClassOf>
          <owl:Class rdf:about="file:F:/...ka.daml#Publication"/>
      </rdfs:subClassOf>   ...
</owl:Class>
```

Given an input ontology, SECCO executes the following four main steps to construct its ontology model:

1. *Class name extraction*: for each class a concept name is created in the SECCO ontology model.
2. *Subclass properties (i.e., ISA) analysis*: for each class definition, SECCO scrutinizes its sub classes defined by the construct *rdfs:subClassOf* and generates the taxonomic structure.
3. *Datatype properties analysis*: values of these properties are data literals. For each datatype property SECCO considers the linguistic information encoded in the property name (e.g., *year* in *Ka*) and includes in its ontology model a new concept name representing the property (i.e., *year*) and a relation (i.e., *has_year* as shown in Fig. 6.2) to relate this new concept with the original class.
4. *Object properties analysis*: these are properties for which the value is an individual. In this case, for each property, SECCO exploits the original OWL encoding and generates a relation that has the same name, domain and range of the original one.

The considered ontology fragment contains the definition of a class *Publication* along with some object and datatype properties and related classes. By running SECCO, we obtain the ontology model representation depicted in Fig. 6.2. Note that the construction of this representation also exploits the definitions of classes (e.g., *Project*, *Event*) that are not represented in the excerpt above shown.

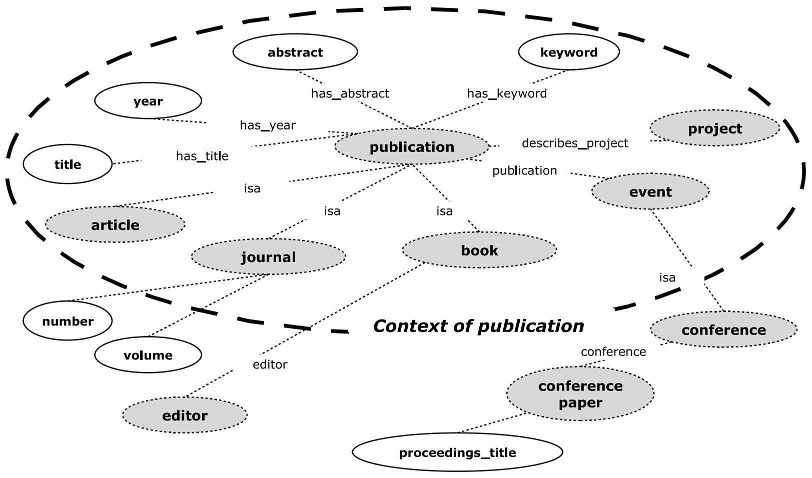


Fig. 6.2. The SECCO ontology model

In Fig. 6.2 filled oval represent ontology concepts (i.e., classes) as defined in the original OWL ontology whereas empty ovals are the concepts introduced in the SECCO ontology model to exploit information encoded in *Datatype* properties. For instance, the context of the *Publication* concept is the dashed area in Fig. 6.2. In more detail, *ctx(Publication)=title, year, abstract, keyword, Project, Event, Book, Journal, Article*.

6.1.3 The SECCO Algorithm

SECCO aims at discovering a concept mapping between a *seeker* peer concept and ontology concepts belonging to ontologies of *provider* peers. Each peer in the network plays a twofold role: (i) seeker peer, when it sends a request to the network; (ii) provider peer, when it executes locally the SECCO algorithm. Whenever a provider peer receives a request, it runs SECCO with an input of the following form:

$$I = \langle c_s, ctx(c_s), O_p, Th, w_s, w_l, w_c \rangle$$

where: c_s is a concept belonging to a seeker peer ontology; $ctx(c_s)$ is the context of c_s; O_p is the provider ontology, $Th \in [0, 1]$ is a threshold value that can be used for filtering results. Moreover, w_s, w_l, and w_c are the weights assigned to the values of syntactic, lexical and contextual similarity respectively. The overall similarity value is computed by the *Combiner* module that weights the similarity values provided by the individual matchers (see Fig. 6.3) and discards values that do not exceed a given threshold (i.e., the Th parameter in Fig. 6.3). Once SECCO has terminated, it returns a concept mapping. The overall approach is described in Fig. 6.3. A seeker peer issues an information request by picking a concept along with the related context from its ontology. This request reaches provider peers that run the SECCO algorithm on their ontologies and return to the seeker peer concept mappings that will be stored in the mapping store for future reuse. Fig. 6.4 describes the SECCO algorithm in

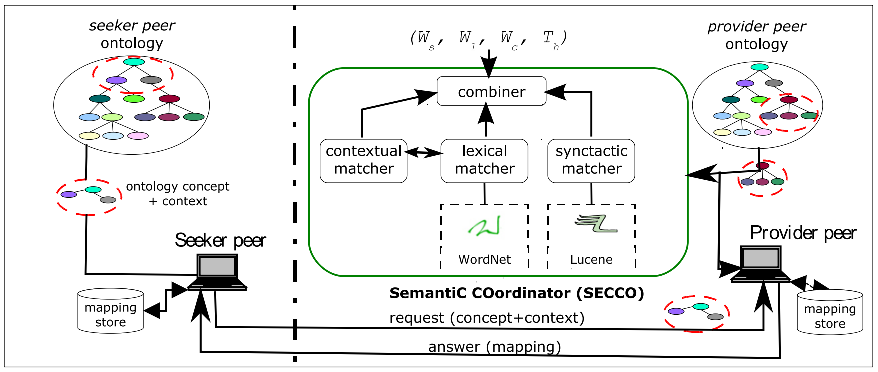


Fig. 6.3. The SECCO architecture and usage scenario

pseudo-code. The function *evaluate syntactic similarity* is implemented by the *syntactic matcher* while the function *evaluate lexical similarity*, is implemented by the *lexical matcher*. The function *evaluate contextual similarity* (see Fig. 6.5), implemented by the contextual matcher, relies on the function *evaluate how it fits* (see Fig. 6.6) that adopts a ”see how it fits” strategy that is founded on the idea that two concepts are related if they fit well in each other’s context. The contextual matcher takes as input the context obtained by the function *extract context* (see Fig. 6.7). In the next section, the individual matchers of SECCO are described with particular emphasis on the *contextual matcher*.

6.2 The Building Blocks of SECCO

The idea of combining different heuristics, each of which implemented by an individual matcher, for the assessment of an overall similarity value between two ontology entities is not new (see [45, 47]). The main motivation of adopting such a strategy is that from some ontology mapping initiatives (e.g., [54]) emerged that a combination of mapping strategies, in general, allows to obtain better results. Moreover, it is not arguable that a single heuristic is able to exploit all the types of information (e.g., lexical, structural) encoded in ontology entities. With these motivations in

```
                         The SECCO algorithm
Input: An input $I=<c_s, ctx(c_s), O, T_h, w_S, w_L, w_C>$ where O= C,R  is the
SECCO ontology model
Output: The concept mapping M
Method:
  1.  M= ;
  2.  for each  c C do
  3.    $sim_{syn}$=evaluate_syntactic_similarity($c_s$,c);
  4.    $sim_{lex}$=evaluate_lexical_similarity($c_s$,c);
  5.     ctx(c)=extract_context(c,O);
  6.     $sim_{con}$=evaluate_contextual_similarity($c_s$,Ctx($c_s$),c,Ctx(c));
  7.     sim=($w_S$*$sim_{syn}$+$w_L$*$sim_{lex}$+$w_C$*$sim_{con}$); /* overall similarity value */
  8.      if sim>$T_h$ then
  9.         m.s=$c_s$
  10.        m.p=c;
  11.        m.σ=sim
  12.        M=M m;
  13.     end-if
  14. end-for
  15. return M;
```

Fig. 6.4. The SECCO algorithm in pseudo-code

```
                    Function extract_context
Input: An ontology O= C,R  and a concept c C
Output: The context ctx(c)
Method:
    1.    $ctx_c$= ; $ctx_r$= ;
    2.    for each $c_c$ C do
    3.        for each $r_c$ R do
    4.            if  $r_c$(c,$c_c$)| $r_c$($c_c$,c) then
    5.               $ctx_c$=$ctx_c$ {$c_c$}
    6.               $ctx_r$=$ctx_r$ {$r_c$}
    7.            end-if
    8.        end-for
    9.    end-for
return ctx(c)= $ctx_c$,$ctx_r$ ;
```

Fig. 6.5. The *extract context* function

```
              Function evaluate_contextual_similarity
Input: Two concepts $c_1$ and $c_2$ and their contexts ctx($c_1$) and ctx($c_2$)
Output:  A  numerical  value  $sim_{con}$ [0,1]  representing  the  contextual
similarity between the concepts $c_1$ and $c_2$
Method:
    1.    s2s=evaluate_how_it_fits($c_1$, ctx($c_1$));
    2.    s2t=evaluate_how_it_fits($c_1$, ctx($c_2$));
    3.    t2s=evaluate_how_it_fits($c_2$, ctx($c_1$));
    4.    t2t=evaluate_how_it_fits($c_2$, ctx($c_2$));
    5.    $sim_{con}$=((1-||s2s-t2t|-|s2t+t2s||));
    6.    return $sim_{con}$
```

Fig. 6.6. The *evaluate contextual similarity* function

```
        Function evaluate_how_it_fits
Input: A concept c and a context ctx(x)= $C_x$,$R_x$
Output: A numerical value m [0,1] representing
the fitness between the concept c and the context ctx(x)
Method:
   T=0;
   for each $c_e$ $C_x$ do
          T+=evaluate_lexical_similarity(c,$c_e$);
   end-for
   return m=T/|ctx(x)|;
```

Fig. 6.7. The *evaluate how it fits* function

mind, we decided to endow SECCO with three different matchers. Each matcher respectively exploits syntactic/linguistic (syntactic matcher), lexical (lexical matcher) and contextual (contextual matcher) information contained in ontology entities. We adopt the syntactic matcher, since as shown in Chapter 4 we noticed that a merely syntactic approach can be effective and fast in discovering mappings. The lexical matcher is successful in discovering mappings in a semantic way, that is, by considering the semantic meaning of the compared terms and not treating them just as strings. Through this approach, it is possible, for instance, to discover that the *automobile* concept used in a *seeker* peer ontology is similar to the concept of *car* used in a *provider* peer ontology. Finally, the *contextual matcher* that relies on the lexical matcher allows refining similarity between concepts by considering the contexts in which they appear. The contextual matcher rationale complies with the contextual theory of meaning [113] according to which the relatedness between concepts can be defined in terms of their interchangeability within the contexts in which they appear. The contextual matcher allows the assessment of similarity between two concepts in terms of their structure/properties, but on a local basis, that is, by only considering the properties and neighbors of the two concepts and not the whole ontology structures in which they appear. Note that for the scenario in which SECCO has to work (i.e., a P2P network) a structural matching strategy could affect the requirement of fastness given that it requires to compare entire ontologies (e.g., [179]). Indeed, in SECCO, a provider peer only receives a request (i.e., a concept along with its context) from a seeker peer and not its entire ontology. Through experimental evaluations (see Section 6.3), we prove that the lack of a structural matcher will not significantly affect mapping results. Furthermore, it is worthwhile noting that the modular architecture of SECCO allows easily designing and adding new matchers to be included into the algorithm. In the next sections we provide a description of the three individual matchers with particular emphasis on the novel *contextual matcher*. Finally, we motivate the designing of SECCO by reporting on the advantages of integrating it in the K-link+ system that will be describe in Chapter 10.

Table 6.1. Semantic relations in the WordNet 3.0 noun taxonomy

Relation	**Description**	**Example**
Hypernymy	is a generalization of	Plant is an hypernym of Flower
Hyponymy	is a kind of	Tulip is *hyponym* to Flower
Meronymy	is a part of	Finger is a *meronym* of Hand
Holonymy	contains part	Tree is a *holonym* of Bark
Antonymy	opposite of	Man is an *antonym* of Woman
Instance of	is an instance of	California is an *instance of* American state
Has instance	has instance	American state *has instance* California

6.2.1 The Syntactic Matcher

This matcher that implements the function *evaluate syntactic similarity* (see Fig. 6.4, line 3), relies on the Lucene Ontology Matcher (LOM) described in Chapter 4. In particular, in order to adopt LOM in SECCO we made the following adaptations:

- Ontologies of both seeker and provider peers are indexed. Therefore, each peer exploits its index to search for similar entities for requests coming from seeker peers (i.e., acts as a provider).
- Since in SECCO we do not want to compare whole ontologies, but a seeker concept along with its context with provider ontology concepts, we construct a new type of virtual document that contains linguistic information of the concept along with linguistic information of its context. This way, the linguistic information of a concept is augmented with linguistic information of entities in its context. Therefore, also the syntactic matcher takes into account a certain degree of structural information.

The reader is remained to Chapter 4 for further details on the algorithm and a complete evaluation of its performance.

6.2.2 The Lexical Matcher

The lexical matcher, that implements the function *evaluate lexical similarity* (see Fig. 6.4, line 4), is the central component of the whole system. It allows implementing the semantic mapping by ”interpreting” the semantic meaning of concepts to be compared. The lexical matcher exploits WordNet [112] as a source of knowledge about the world. WordNet is a lexical ontology organized in *synsets* (or senses) that encompass terms with synonymous meaning. Each *synset* has a gloss, which is a description in natural language of the concepts it represents. Synsets are connected to one another by a predefined set of semantic relations, some of which are reported in Table 6.1. Some of these relations define inheritance relations (*Hypernymy* and *Hyponymy*), other *part-of* relations (*Holonymy* and *Meronymy*). The *Antonymy* relation is used to state that a noun is the opposite of another. The relations *instance of* and *has instance* have been introduced in WordNet 3.0 and represent instantiation relations. Fig. 6.8 shows an excerpt of the WordNet noun taxonomy. Through the lexical

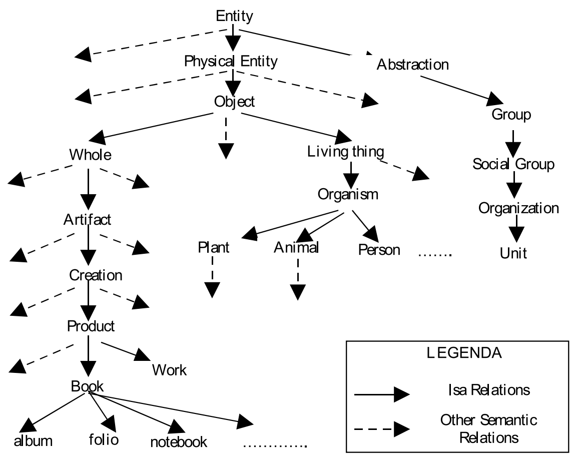


Fig. 6.8. An excerpt of the WordNet 3.0 noun taxonomy

matcher we aim at assessing the relatedness between ontology entities by exploiting their definitions within the WordNet database and position in the taxonomy. Semantic relatedness is the question of how related two concepts are by considering different kinds of relations connecting them. On the other hand, semantic similarity only considers the hypernymy/hyponymy relations among concepts. For instance, *Car* and *Gasoline* may be closely related to each other, e.g. because gasoline is the fuel most often used by cars. *Car* and *Bicycle* are semantically similar, not because they both have wheels and means of steering and propulsion, but because they are both kinds of *Vehicle*. The relation between semantically similar and semantically related is asymmetric: if two concepts are similar, they are also related, but they are not necessarily similar just because they are related. In the literature (see [131]), there are several metrics for assessing similarity and relatedness among concepts in WordNet. In the context of ontology mapping, several approaches (e.g., [53, 91]) compute semantic similarity between concepts by exploiting semantic similarity metrics. However, these approaches only consider the hypernymy/hyponymy relations linking synsets. In order to take into account a wide range of semantic relations connecting synsets we included two components in the *lexical matcher*. A *similarity assessor* aimed at assessing semantic similarity and a *relatedness assessor* aimed at assessing semantic relatedness. The final lexical similarity value is obtained by combining the contribution of the two assessors.

The Similarity Assessor

The semantic similarity assessor aims at exploiting the structure of WordNet, which contains *per se* a certain degree of semantic information encoded in synsets. In the literature several approaches to compute semantic similarity have been proposed. In order to choose the most appropriate one, we evaluated results of several approaches and correlated them w.r.t human judgments of similarity. A detailed description of the dataset and evaluation methodology along with complete experimental results can be found at `http://grid.deis.unical.it/similarity.` Among the evaluated metrics, the most performant are these based on the notion of Information Content (IC). IC can be considered as a measure that quantifies the amount of information a concept expresses and is computed as log the negative likelihood of the occurrences of a concept in a large corpus. Resnik in [142] exploited the notion of IC for assessing semantic similarity between terms in a taxonomy. The basic intuition behind the use of the negative likelihood is that the more probable a concept is of appearing then the less information it conveys, in other words, infrequent words are more informative than frequent ones. Knowing the IC values for each concept, we may then calculate the similarity between two given concepts. In the lexical matcher we adapt the Jiang and Conrath distance metric (J&C) [84]. This metric computes the semantic similarity between two concepts c_s and c_p as follows:

$$sim(c_s, c_p) = 1 - \frac{IC(c_s) + IC(c_p) - 2 \cdot IC(msca(c_s, c_p))}{2}. \tag{6.2}$$

We consider the opposite of the semantic distance metric defined by J&C, as a similarity measure. Moreover, in order to quantify IC of concepts we exploit the function IC defined as follows [155]:

$$IC(c) = 1 - \frac{log(hypo(c) + 1)}{log(max_{wn})}. \tag{6.3}$$

where the function *hypo* returns the number of hyponyms of a given concept c. Note that concepts that represent leaves in the taxonomy will have an IC equals to one. Moreover, max_{wn} is a constant that indicates the total number of concepts in the WordNet noun taxonomy (i.e., 82115 in WordNet 3.0). The function $msca(c_s, c_p)$ in equation 6.2 returns the concept (the lowest in the taxonomy) that subsumes both c_s and c_p.

The Relatedness Assessor

In order to select the most appropriate relatedness assessor we evaluated several approaches. A complete description of the dataset and evaluation methodology along with complete experimental results is available at the similarity experiment website: `http://grid.deis.unical.it/similarity`. In our evaluation, we found that the gloss vector relatedness metric described in [130] is the most correlated w.r.t human judgment. This metric is based on the following intuition: the relatedness between two concepts can be assessed by comparing their glosses. In particular, this approach exploits ”second order” vectors for glosses, that is, rather than just matching words that occur in glosses, the words in the gloss are replaced with co-occurrence (extracted from a corpus) vectors. Therefore, each gloss is represented by the average of its word vectors. Hence, pairwise comparisons can be made between vectors to measure relatedness between the concepts they represent. In the following, we summarize the steps followed to compute the relatedness between two concepts c_s and c_p:

1. Get the gloss of c_s from WordNet. Create a gloss vector by adding the word vectors of all the words in the gloss.
2. Get the gloss of c_p from WordNet. Create a gloss vector by adding the word vectors of all the words in this gloss.
3. Compute the cosine of the gloss-vectors. In addition, this metric use the relations represented in Table 6.1 to augment the glosses of c_s and c_p, with gloss information of concepts that are directly linked to c_s and c_p. This makes the augmented glosses of c_s and c_p much bigger than the just the glossed of c_s and c_p.

If v_s and v_p are the gloss vectors for c_s and c_p, their relatedness in computed as follows:

$$relat(c_s, c_p) = \frac{v_s \cdot v_p}{|\overrightarrow{v_s}| \cdot |\overrightarrow{v_p}|}. \tag{6.4}$$

Computing the Overall Lexical Similarity Score

Overall, the *lexical similarity* is computed as a weighted sum of the scores provided by the two assessors:

$$sim_{lex}(c_s, c_s) = w_s \cdot sim(c_s, c_p) + w_r \cdot relat(c_s, c_p). \tag{6.5}$$

From experimental evaluation, we found that equally weighting the two contributions (i.e., assigning 0.5 to both w_s and w_r) gives the best accuracy in terms of correlation w.r.t human judgment.

Reducing the Time Elapsed

Since WordNet is a huge lexical database, some performance issues related to its access can arise. In order to provide a fast access to the database and implement our similarity and relatedness measures we built an ad-hoc Lucene index that maintains the information about synsets. In particular, both values of IC and gloss vectors are stored in the index. The index is built by parsing the Prolog release of WordNet [114].

A Running Example

In order to see how the lexical matcher works, let us compute the lexical similarity between the concepts *Animal* and *Person* represented in Fig. 6.8. According to equation 6.5 we have to compute the semantic similarity and relatedness between the two concepts.

Computing semantic similarity

For the semantic similarity, we have to calculate the following coefficients:

- IC(Animal): Animal in the WordNet taxonomy has 3998 hyponyms, therefore according to equation 6.3 we have that IC(Animal)=0.2670.
- IC(Person): Person has 6978 hyponyms, in this case we have: IC (Person)=0.2178.
- IC(Organism): Organism, which subsumes both concepts (see Fig. 6.8) has 16110 hyponyms. Therefore we have IC(Organism)=0.1439.

The semantic similarity value according to equation 6.2 is:

$$sim(Animal, Person) = 0.9014$$

Computing semantic relatedness

In order to compute semantic relatedness between *Animal* and *Person* we have to compare their glosses augmented with glosses of neighbors concepts. The neighbors concepts of a given concept are concepts related to it by any of the relations reported in Table 6.1. In our example, the gloss of the concept Person is *"a human being". The gloss of the concept Animal is "a living organism characterized by voluntary movement* ... ". Each of these concepts has a representative vector which contains for each dimension a number indicating the frequency of the word encoded in that dimension. Here we do not report the vectors of the two concepts since they have a very large dimension (about 12000). The semantic relatedness between *Animal* and *Person* is:

$$relat(Animal, Person) = 0.4667$$

Overall, the lexical similarity between *Animal* and *Person* is:

$$sim_{lex}(Animal, Person) = 0.5 \cdot 0.9014 + 0.5 \cdot 0.4667 = 0.6840$$

Compound Terms

The lexical matcher treats compound terms by following the heuristic that in English the last token appearing on the right side of a compound term denotes the central concept, while other concepts encountered from the left side to the right side to denote a qualification of its meaning [95].

6.2.3 The Contextual Matcher

The aim of the contextual matcher is to implement the *evaluate contextual similarity* function (see Fig. 6.4, line 6) exploited to refine similarity values assessed by the syntactic and/or lexical matcher. It advances a contextual approach to semantic relatedness that builds upon Miller et al. definition in terms of the interchangeability of words in contexts [113]. Contexts help to refine the search of correct mappings since they intrinsically contain both information about the domains in which concepts to be compared are used and their structure in terms of properties and neighbors concepts. Contexts represent possible patterns of usage of concepts and the contextual matcher is founded on the idea that similar concepts have similar patterns of usage. If two concepts can be used in a similar context then they are related. A concept c_s (i.e., *seeker* concept) in a context $ctx(c_s)$ (i.e., *seeker context*) not similar to a concept c_p (i.e., *provider* concept) in a context $ctx(c_p)$ (i.e., *provider context*) will likely fit bad into $ctx(c_p)$ as well as c_p will do in $ctx(c_s)$. Conversely, if the two concepts can be interchangeably used, that is fit well in each other's contexts, then they can be considered related. We call this strategy *how it fits* and, in order to quantify how well a concept fits in a context, we calculate the lexical similarity between the concept and all the concepts in the considered context and take the average value (see Fig. 6.7). The overall contextual similarity is computed by exploiting the following similarity indicators:

- *s2s*: indicates how the seeker concept fits in the seeker context
- *s2t*: indicated how the seeker concept context fits in the provider context
- *t2t*: indicated how the provider concept fits in the provider context
- *t2s*: indicates how the provider concept fits in the seeker context

The overall contextual similarity is calculated according to the following equation.

$$sim_{ctx}(c_s, c_p) = 1 - (|s2s - t2t| + |s2s - t2t|). \tag{6.6}$$

It is worth noting that this strategy aims at taking into account structural information about concepts on a local basis, that is, by only considering properties and nearest neighbor concepts in the taxonomy. This is justified by the fact that a complete mapping among peer ontologies is not required; they only need to map their part of ontologies contextual to the interaction in which they are involved. Moreover, in computing a concept mapping by SECCO, a *provider* peer is not aware of the whole ontology of the *seeker* peer. Here we provide a detailed evaluation of the contextual matcher on the two excerpts of ontologies depicted in Fig. 6.9. We consider *Book* in

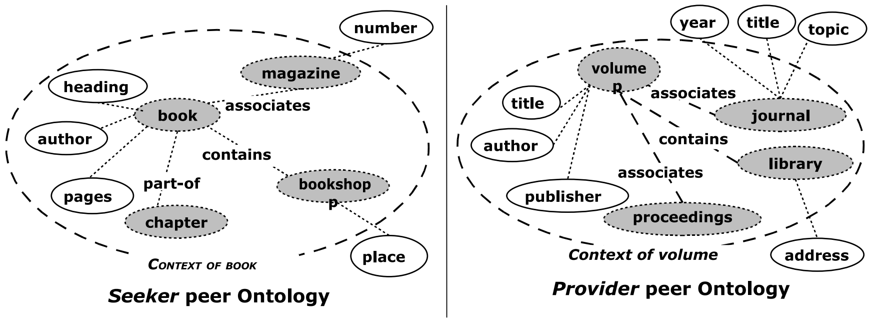


Fig. 6.9. Excerpts of a *seeker* and *provider* peer ontologies

the ontology of the *seeker* peer, as seeker concept, and *Volume* in the ontology of the

Table 6.2. Calculation of the *s2t* coefficient

***Seeker* Concept (Book)**	**Context of Volume**	**Lexical Similarity Value**
Book	Journal	0.8554
Book	Library	0.5102
Book	Proceedings	0.3961
Book	Title	0.5553
Book	Author	0.4735
Book	Publisher	0.3105

Table 6.3. Calculation of the *t2s* coefficient

***Provider* Concept (Volume)**	**Context of Book**	**Lexical Similarity Value**
Volume	Magazine	0.7088
Volume	Bookshop	0.2574
Volume	Chapter	0.3179
Volume	Heading	0.3279
Volume	Author	0.3944
Volume	Pages	0.3967

Table 6.4. Results of contextual similarity for some concepts in Fig. 6.9

***Seeker* Concept**	***Provider* Concept**	**Contextual Similarity**	**Time Elapsed**[2]
Book	Journal	0.60567	0.26
Book	Library	0.3278	0.25
Book	Proceedings	0.1789	0.28
Bookshop	Journal	0.3878	0.24
Bookshop	Library	0.8067	0.25
Chapter	Proceedings	0.60879	0.16
Chapter	Volume	0.22345	0.28

provider peer, as provider concept. In order to assess the contextual similarity between *Book* and *Volume* we start with calculating the coefficients defined in equation 6.6. In particular, Table 6.2 shows how the *s2t* coefficient is calculated. In this case the *s2t* coefficient value is 0.5168 and the time elapsed to compute it is 0.21 secs. Similarly, Table 6.3 shows how the *t2s* coefficient is calculated. In this case the *t2s* coefficient value is 0.4 and the time elapsed to compute it is 0.2 secs. In a similar way, SECCO computes values for *s2s* and *t2t*. In the considered example such values are: *s2s*=0.6578 and *t2t*=0.5467. The final contextual similarity between *Book* and *Volume* is 0.7057. Contextual similarity values for other couples of concepts are shown in Table 6.4.

Discussion of the Results

Similarity values obtained by the contextual matcher underline the fact that the contextual similarity between two concepts is affected by concepts and properties included in their contexts. For instance, even if *Book* and *Volume* can be linguistically considered very similar (their lexical similarity is 1), the contextual matcher correctly decreases their similarity value to 0.7057 (see equation 6.6) since they respectively appear in a *Bookshop* and *Library* context. Moreover, properties included in the definition of *Book* and *Volume* only share the concept of *author*. The highest contextual similarity value is obtained by the couple *Bookshop* and *Library*. Even being the two concepts linguistically not so much similar, their lexical similarity is 0.3467, if we only consider the contexts to which they belong, we can observe that the seeker context defines a *Bookshop* with *Place* as a property while the provider context defines a *Library* with *Address* as property. Both contexts refer to places containing books (one in which they are sold and another in which they are stored) that are characterized by an attribute indicating their location. In this case, the high similarity value obtained by the contextual matcher will be refined by the lexical similarity value (which is lower) when weighing their individual contributions.

By continuing to evaluate further results, the couple *Book* and *Proceedings* receives a low contextual similarity value. They are not lexically very similar (their lexical similarity is 0.3987) and their respective contexts represent different things. The seeker context defines a set of properties of a *Book* (e.g., *author*, *pages*) and provides relations with its constituent parts (i.e., *chapter*), with the place where it can be sold (i.e., *Bookshop*) and so forth. Conversely, the provider context just provides information about the fact that a *Proceedings* is related to a *Volume*. Similar considerations can be done for the other couples of concepts. In the light of these considerations, we can conclude that the contextual matcher is a suitable approach to interpret the use of concepts in different contexts. In fact, it can correctly interpret similarity between contexts, as in *Library* and *Bookshop*, while it is also able to interpret their dissimilarity, as in the case of the couple *Book* and *Proceedings*. However, it is worth noting that it becomes more effective when combined with the lexical matcher, as in the case of the couple *Book* and *Volume*. Finally, a consideration about elapsed time (see the last column of Table 6.4). We can see, as one can expect, that the elapsed time depends on the number of concepts/properties contained in the seeker and provider contexts. However, the elapsed time values, even in the case in which the dimension of the contexts in terms of number of concepts/properties is quite high (both the couples *Book* and *Volume* have 6 entities in total), never reach 0.3 secs.

6.2.4 Why Do We Need SECCO?

This section explains why SECCO has been designed and how it can be practically exploited. The main motivation for designing SECCO is to provide an ontology mapping algorithm in open environments (e.g., P2P, Grid). As pointed out in Chapter 3, there are several mapping algorithms, but there is a lack of algorithms especially designed for open environments. In such scenario, fastness is a mandatory requirement to perform *online* mapping and the amount of ontological information exploitable to discover mappings is quite limited. SECCO has been designed to provide the semantic foundation for the K-link+ system which will be described in chapters 8 to 10. K-link+ is a P2P system for collaborative work based on the concept of workspace. The system allows workers to work concurrently in the same and shared environment (i.e., workspace) by a set of tools for sharing and exchanging knowledge in a semantic way. In such an open architecture, it would be very useful to discover and interact with semantically neighbor peers. The concept of semantic proximity can be represented by exploiting SECCO. In fact, mappings discovered by SECCO, establish semantic links among peers of a K-link+ network. These links can be exploited in the following ways:

- *Semantic based search:* content (e.g., Web pages, documents) can be annotated to ontology concepts in order to provide them with an explicit and machine understandable semantic meaning. Therefore, content search can be performed by specifying ontology concepts instead of keywords. Retrieving similar concepts by SECCO will result in discovering content annotated to such concepts.
- *Semantic building of workspaces:* semantic links between peers are supposed to reflect common interests shared by the peers involved in the links. Therefore, by following these links peers with common interests can be discovered and grouped together.
- *Semantic query routing:* semantic links can be exploited to forward queries. When a query reaches a peer, it can forward this query to other peers with which it has semantic links. This way a new semantic path between ”unknown” peers can be constructed. Moreover, the amount of network traffic generated by queries (as compared with flooding techniques) can be significantly reduced by adopting a semantic-aware routing strategy.

We want to point out how, in designing a comprehensive semantic P2P solution, the central problem is to find out semantic links among peers. Once found, these can be exploited for several purposes. Therefore, differently from other approaches (e.g., [2, 72]) where the preexistence of mapping ensures semantic interoperability among peers, we provide a comprehensive solution that tackles the problem of designing semantic P2P systems from all the perspectives, that is, construction of the semantic overlay (provided by SECCO) and underling physical P2P architecture (provided by K-link+).

6.3 SECCO: a Double Evaluation

In this section, we show how the requirements that driven the design of SECCO are fulfilled in real case scenarios. In the previous section the matchers of SECCO have been described and how they cope with the requirements of fastness and accuracy has been shown. The syntactic matcher has been evaluated in Chapter 4. The lexical matcher has been extensively evaluated through the similarity experiment whose results are available at `http://grid.deis.unical.it/similarity`. The rationale of the contextual matcher has been described through the example depicted in Fig. 6.9. In this section, we want to evaluate SECCO as a whole. The evaluation has been split in two parts (referred to as *Experiment 1* and *Experiment 2* in the following). In *Experiment 1*, we evaluated SECCO by comparing it with H-Match [21, 23] that actually is the only system designed for mapping ontologies in open environments offering very similar features. Moreover, we perform a sensitivity analysis of the assignment of weights to the individual matchers and observe how results provided by SECCO in *Experiment 1* and the correlation w.r.t those produced by H-Match vary. In *Experiment 2*, we evaluate how SECCO performs as a general mapping algorithm. In this experiment, we evaluate it on four real-life ontologies included in the OAEI 2006 benchmark test suite [3] and compare its results with those of other algorithms not explicitly designed for ontology mapping in P2P networks. We evaluate SECCO only on ontologies 301-304 of the OAEI 2006 in order to have an indicator of how it performs with real-life ontologies.

6.3.1 Experiment 1: Comparing SECCO with H-Match

This section presents the comparison of SECCO w.r.t H-Match on two excerpts of (online available) ontologies. The first ontology (*Ka*) describes research projects while the second one (*Portal*) describes content of a Web portal. We suppose that *Ka* belongs to a seeker peer while *Portal* to a provider peer. These ontologies have also been adopted to evaluate the H-Match system as described in [21]. We have chosen to adopt the same two ontologies in order to have an objective comparison between the two approaches. Fig. 6.10 shows two excerpts of *Ka* and *Portal* describing the concept of *Publication* are shown. In this evaluation, we aim at constructing, by exploiting SECCO, a mapping between the concept *Publication* in *Ka* and some concepts belonging to *Portal*. In particular, we want to emphasize how SECCO can profitably discover similarities even among terms apparently not related and how it behaves w.r.t H-Match.

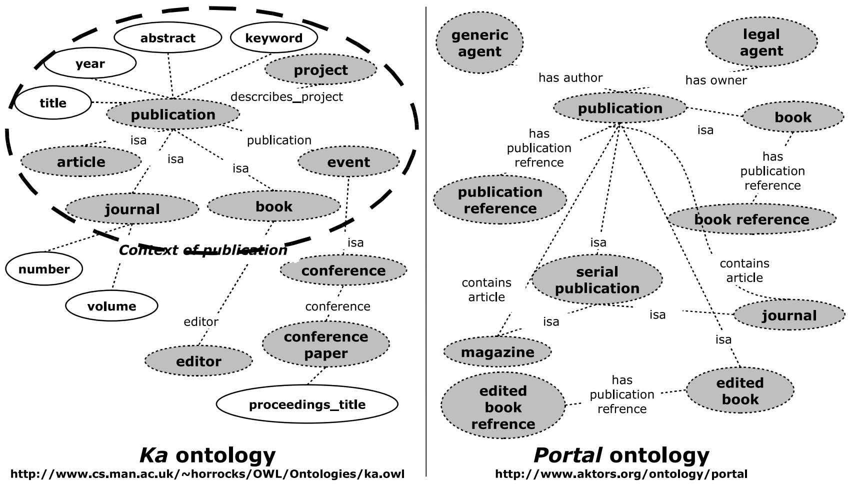


Fig. 6.10. Excerpts of *Portal* and *Ka* for the concept *Publication*

[3] `http://oaei.ontologymatching.org/2006`

Table 6.5. The input *I* of SECCO for Experiment 1

Parameter	**Value**
c_s Seeker Concept	Publication
$ctx(c_s)$ Seeker Context	ctx(Publication)
O Provider Ontology	Portal (the excerpt shown in Fig. 6.10)
Th Threshold	0
w_s Syntactic similarity weight	0.1
w_l Lexical similarity weight	0.6
w_c Contextual similarity weight	0.3

Table 6.6. Results considering *Publication* as a *seeker* concept

Ka concept	**Portal concept**	**Syntactic**	**Lexical**	**Contextual**	**Overall**	**Elapsed time (s)** [4]
Publication	Publication	1	1	0.697	0.909	0.49
Publication	Book	0	0.823	0.199	0.553	0.29
Publication	Journal	0	0.767	0.221	0.526	0.31
Publication	Magazine	0	0.737	0.088	0.468	0.29
Publication	Edited Book	0	0.823	0.674	0.696	0.29
Publication	Publication Reference	0.3	0.549	0.118	0.395	0.27
Publication	Book Reference	0	0.549	0.118	0.365	0.28
Publication	Edited Book Reference	0	0.549	0.118	0.365	0.31

Configuration of SECCO for Experiment 1

In this experiment, the input *I* of SECCO (see Section 6.1) takes the values shown in Table 6.5. We do not set a threshold value (the *Th* parameter) since we want to create one-to-many mappings. Since we want to give more emphasis to the semantic component of the algorithm, we consider lexical similarity more reliable than syntactic similarity or contextual similarity (i.e., we assign a higher value to w_l). A detailed analysis on how the assignment of weights can affect results will be provided afterward.

Results Obtained by SECCO in Experiment 1

Table 6.6 shows the results obtained by SECCO with the input *I* (see Table 6.5) along with overall elapsed times. These examples show the suitability of the *lexical matcher* which allows to discover mappings in a semantic way. In fact, by considering the analyzed couples of concepts only from a syntactic point of view we would obtain similarity values equal to 0 apart from the couples *Publication* (*Ka*) and *Publication* (Portal) and *Publication* (*Ka*) and *Publication Reference* (*Portal*). In the following, we compare these results with those obtained by H-Match.

Discussion of Results and Comparison with H-Match

Comparing ontology mapping algorithms is a hard task, especially when an objective and reliable reference alignment is not provided. Moreover in a P2P scenario, since

Table 6.7. SECCO vs. H-Match on the example of Fig. 6.9

		H-Match			
Couple	***SECCO***	**Shallow**	**Deep**	**Intensive**	**Average**
Book-Volume	0.8117	1	0.78	0.70	0.8266

a mapping algorithm usually aims at finding one-to-many mappings (it provides a similarity ranking between concepts) it is very difficult to interpret ranking values. In the literature, there exist very few algorithms that address the ontology mapping problem in P2P environments. The approach closest to SECCO is H-Match. In order to make an objective comparison between them, we considered the results obtained by H-Match for the same couples of concepts on which SECCO has been evaluated. In the example depicted in Fig. 6.9 authors in [21] provided only a similarity value (related to the couple *Book* and *Volume*). For this couple, H-Match obtained the results shown in Table 6.7. The overall similarity value between the couple *Book* and *Volume* obtained by SECCO is computed as follows:

$$sim(Book, Volume) = w_s * sim_{syn} + w_l * sim_{lex} + w_c * sim_{ctx}$$

Therefore, we obtain:

$$sim(Book, Volume) = 0.1 * 0 + 0.6 * 1 + 0.3 * 0.7057 = 0.8117$$

The lexical matcher correctly interprets the linguistic similarity between the *Book* and *Volume* concepts; in fact, it gives 1 as output. The high value of lexical similarity is because the *Book* and *Volume* concepts belong to the same WordNet synset and therefore are synonyms. Same things are valid for H-Match, whose shallow matching model has similar features to the lexical matcher of SECCO. The contextual matcher of SECCO, since *Book* and *Volume* respectively appear in a *Bookstore* context and a *Library* context, correctly decreases the overall similarity value (this aspect has been discussed is Section 6.2.3). H-Match obtains a similarity score of 0.78 with the intermediate matching model, which takes into account concept names and properties. Through the deep matching model, which considers the whole context of concepts (i.e., all the properties) H-match obtains 0.70. This matching model is the most similar to that implemented by SECCO. The average value given by H-Match, obtained by averaging results of the three matching models, is 0.8266, which is very close to the result obtained by SECCO. Therefore, in this case, we can conclude that the similarity value between *Books* and *Volume* obtained by SECCO is comparable with that obtained by H-Match. A more detailed comparison between the two approaches can be done by considering the two excerpts of the *Ka* and *Portal* ontologies depicted in Fig. 6.10. Similarity values obtained by both SECCO and H-Match [23] are shown in Table 6.8. Since we are interested in comparing only the semantic features of the two approaches we do not considered the contribution of the syntactic similarity of SECCO for the couples *Publication* (*Ka*) and *Publication* (*Portal*) and *Publication* (*Ka*) *Publication Reference* (*Portal*). In this experiment, it is interesting noting that the higher similarity values obtained by SECCO and H-Match are related to the cou-

Table 6.8. SECCO vs. H-Match on the example of Fig. 6.10

Ka concept	Portal concept	SECCO	H-Match				
			Surface	**Shallow**	**Deep**	**Intensive**	**Average**
Publication	Publication	0.909	1	0.7384	0.8047	0.7814	0.8318
Publication	Book	0.553	0.8	0.6184	0.66	0.6394	0.6795
Publication	Journal	0.526	0.64	0.5224	0.5538	0.5381	0.5636
Publication	Magazine	0.468	0.8	0.6184	0.6498	0.6341	0.6756
Publication	Edited Book	0.696	0.64	0.5224	0.5641	0.5434	0.5675
Publication	Publication Reference	0.395	0.64	0.5531	0.5741	0.5503	0.5794
Publication	Book Reference	0.365	0.64	0.5531	0.5733	0.5497	0.5790
Publication	Edited Book Reference	0.365	0.64	0.5531	0.5637	0.5420	0.5747

Table 6.9. Correlation between SECCO and H-Match

	H-**Match**				
	Surface	**Shallow**	**Deep**	**Intensive**	**Average**
SECCO	0.898	0.8559	0.9051	0.908	0.8919

ple *Publication* (*Ka*) and *Publication* (*Portal*). These two values are very close. Same considerations are valid for the couple *Publication* (*Ka*) and *Book* (*Portal*).

An interesting consideration can be done for the last three rows of Table 6.8. While SECCO obtains low similarity values, H-Match obtains values that always exceed 0.5. For instance, for the couple *Publication* and *Publication Reference*, SECCO obtains 0.395 while H-Match obtains 0.5794 as average result. However, by objectively analyzing the concepts, one can assess that these concepts are not much similar. In fact, the first describes the concept of publication while the second defines a reference to a publication. Indeed, it is very difficult to compare results between the two strategies with very few matching of couples, as in the case of *Book* and *Volume* (see Table 6.7). Moreover, comparing mapping results, without a reference alignment, implicitly includes a certain degree of subjective interpretation. In order to obtain an overall indicator of how the two approaches are (un)related, we computed the Pearson correlation coefficient [38] between their results. This coefficient represents an agreement between the values of two data sets (in our case between similarity results) by expressing the degree of association between them (see Table 6.9). As can be noted the higher value of correlation is 0.908 meaning that results obtained by SECCO are closer to these obtained by H-Match through the intensive matching model. Through this model of matching, H-Match considers both linguistic features of ontology concepts and the whole context of concepts (in terms of properties and semantic relations) in which they appear. Besides, also the correlation w.r.t the deep model is high. The average correlation value is 0.8919, which underlines how he two approaches are very close. In fact, a value of correlation higher that 0.7 can be interpreted as an indicator of high similarity [144]. It is very important noting that SECCO performs very close to those of H-Match even if SECCO does not adopt complex matching strategies. Since both approaches heavily rely on linguistic features of ontologies, we also computed the correlation (see Table 6.10) between re-

Table 6.10. *Lexical Matcher* compared to the H-Match *surface* matching model

	H-Match *surface*
SECCO *lexical matcher*	0.7123

sults of our *lexical matcher* that relies on WordNet and the surface matching model of H-Match that relies on an ad-hoc thesaurus built by exploiting WordNet. As can be noted, the value of correlation is high even if it is very difficult to estimate which approach is more accurate. However, the lexical matcher of SECCO is not an ad-hoc thesaurus but it is able to exploit the whole structure of WordNet by including in the similarity computation a wide set of semantic relations between concepts. Moreover, the metrics included in the lexical matcher have been extensively evaluated by the similarity experiment [5].

Discussion on Similarity Aggregation and Weights Assignment

SECCO, in order to perform similarity aggregation, adopts a weighted sum of similarity values given by the individual matchers. Do and Rahm [40] address some aspects of weights assignment and similarity aggregation for database structures. A similarity aggregation function is a function that takes results from several matchers, weights these results, and gives as output an overall similarity indicator. The weights are assigned manually or learned, e.g., using machine learning on a training set. Berkovsky et al. [8] have thoroughly investigated the effects of different weights on the alignment results. We chose to adopt a strategy based on multiple matchers since experimental results have shown that a combination of similarity measures (provided by different matchers) leads to better alignment results than using only one matcher at a time. We realize that this technique needs a certain degree of expertise from the SECCO user. In fact, if the different weights are not correctly assigned, mapping results can be affected. However, note that in a P2P scenario it is not possible to a priori analyze the structure of ontologies to be compared in order to find the best mapping strategy (as done in [77]) since peers are not aware of the ontologies of other peers. In the experiments, we manually settled values of the different weights (i.e., the w_s, w_l, w_c parameters). However, it would be interesting to see how the correlation coefficient w.r.t H-Match (*Experiment 1*) changes when assigning different weights. Table 6.11 shows correlation values between the two approaches by assigning different values of w_s, w_l and w_c. For sake of space, we do not report, for each variation of the weights, the similarity values obtained by SECCO. An interesting consideration arises from results shown in Table 6.11. As it can be noted, if we assign equal weights to the matchers (row 2) the correlation raise up to 0.9117 with the intensive model of H-Match and to 0.8974 in the average. Moreover, it is interesting to point out that if we assign a little higher value to the contextual matcher (row 4), the correlation remains high (0.8756 for the intensive model and 0.8438 in the average). Even in the case in which contextual similarity has a higher value (row

[5] The similarity experiment: `http://grid.deis.unical.it/similarity`

Table 6.11. Correlation between results of SECCO and H-Match

SECCO			H-Match				
w_s	w_l	w_c	**Surface**	**Shallow**	**Deep**	**Intensive**	**Average**
0.1	0.6	0.3	0.898	0.8559	0.9051	0.908	0.8917
0.3333	0.3333	0.3333	0.9023	0.8657	0.9102	0.9117	0.8974
0.3333	0.3333	0.3333	0.9023	0.8657	0.9102	0.9117	0.8498
0.3	0.4	0.3	0.8415	0.8367	0.8567	0.8645	0.8438
0.3	0.3	0.4	0.8218	0.8123	0.8657	0.8756	0.7886
0.1	0.3	0.6	0.7656	0.7567	0.8123	0.8198	0.4949

5), the average correlation value remains quite high. Conversely, if we give much more emphasis to the contextual matcher (row 6), the average correlation drastically decreases to 0.4949 in the average. As final remark, we can conclude that assigning equal weight to the matchers can increase the correlation value w.r.t H-Match, that does not necessarily mean better results since in the considered example alignments are not provided. However, in the light of these considerations in *Experiment 2* we assign equal weights to the different matchers.

6.3.2 Experiment 2: Comparing SECCO with the State of the Art

This section provides an extensive evaluation of SECCO on four real-life ontologies contained in the OAEI 2006 test suite [6]. We compared SECCO with other mapping algorithms not explicitly designed to tackle the OMP in P2P networks. This way we want to show how much the designing strategy of SECCO, which has to ensure fastness and cannot exploit the whole structures of ontologies to be mapped, affects accuracy (i.e., quality of results). We focused on the group of tests that contain four real-life ontologies (i.e. tests from 301 to 304) in order to investigate how SECCO performs in mapping real ontologies. For each of these ontologies the OAEI organizers provided a reference alignment. We computed measures of Precision (i.e., the number of correct mapping among all the mapping found), Recall (i.e., the number of correct mapping among all the existing mapping) and F-measure [39] (i.e., the harmonic mean of Precision and Recall). In particular, we compared results obtained by SECCO with those provided by the OAEI organizers. Note that SECCO, even being designed for P2P networks and therefore to work ”online”, can be exploited to compare entire ontologies by reiterating the process described in Section 6.1 for each concept in the source ontology (i.e., the reference ontology 101 contained in the OAEI tests).

Configuration of SECCO for *Experiment 2*

Table 6.12 shows the values of the input of SECCO for this experiment. Here we are interested in obtaining one-to-one mappings.

[6] http://oaei.ontologymatching.org/2006

Table 6.12. The input I of SECCO for Experiment 2

Parameter	**Value**
c_s Seeker Concept	Each concept c_i contained in the reference ontology (i.e., 101)
$ctx(c_s)$ Seeker Context	$ctx(c_i)$
O Provider Ontology	301-302-303-304
Th Threshold	0.51
w_s Syntactic similarity weight	0.333
w_l Lexical similarity weight	0.333
w_c Contextual similarity weight	0.3333

Results Obtained by SECCO in *Experiment 2*

Fig. 6.11 shows values of Precision, Recall and F-Measure obtained by SECCO. As can be noted, SECCO performs well. It always obtains a Precision around 0.9. The Recall, reaches the highest value (i.e., 0.9387) for ontology 304 while the lowest value (i.e., 0.6211) for ontology 302. However, it always remains higher than 0.5. The F-Measure values are 0.8269 for ontology 301, 0.7375 for ontology 302, 0.8012 for ontology 303 and 0.949 for ontology 304. Values of F-Measure that represent an overall indicator of the performance of a mapping algorithm are in all the cases high.

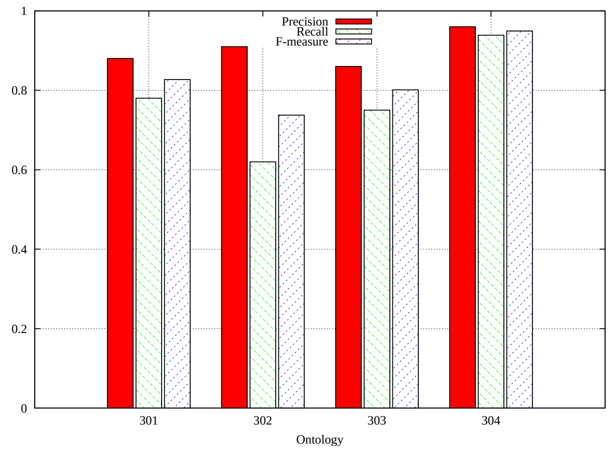


Fig. 6.11. Results of SECCO on the OAEI 2006 real life ontologies

Discussion of Results

In order to have an objective evaluation of SECCO, we decided to compare its average results with those of other ontology mapping approaches. The results are shown in Table 6.13. SECCO obtained an average Precision of 0.81, an average Recall of 0.81 and an average F-measure of 0.81. As can be noted, SECCO is one of the most precise algorithms. It is only slightly outmatched by Automs and Falcon. In terms of Recall SECCO is outperformed only by RiMOM. In terms of F-Measure, SECCO is only dominated by Falcon and RiMOM. An important consideration emerges from these results. SECCO is an ontology mapping algorithm that in its current implementation cannot exploit the whole structural information encoded in ontologies. Conversely, most of the presented approaches have a solid structural matching strategy. For instance, Falcon relies on the GMO approach [51] that exploits a graph-matching algorithm for discovering mappings while RiMOM exploits an adaptation of the Similarity Flooding algorithm. Such strategies require a complex analysis of the ontologies that is not conceivable in a P2P environment for two reasons: (i) peers are not aware of the whole ontologies of other peers; (ii) the fundamental requirement of fastness in P2P networks can be affected. It is worthwhile noting that SECCO obtains very good results without using that strategy.

Table 6.13. Results on the OAEI 2006 real-life ontologies

	SECCO	**Jhu/apl** [11]	**Automs** [90]	**Falcon** [77]	**RiMOM** [183]	**H-Match** [20]
Precision	0.81	0.18	0.91	0.89	0.83	0.78
Recall	0.81	0.50	0.70	0.78	0.82	0.57
F-Measure	0.81	0.26	0.79	0.83	0.82	0.65
Elapsed Time (s)	3.05	Na	70.25	7.22	3.14	Na

Note that the elapsed time by SECCO is the lowest. In particular, it is 25 times lower that that obtained by Automs that also exploits WordNet and about 3 times lower than that of Falcon, which adopts a structural matching strategy. Moreover, the comparison with H-Match, the system actually very similar to SECCO, shows how SECCO is better in terms of Precision, Recall and F-Measure. It would be also interesting to compare the approaches in terms of elapsed time but unfortunately, authors in [20] do not provide information about execution times. On one side, these results show how a structural mapping strategy can improve mapping results as in the case of Falcon. On the other side, they show that SECCO obtains results comparable with those of the most performant ontology mapping algorithms without adopting complex structural analysis of ontologies. Finally, we can conclude SECCO is faster than other mapping algorithms and the cost paid, in terms of accuracy, is not so high.

6.4 Discussion and Lesson Learned

This chapter described SECCO, an ontology mapping algorithm aimed at discovering concept mappings in P2P networks. A concept mapping has been defined as a similarity ranking between a request (composed by a concept along with its context) performed by a seeker peer and concepts belonging to provider peer ontologies.

Since we assume that peers are not aware of one another's ontologies, in order to discover mappings, we designed an ad-hoc mapping strategy. This strategy aims at fulfilling two important requirements (i.e., fastness and accuracy) through three individual matchers. The main problem we faced is related to the fact that we cannot adopt sophisticated and time-consuming structural matching strategies that require to know the whole two (peer) ontologies to be compared. Hence, we adopted the notion of context, defined as a concept along with its properties (obtained as described in Section 6.2.3) and nearest neighbor concepts. Through contexts, we aim at encoding the amount of structural information needed in a particular request. We compare the contextual information of different concepts by the "how it fits" strategy that is founded on the idea that two concepts are related if they fit well in each other's context. This strategy is supported by the lexical matcher whose aim is to exploit an accurate (proven by the similarity experiment [7]) similarity metric in WordNet. This metric allows assessing similarity even among syntactically unrelated concepts. Moreover, in order to exploit all the linguistic information of ontology entities (i.e. ontology metadata) we adopt the syntactic matcher. This matcher encodes linguistic information in virtual documents that are created and .compared by an information retrieval approach. All these matching strategies have been extensively evaluated. Along the chapter, we discussed the exploiting of SECCO in the context of P2P networks and proven through experimental evaluation the suitability of the algorithm. In particular, SECCO has been compared *Experiment 1* with the H-Match algorithm, designed for ontology mapping in open environments, with very promising results. Furthermore, SECCO has been compared *Experiment 2* with other mapping algorithms not explicitly designed for mapping in P2P networks and even in this case results are satisfactory. We also performed a sensitivity analysis from which emerged an interesting aspect related to weight assignments to the different matchers.

[7] `http://grid.deis.unical.it/similarity`

7

Computing Semantic Similarity between Ontology Concepts

Assessing semantic similarity between words is a central issue in many research fields such as Psychology, Linguistics, Cognitive Science, Biomedicine, and Artificial Intelligence. Semantic similarity can be exploited to improve accuracy of current Information Retrieval techniques (e.g., [97, 75]), to discover mapping between ontology entities (see Chapter 6), to validate or repair mappings [110], to perform word-sense disambiguation [141]. Recently, Li and colleagues in [101] have proposed a methodology to compute similarity between short sentences through semantic similarity. Semantic similarity has found its way also in the context of Peer-to-Peer networks (e.g., [71]). In particular, assuming a shared taxonomy among the peers to which they can annotate their content, semantic similarity is exploited to infer similarity among peers by computing similarity among their representative concepts in the shared taxonomy, this way, the more two peers are similar the more efficient is to route messages toward them. In [143] are discussed several applications of similarity in Artificial Intelligence. In the biomedical domain there exist some applications to compute semantic similarity between concepts of ontologies such as MeSH or Gene (e.g., [132]). However, despite the numerous practical applications of semantic similarity, it is important pointing out its theoretical underpinning in Cognitive Science and Psychology where several investigations (e.g., [150]) and theories (e.g., [112, 170]) have been proposed.

As a matter of fact, semantic similarity is relevant in many research areas and therefore, designing accurate methods is mandatory for improving the ”performance” of the bulk of applications relying on it. Basically, similarity or distance methods (e.g.,[18]) aim at assessing a score between a pair of words by exploiting some information sources. These can be search engines (e.g., [30, 36]) or a well-defined semantic network such as WordNet [113] or MeSH [1]. To date, several approaches to assess similarity have been proposed ([82] provide an exhaustive list of references) which can be classified on the basis of the source of information they exploit. There are ontology-based approaches (e.g., [138]), information-theoretic ap-

[1] `http://www.nlm.nih.gov/mesh`

proaches which exploit the notion of Information Content (IC) (e.g., [103, 84, 142]), hybrid approaches (e.g., [100, 145, 153]) just to cite a few.

In this chapter, our purpose is to systematically design, evaluate and implement a new similarity metric. This metric has not to be derived empirically but has to be justified by a theoretical underpinning (i.e., a theory of semantic similarity). The contributions of this chapter can be summarized as follows:

1. We propose a new similarity metric which exploits some of the early work on the feature based theory of semantic similarity proposed by Tversky [170], and projects it into the information theoretic domain which has attained impressive results. As our results will show, this metric coupled with the notion of *intrinsic* Information Content [155] outperforms current implementations on different datasets.
2. We performed a similarity experiment to collect human similarity ratings to evaluate similarity metrics. In particular, we used the 65 word pairs dataset originally proposed by Rubenstein and Goodenough (R&G)[146]. Note that even if similar experiments have been carried out during the years (e.g., [113, 142]), none of these considered the whole R&G set of 65 word pairs. As we will discuss, the number of participants in our experiment is significantly higher than that of other experiments and hence we hope to provide a more robust and reliable evaluation tool. Moreover, by correlating our ratings with those collected by R&G we investigate a possible upper-bound for results that we can expect from computational methods.
3. We evaluated the proposed metric on different datasets and analyzed its structure to identify commonalities and differences w.r.t the state of the art. Moreover, we evaluated the impact of the *intrinsic* IC formulation on our and other IC based metrics.
4. We implemented our metric and several others in the Java WordNet Similarity Library (JWSL). JWSL, to the best of our knowledge, is the only tool written in Java devoted to compute similarity in WordNet. JWSL, by exploiting an ad-hoc index, allows to speed up the similarity computation without requiring the WordNet software to be installed.

The remainder of this chapter is organized as follows. Section 7.1 provides some background information regarding WordNet and popular similarity metrics. Section 7.2 presents our similarity metric and the intuitions that motivated its origin. In Section 7.3 we explain how the new dataset was created and compare it with previously used datasets. Section 7.4 uses the new dataset to analyze and compare several similarity metrics, by correlating them to the human assessments. Moreover, here we evaluate the impact of the *intrinsic* IC formulation. In this section we also propose a new upper bound on the degree of correlation that may be obtained using computational approaches and briefly introduce the JWSL. Finally, Section 7.5 concludes.

7.1 Background on Semantic Similarity Metrics

WordNet is a light-weight lexical ontology where concepts are connected to each other by well-defined types of relations. It is intended to model the human lexicon, and took psycholinguistic findings into account during its design [112]. We call it a light-weight ontology because it is heavily grounded on its taxonomic structure that employs the IS-A inheritance relation and lexical ontology because it contains both linguistic and ontological information. In WordNet concepts are referred to by different words; for example if we want to refer to the concept expressed by ”someone deranged and possibly dangerous” we could use any of the words contained in the set {*crazy*, *loony*, *looney*, *weirdo*}. So in a given context we can say that the words in the above set are synonyms. Hence, a synset (Synonym Set), the term adopted by the founders of WordNet, represents the underlying lexical concept. Each concept contains a gloss that expresses its semantics by means of a textual description and a list of words that can be used to refer to it. There are several types of relations used to connect the different types of synsets. Some of these define inheritance (IS-A) relations (hypernymy/hyponymy), other part-of relations (holonymy/meronymy). The antonymy relation is used to state that a noun is the opposite of another. The relations *instance of* and *has instance* have been introduced in WordNet 3.0. However, note that the hypernymy/hyponymy relations constitute 65% of the relations connecting noun synsets. The prototypical definition of a noun consists of its immediate superordinate followed by a relative clause that describes how this instance differs from all other instances. For example, *Fortified Wine* is distinguished from *Wine* because *”... alcohol (usually grape brandy)”* has been added just as the gloss mentions. This type of model is usually said to employ a differential theory of meaning, where each subordinate differentiates itself from its super ordinate.

7.1.1 Similarity Metrics on WordNet

Similarity metrics between concepts can be divided into four general, and not necessarily disjoint, categories [185]: Ontology Based Approaches, Corpus Based Approaches, Information Theoretic and Dictionary based approaches. In this chapter we will focus on popular metrics that may use WordNet as their main knowledge resource and that belong to either the information theoretic or ontology based category. A complete survey of existing metrics is out of the scope of this chapter (for a list of related references refer to [82]).

Information theoretic approaches usually employ the notion of Information Content (IC), which can be considered a measure that quantifies the amount of information a concept expresses. Previous information theoretic approaches [142, 84, 103] obtain the needed IC values by statistically analyzing corpora. They associate probabilities to each concept in the taxonomy based on word occurrences in a given corpus. These probabilities are cumulative as we go up the taxonomy from specific concepts to more abstract ones. This means that every occurrence of a noun in the corpus is also counted as an occurrence of each taxonomic class containing it. The IC value is obtained by considering negative the log likelihood:

$$IC(c) = -log p(c) \ . \tag{7.1}$$

where c is a concept in WordNet and *p(c)* is the probability of encountering c in a given corpus. It should be noted that this method ensures that IC is monotonically decreasing as we move from the leaves of the taxonomy to its roots. Resnik [142] was the first to consider the use of this formula, which stems from the work of Shannon [157], for the purpose of semantic similarity judgments. The basic intuition behind the use of the negative likelihood is that the more probable a concept is of appearing then the less information it conveys, in other words, infrequent words are more informative than frequent ones. Knowing the IC values for each concept we may then calculate the similarity between two given concepts. According to Resnik, similarity depends on the amount of information two concepts have in common, this shared information is given by the Most Specific Common Abstraction (*msca*) that subsumes both concepts. In order to find a quantitative value of shared information we must first discover the *msca*, if one does not exist then the two concepts are maximally dissimilar, otherwise the shared information is equal to the IC value of their *msca*. Resniks formula is modeled as follows:

$$sim_{res}(c_1, c_2) = max_{c \in S(c_1, c_2)} IC(c) \ . \tag{7.2}$$

where $S(c_1, c_2)$ is the set of concepts that subsume c_1 and c_2. This formulation of similarity is actually very similar to the one proposed by Tversky [170] but using a set theoretic framework. Following Resniks first work two other distinguishable metrics where postulated, that of Jiang and Conrath [84] and the work of Lin [103]. Both metrics used the notion of IC and calculated it in the same manner proposed by Resnik. Both Lin's and Jiang's formulations correct a problem with Resniks similarity metric; if one were to calculate $sim_{res}(c_1, c_1)$ one would not obtain the maximal similarity value of 1, but instead the value given by IC(c_1). Moreover, with this approach any two pairs of concepts having the same *msca* have exactly the same semantic similarity. For example, $sim_{res}(horse, plant) = sim_{res}(animal, plant)$ because in each case the *msca* is *Living Thing*. According to Lin *"The similarity between c_1 and c_2 is measured by the ratio between the amount of information needed to state the commonality of c_1 and c_2 and the information needed to fully describe what c_1 and c_2 are"*. Formally this formula is given in the following equation:

$$sim_{Lin}(c_1, c_2) = \frac{2 \cdot sim_{res}(c_1, c_2)}{IC(c_1) + IC(c_2)} \ . \tag{7.3}$$

The Jiang et al. metric is a semantic distance measure, but as shown in [154] it can be transformed to a similarity metric yielding:

$$sim_{J\&C}(c_1, c_2) = 1 - \frac{IC(c_1) + IC(c_2) - 2 \cdot sim_{res}(c_1, c_2)}{2} \ . \tag{7.4}$$

Regarding the ontology based approaches we review two noteworthy approaches, one of Rada et al. [138] and the one of Hirst et al. [74]. The first is also referred to as a depth based approach and the second as a path based approach. The Rada metric is similar to the Resnik metric in that it also computes the *msca* between two concepts, but instead of considering the IC as the value of similarity, it considers the number of links that were needed to attain the *msca*. Obviously, the less number of links separating the concepts the more similar they are. The approach of Hirst et al.[2] is similar to the previous but instead they use all types of relations in WordNet coupled with rules that restrict the way concepts are transversed. Nonetheless, the intuition is the same; the number of links separating two concepts is inversely proportional to the degree of similarity. Finally, an approach combining structural semantic information in a nonlinear model is that proposed by Li et al. [100]. The authors empirically defined a similarity measure that uses shortest path length, depth and local density in a taxonomy. The next equation reflects their metric:

$$sim_{Li}(c_1, c_2) = \begin{cases} e^{-\alpha l} \frac{e^{\beta h} - e^{-\beta h}}{e^{\beta h} + e^{-\beta h}} & \text{if } c_1 \neq c_2 \\ 1 & \text{if } c_1 = c_2 \end{cases} \tag{7.5}$$

In equation 7.5, l is the length of the shortest path between c_1 and c_2 in the graph spanned by the IS-A relation, h is the level in the tree of the *msca* from c_1 and c_2. The parameters α and β represent the contribution of the shortest path length l and depth h. The optimal values for these parameters, determined experimentally, are: $\alpha = 0.2$ and $\beta = 0.6$ as discussed in [100].

7.2 The Pirró and Seco (P&S) Similarity Metric

In this section we introduce our new similarity metric which is conceptually similar to the previous ones, but is founded on the feature-based theory of similarity posed by Tversky [170]. We argue that this theory fits nicely into the information theoretic domain, and obtains results that improve the current state of the art. Moreover, to avoid the problem of corpus-dependence of IC based metrics we exploit the method discussed in Section 7.2.1. The argumentation presented here follows from the work conducted in [154].

Tversky presented an abstract model of similarity that takes into account the features that are common to two concepts and also the differentiating features specific to each. More specifically, the similarity of a concept c_1 to a concept c_2 is a function of the features common to c_1 and c_2, those in c_1 but not in c_2 and those in c_2 but not in c_1. Admitting a function $\psi(c)$ that yields the set of features relevant to c, he proposed the following similarity function:

$$sim_{tvr}(c_1, c_2) = \alpha \cdot F(\Psi(c_1) \cap \Psi(c_2)) - \beta \cdot F(\Psi(c_1)/\Psi(c_2)) - \gamma \cdot F(\Psi(c_2)/\Psi(c_1)) \ . \tag{7.6}$$

where F is some function that reflects the salience of a set of features, and α, β and γ are parameters that provide for differences in focus on the different components. According to Tversky, similarity is not symmetric, that is, $sim_{tvr}(c_1, c_2) \neq sim_{tvr}(c_2, c_1)$

[2] This approach actually measures relatedness, but since similarity is a special case of relatedness (see [154])we consider it in our study

because subjects tend to focus more on one object than on the other depending on the way the comparison experiment has been laid out. Obviously, the above formulation is not framed in information theoretic terms. Nonetheless, we argue that a parallel may be established that will lead to a new similarity function. Resnik considered the *msca* of two concepts c_1 and c_2 as reflecting the information these concepts share, which is exactly what is intended with the intersection of features from c_1 and c_2 (i.e., $\Psi(c_1) \cap \Psi(c_2)$). Now, remembering that function F quantifies the salience of a set of features, then we postulate that we may find that quantification in the form of information content. The above reasoning will lead us to the following analogy represented in the following equation:

$$\begin{aligned} sim_{res}(c_1,c_2) = {} & IC(msca(c_1,c_2)) \approx F(\Psi(c_1) \cap \Psi(c_2)) \\ & = 1 \cdot F(\Psi(c_1) \cap \Psi(c_2)) - 0 \cdot F(\Psi(c_1)/\Psi(c_2)) - 0 \cdot F(\Psi(c_2)/\Psi(c_1)) \end{aligned} \quad (7.7)$$

Since the *msca* is the only parameter taken into account we may say that his formulation is a special case of equation 7.6 where β=γ=0. The above discussion lends itself to the proposal of an information theoretic counterpart of equation 7.7 that can be formalized as:

$$\begin{aligned} sim_{tvr'}(c_1,c_2) = {} & IC(msca(c_1,c_2)) - (IC(c_1) - IC(msca(c_1,c_2)) - (IC(c_2) - IC(msca(c_1,c_2)) \\ & = 3 \cdot IC(msca(c_1,c_2)) - IC(c_1) - IC(c_2) \end{aligned} \quad (7.8)$$

A careful analysis of equation 7.8 shows that this metric suffers from the same problem as Resnik's metric. When computing the similarity between identical concepts the output yields the information content value of their *msca* and not the value corresponding to maximum similarity. In order to overcome this limitation we assign the value of 1 if the two concepts are the same, hence yielding the similarity metric that can be formalized as follows:

$$sim_{P\&S}(c_1,c_2) = \begin{cases} sim_{tvr'} & \text{if } c_1 \neq c_2 \\ 1 & \text{if } c_1 = c_2 \end{cases} \quad (7.9)$$

Note that in equation 7.9 we use $sim_{tvr'}$ which is the information theoretic counterpart of Tversky's set theoretic formulation. This new formulation will be dubbed as the $sim_{P\&S}$ metric in the rest of the chapter. At this point a possible drawback related to IC-metrics remains to be solved: how to obtain IC values in a more direct and corpus-independent way ? We address this problem in the next section.

7.2.1 Intrinsic Information Content

As pointed out before, similarity metrics grounded on IC obtain IC values for concepts by statistically analyzing large corpora and associating a probability to each concept in the taxonomy based on its occurrences within the considered corpus. From a practical point of view, this approach has two main drawbacks: (i) it is time consuming and (ii) it heavily depends on the type of corpora considered. Research toward mitigating these drawbacks has been proposed by Seco et al. [155]. Here, values of

IC of concepts rest on the assumption that the taxonomic structure of WordNet is organized in a ”meaningful and structured way”, where concepts with many hyponyms convey less information than concepts that are leaves, that is, the more hyponyms a concept has the less information it expresses. Hence, the IC for a concept c is defined as:

$$IC(c) = 1 - \frac{log(hypo(c) + 1}{log(max_{wn})} \ . \tag{7.10}$$

where the function *hypo* returns the number of hyponyms of a given concept c. Note that concepts that represent leaves in the taxonomy will have an IC of one, since they do not have hyponyms. The value of 1 states that a concept is maximally expressed and cannot be further differentiated. Moreover max_{wn} is a constant that indicates the total number of concepts in the WordNet noun taxonomy. This definition of IC will be exploited in the P&S similarity metric thus enabling to obtain IC values in a corpus independent way. In Section 7.4.4 we will show how the intrinsic IC improves the accuracy of all IC based metrics.

7.3 The Similarity Experiment

In order to assess the quality of a computational method to determine similarity between words, that is, its accuracy, a natural way is to compare its behavior w.r.t human judgments. The more a method approaches human similarity judgment the more accurate it is. In evaluating the different methodologies two datasets are commonly used, those of Rubenstein and Goodenough (R&G in the following) and Miller and Charles (M&C in the following). R&G [146] in 1965 performed a similarity experiment by providing 51 human subjects, all native English speakers, with 65 word pairs and asking them to assess similarity between word pairs on a scale from 0 (”semantically unrelated”) to 4 (”highly synonymous”). M&C [113], 25 years later, repeated the R&G experiment by only considering a subset of 30 word pairs from the original 65, and involving 38 undergraduate students (all native English speakers). In this case humans were also asked to rate similarity between pairs of words on a scale from 0 to 4. Although the M&C experiment was carried out 25 years later, the correlation between the two sets of human ratings is 0.97 which is a very remarkable value considering the diachronic nature of languages. Resnik [142] on his turn in 1995 replicated the M&C experiment by involving 10 computer science graduate students and post-doc (all native English speakers) obtaining a correlation of 0.96, also in this case a high value.

The results of these experiments point out that human knowledge about semantic similarity between words is remarkably stable over years (25 and 30 years later the R&G, for the M&C and Resnik experiment respectively). Moreover, they also point out how the usage of human ratings could be a reliable reference to compare computational methods with. However, researchers tend to focus on the results of the M&C experiment to evaluate similarity metrics and, to the best of our knowledge, no systematic replicas of the entire R&G experiment have been performed. Therefore, we argue that it would be valuable to perform a ”new” similarity experiment in order to obtain a baseline for comparison with the entire R&G dataset.

7.3.1 Experiment Setup

We replicate the R&G experiment (naming it Pirró and Seco, P&S in the following) but one step closer to the 21st century, the century of the Internet and global information exchange. In particular, we performed the experiment on the Internet by advertising it in some of the most famous computer science mailing lists (e.g., DBWORLD, CORPORA, LINGUIST) with the aim to involve as many people as possible. Each participant, after a registration process on the similarity experiment website[3] could take part in the experiment. In the website were provided all the instructions to correctly perform the experiment. The similarity scores along with the emails provided by the participants have been stored in a relational database for subsequent analysis. As one can imagine, and as our results confirmed, the participants were mostly graduate students, researches and professors. Note that we also opened the experiment to non native English speakers. As said above, in the era of globalization more and more people speak English thus participating in the creation and spreading of new forms of interpreting terms. Furthermore, semantic relations among words are affected by language evolution that, on its turn, is affected by the presence of a larger number of speakers of a particular language. Our objective is to investigate if and how the presence of non native speakers affects similarity judgments. Among the participants, about 70% are native American English speakers, 30% British English speakers while non native speakers are for the most part European. Table 7.1 provides some information about the experiment. As can be noted, even if we collected 121 similarity ratings we discarded some of them for the reasons explained in the next section.

Table 7.1. Information about the *P&S* experiment

Start of the experiment	07/15/2007
Result considered until	04/15/2008
Overall number of similarity judgments collected	121
Number of similarity judgments considered in the gold standard	101
Number of similarity judgments provided by native English speakers	76
Number of similarity judgments provided by non native English speakers	25

7.3.2 Elaborating the Collected Similarity Ratings

In order to design a systematic experiment and consider its results reliable, an a posteriori analysis of its results is required. In our case, this analysis is particularly important for ratings provided by non native speakers since the group of non native

[3] All the details along with extensive evaluations are downloadable at the JWSL website `http://grid.deis.unical.it/similarity`

speakers could be quite large and heterogeneous, ranging from near-native speakers over very fluent speakers to speakers with only rudimentary knowledge of English. In order to check the quality of the ratings provided by the participants, we calculate, for each participant, a rating coefficient (i.e., C) defined as follows:

$$C = \sum_{i=1}^{65} |C_i - avg_i| \, . \tag{7.11}$$

In particular, for each word pairs the distance between the score provided by the participant and the average score provided by the others is measured. The distance values for all the 65 pairs are then summed up. Once computing all the coefficients C we could discard the participants that present values of C differing too much from the average. Fig. 7.1 represents the C values for all the 121 participants. As can be

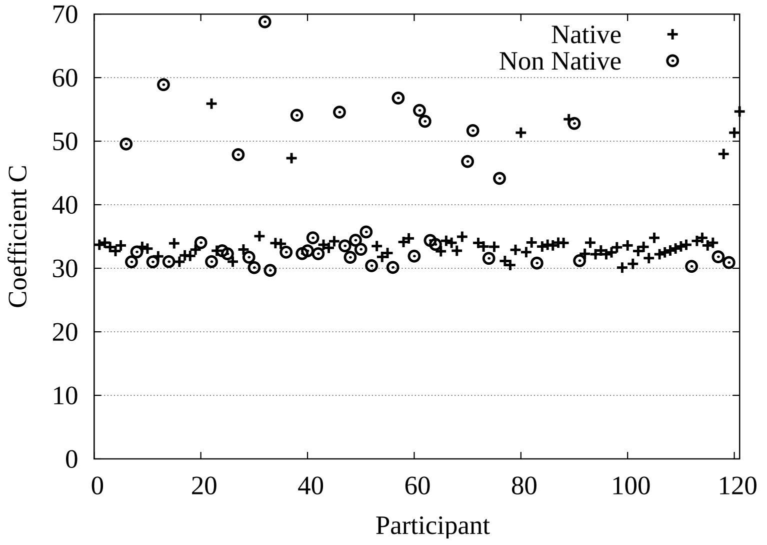


Fig. 7.1. Values of the coefficient C for the participants to the P&S experiment

noted, most of the C coefficients lie between 30 and 40. However, ratings provided by some participants (and then C coefficients) clearly differer from the average. The ratings provided by these participants have been discarded. In particular, by observing the results provided in Fig. 7.1 it can be noted that the anomalous ratings were for the most part given by non native speakers (about 90%). Table 7.2 provides an overall view of the different similarity experiments. Note that even if we collected 121 similarity ratings, we only considered 101 as reliable. We collected a larger number of similarity ratings than R&G, M&C and Resnik experiments and about 30% of participants in our experiment are (reliable) non native English speakers. Moreover, differently from M&C and Resnik we performed the experiment by considering the whole initial R&G dataset.

Table 7.2. Overall view of the different similarity experiments

Experiment	Year	Number of pairs	Number of participants
R&G	1965	65	51 (all native speakers)
M&C	1991	30	38 (all native speakers)
Resnik	1995	30	10 (all native speakers)
P&S	2008	65	101(76 native speakers and 25 non native)

7.3.3 Comparison among Experiments

We split the collected similarity judgments in two sets. The first set ($S_{M\&C}$ in the following) contains the judgments for the 28 word pairs in the M&C experiment. We consider only 28 pairs of the initial 30 used by M&C since due to a word missing in WordNet it is only possible to obtain computational rating for 28 word pairs. The second set ($S_{R\&G}$ in the following) contains the 65 word pairs in the R&G dataset. In particular, this latter dataset is used to define a possible upper-bound for computational methods to assess semantic similarity. Note that the word pairs in M&C, extracted from the original R&G dataset, are chosen in a way that they range from ”highly synonymous” (e.g., *car-automobile*) to ”semantically unrelated” (i.e., *cord-smile*). In order to have a more accurate view of the values of the ratings provided by the different experiments and investigate the regularity of the decreasing similarity trend demanded by R&G, we considered the similarity ratings of the four experiments as virtually connected, thus obtaining the representation in Fig. 7.2. As can

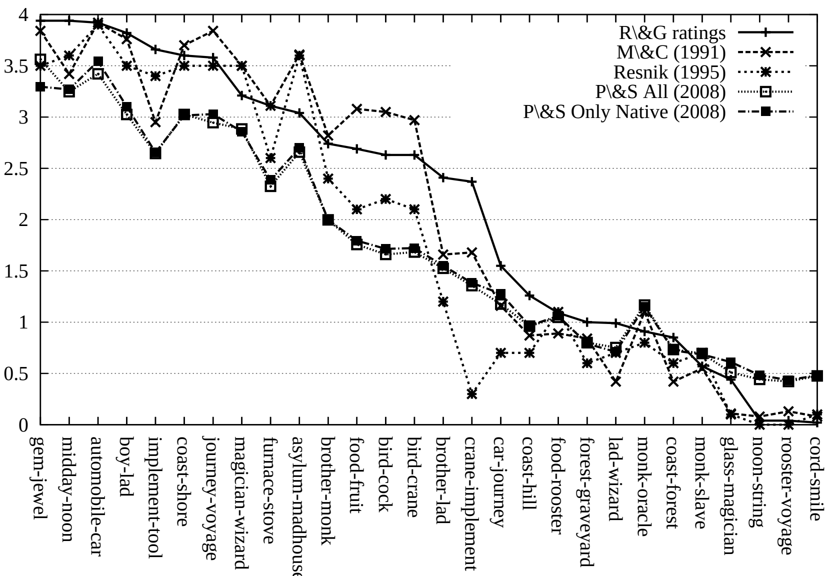


Fig. 7.2. Human ratings collected by the different experiments

be observed in Fig. 7.2, the decreasing trend of the R&G judgments is quite regular whereas that of M&C is quite irregular due to some pairs of concepts that are judged more similar/dissimilar by participants to the M&C experiment.

For instance, the pairs *implement-tool*, *asylum-madhouse*, and *brother-lad* are evident alterations of the decreasing trend of the R&G rating curve. The Resnik rating curve seems to be the most irregular, in particular, the pairs *furnace-stove*, *asylum-madhouse* and *crane-implement* are evident singular points. The P&S rating curves considering native speakers and all speakers also present some singular points (e.g., *implement-tool* and *asylum-madhouse*).

As a final comparison, in Tables 7.3 and 7.4 the Pearson correlation coefficient among the different experiments are reported. For sake of space we do not report the scores obtained for the whole $S_{R\&G}$ dataset. As can be noted, the correlation values obtained by our experiment are high. In particular, the correlation values considering only native ($P\&S_{nat}$) and all the participants ($P\&S_{full}$) are almost the same. Therefore, we argue that results of the P&S experiment can be adopted as a reliable basis for comparing similarity metrics with. Moreover, since the number of judgments collected is larger than that collected by previous experiments and the presence of non native speakers does not affect the similarity judgments we hope to provide a more reliable and robust evaluation tool.

Table 7.3. Correlation among experiments on $S_{M\&C}$

Experiment	$P\&S_{full}$	$P\&S_{nat}$
R&G(1965)	0.961	0.964
M&C(1991)	0.951	0.955
Resnik(1995)	0.970	0.972

Table 7.4. Correlation among experiments on $S_{R\&G}$

Experiment	$P\&S_{full}$	$P\&S_{nat}$
R&G(1965)	0.972	0.971

7.3.4 Some Statistical Indicators

To complete the elaboration of the collected results, we computed two additional parameters: (i) the inter-subject agreement also known as kappa-statistic and (ii) the correlation between ratings provided by native and non native speakers. Considering the $S_{M\&C}$ the kappa-statistic obtained is 0.82 which symbolizes the agreement among participants in rating the word pairs. On the same set, the correlation between the average judgments of native and non native speakers is 0.97, which is a very high value. Considering the $S_{R\&G}$ the kappa-statistic obtained is 0.81 while the correlation between the average judgments of non native and native speakers in this case is 0.98. Finally, note that the experiments involved a different number of participants (51 for R&G, 30 for M&C, 10 for Resnik and 101 for P&S).

7.4 Evaluation and Implementation of the P&S Metric

In this section, to substantiate the investigation that led to the definition of the P&S metric we evaluate and compare it w.r.t the state of the art. In performing this evaluation we consider the results of the P&S experiment on the $S_{M\&C}$ and $S_{R\&G}$ datasets. All the evaluations have been performed using WordNet 3.0.

In our evaluation, instead of reporting for each metric the results obtained in a tabular form we represent them as shown in Fig. 7.3. This way, we can further discuss and characterize in more details the peculiarities, analogies and differences of the different metrics. However, to have an overall view of the outcome of our evaluation and compare the different metrics we calculated, for each metric, the Pearson correlation coefficient between its results and human judgments (see Tables 7.5 and 7.6).

Table 7.5. Correlation of the different metrics on $S_{M\&C}$

	P&S (2008)	
	$P\&S_{full}$	$P\&S_{nat}$
Length	0.611	0.602
Depth	0.841	0.839
Resnik	0.854	0.842
Lin	0.875	0.871
J&C	0.884	0.883
Li	0.911	0.904
P&S	0.912	0.908

Table 7.6. Correlation of the different metrics on $S_{R\&G}$

	P&S (2008)	
	$P\&S_{full}$	$P\&S_{nat}$
Length	0.587	0.578
Depth	0.807	0.805
Resnik	0.877	0.869
Lin	0.892	0.888
J&C	0.878	0.877
Li	0.900	0.897
P&S	0.908	0.905

The similarity values for the Length and Depth metrics are obtained by considering the shortest path between the two words to be compared and the depth of their subsumer respectively. For the metrics based on IC and the P&S metric the values of IC are obtained by the method described in Section 7.2.1. Moreover, for the Li metric the similarity results are those reported in [100].

7.4.1 Discussion

From the values reported in Tables 7.5 and 7.6 emerges that edge counting approaches reach the lowest correlation with human ratings. That is mainly due to the fact that path lengths and depth approaches are appropriate only when the values of path and depth have a ”consistent interpretation”. This is not the case of WordNet, since concepts higher in the hierarchy are more general than those lower in the hierarchy. Therefore, a path of length one between two general concepts can suggest a larger semantic leap whereas one between two specific concepts may not (e.g., *Entity – Psychological Feature* and *Canine – Dog*). Resnik’s metric, which only considers the IC of the *msca* in assessing semantic similarity, obtained the lowest value of correlation among the IC metrics using $S_{M\&C}$. The Lin and J&C metrics, which also consider the IC of the two words to be compared, obtained higher values of correlation on the same dataset. Note that the Li metric which combines the depth of the *msca* and the length of the path between two concepts to be compared obtained a remarkable value of correlation even if it relies on two coefficients (i.e., α and β) whose optimal values have been experimentally determined as described in [100]. The J&C

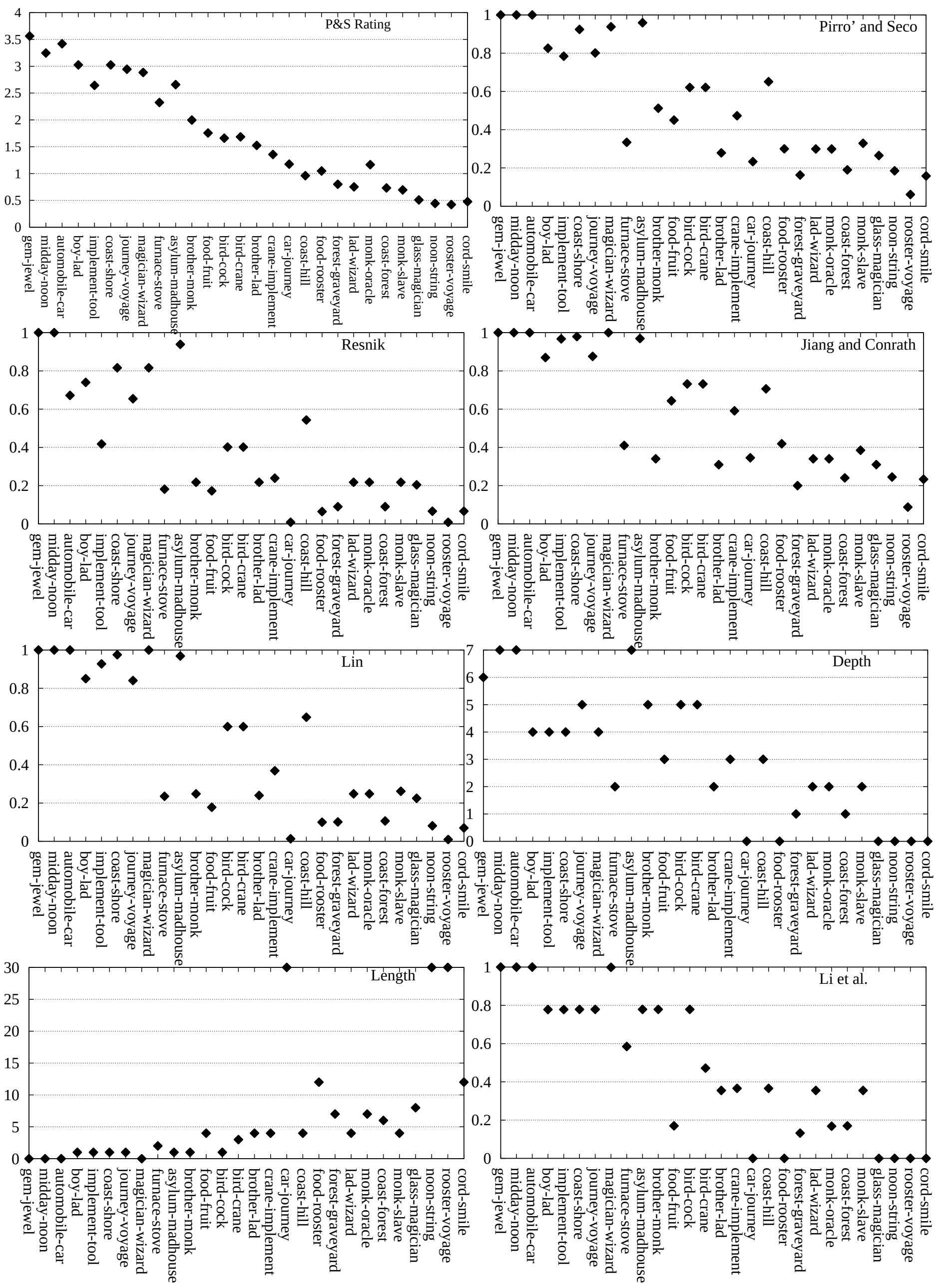


Fig. 7.3. Results and ratings considering the M&C dataset

metric combines an IC formulation of semantic similarity with edge counting. The P&S metric obtained the higher value of correlation on the $S_{M\&C}$ dataset.

On the second dataset, that is $S_{R\&G}$, the correlation values obtained by the different metrics slightly change. Even in this case, the Length metric obtains the poorest correlation. Resnik's metric obtained a correlation comparable to that obtained by the J&C metric. The Lin metric obtained better results. The Li metric, evaluated by considering the optimal parameter determined by authors in [100] obtained a better correlation. However, the P&S metric remains the most correlated w.r.t human judgments also in this dataset. Correlation results reported in Tables 7.5 and 7.6 show that the presence of non native speakers barely affects the values of correlation of the different metrics.

7.4.2 Commonalities and Differences among Metrics

In order to have a deeper insight into the structure of the different metrics, we represent their results as shown in Fig. 7.3. Here, it can be recognized the different nature of edge-counting (i.e., Length, Depth), IC-based and Li's multi-source metrics. In particular, edge- counting metrics give discrete results as output (i.e., integer values). For the Length metric, a lower value of path length corresponds to a higher similarity value between words. For instance the first three pairs (i.e., *gem-jewel*, *midday-noon* and *automobile-car*) have a length equal to zero which is due to the fact that these word pairs belong to the same WordNet synset respectively. On the other side, word pairs as *noon-string* and *rooster-voyage* have a relatively high distance which means that the words in the two pairs are not similar. A potential anomaly could be represented by the pair *car-journey* which gets a length of 30, the maximum value. The two words, even if generally related as a car can be the means to do a journey, are not considered similar. That is because similarity is a special case of relatedness and only considers the relations of hypernymy/hyponymy defined in WordNet which is exactly what the Lenght metric does. For the Depth metric, a number of "similarity levels" can be recognized (in Fig. 7.3 for instance, it can be noted that there are 3 ratings in the level 7, 5 in the level 2 and 6 in the level 0). This metric, differently from that of Resnik takes into account the depth of the *msca* thus allowing more specific concepts to be generally judged more similar than more abstract one. Note that this metric obtained a correlation about 30% better than the Length metric.

A more interesting discussion can be done for the IC based metrics. In particular, the Resnik and Lin metrics present two similar regions, one in the center identified by the pairs *bird-cock* and *bird-crane* (translated by 0.2) and the other comprising all the pairs from *car-journey* to *cord-smile*. Note that when the two words to be compared are leaves, according to the intrinsic IC formulation described in equation 7.10, they have IC equals to 1 and therefore equation 7.5 turns into equation 7.2. A similar condition holds for the transformed Jiang and Conrath metric in equation 7.4. The P&S metric when c_1 and c_2 are leaves gives as result $sim_{P\&S}(c_1, c_2) = 3 \cdot IC(msca) - 2$. In this case, if the *msca* is high in the taxonomy (it receives a low IC) the metric returns a lower similarity value than when the *msca* is low. A similar area can be recognized between the J&C and P&S metric (i.e., from the pairs *forest-graveyard* to *cord-smile*). In this area generally, the J&C obtains higher similarity scores. However, according to the original intent of R&G to chose word pairs from very similar to less similar, the P&S metric seems to better respect this trend in this case. Finally, the Li metric has a very similar region (comprising the pairs from *car-journey* to *cord-smile*) to the Depth metric . The word pairs in this region are rated equally due to the fact that the Li metric exploits the value of Depth and when this is zero, according to equation 7.5 the similarity value returned by the Li metric is zero.

In summary, the results of these experiments demonstrate that our intuition to consider the original formulation of IC provided by Resnik, to some extent, a special case of the formulation given by Tversky is consistent. Moreover, the metric (i.e., Li) that obtained results comparable to the P&S metric has been empirically designed and relies on two parameters to be adjusted.

7.4.3 Some Considerations on the P&S Metric

By scrutinizing equation 7.9 that defines the P&S metric a couple of observations arise. The first is related to the intrinsic IC formulation: if the number of hyponyms of a concept changes, the similarity between a pair of concepts will change as stated in equation 7.10. Note that this formulation of IC takes into account the strengthens of links: links (hypernym/hyponym) higher in the ontology are not as strong as those closer to the bottom. Moreover, links that leaf nodes have with their immediate hypernym are the strongest (have the smallest semantic leap between them). Therefore, if we add a hyponym to a concept we are weakening the relation between the concept and its immediate hypernym hence weakening the relation between the pair being compared. Here, the underlying assumption is that the ontology is organized in a ”meaningful and principled way”, and if there is need to reorganize the ontology then we should accept that similarity values change. The second consideration is related to the branch of equation 7.9 $sim_{P\&S}(c_1, c_1) = 1$. We added this branch to solve the problem of the Resnik metric i.e., $sim_{P\&S}(c_1, c_1) \neq 1$. Moreover, our evaluation show that such a function yields results that correlate better with human judgments.

7.4.4 Impact of Intrinsic Information Content

In this section we evaluate the impact of the intrinsic IC formulation on the IC metrics. Fig. 7.4 shows the results of this evaluation. For sake of space we do not report the scores obtained by considering the two IC formulations. As can be noted, the correlation is improved for each metric. In particular, a notable improvement is reached by the J&C (about 40%) and P&S metrics (about 15%). In the light of these results we can conclude that the intrinsic IC formulation is an effective and convenient way to compute IC values.

7.4.5 New Challenges for Researchers

The results obtained by some metrics in our experiments are very close to human judgments. At this point a question arises: how much we can expect from a com-

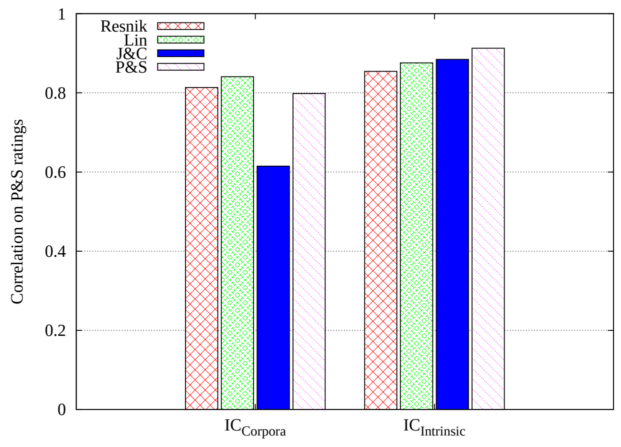


Fig. 7.4. Impact of the Intrinsic IC formulation

putational method for assessing semantic similarity? Resnik in [142] took into account the correlation between experiments in order to obtain a possible upper bound. Resnik obtained a value of correlation w.r.t M&C experiment of 0.9583 while the inter-annotator agreement obtained was 0.9015. This latter result has been considered for many years as the theoretical upper bound. However, we agree with what was observed in [100] and propose to consider as upper bound not the inter-annotator agreement but the correlation between the ratings of the different experiments. This is because semantic similarity should be considered as a collective property of groups of peoples (i.e., all the participant to the experiment) rather than considering them individually as done by Resnik with the inter-subject agreement. Moreover, since we replicated the R&G experiment on all the 65 word pairs dataset we can correlate our results with those obtained by R&G. Hence, we propose to set as new hypothetical upper bound the value of correlation between the R&G and P&S ratings, that is, 0.972. This latter consideration provides new challenges for researches. In fact, even if the metric we presented obtains a correlation value of 0.908 using this dataset, this value is far from the new hypothetical upper bound.

7.5 Concluding Remarks

This chapter presented a new semantic similarity metric combining features with intrinsic information content. This metric has been shown to be the most correlated w.r.t human judgments. In particular, to evaluate our metric we ran an online experiment which per se is a contribution as the number of participants has been higher than that of previous experiments. Moreover, we deeply compared the nature of this metric with other metrics to investigate commonalities and differences.

The motivation of the design of this can be found in the fact that semantic similarity is important in many research areas ranging from ontology mapping to semantic query routing. In particular, as practical outcome of semantic similarity, we have shown its application in the *lexical matcher* presented in Chapter 6.1. We will discuss another application of semantic similarity, in the context of semantic-based service discovery, in Chapter 11.

Part IV

Case Studies and Applications

8

A Framework for Distributed Organizational Knowledge Management

The following three chapters will show the application of Semantic Web technologies and in particular ontologies within the project KMS-Plus. After a short general description of the project, one of the two contribution, that is, an organizational knowledge management architecture based on ontologies and Peer-to-Peer (P2P) will be presented. The other contribution will be introduce in the next chapter.

8.1 The KMS-Plus Project

The KMS-Plus (KMS+) project is an Italian ”pre-competitive” project financed by the Italian Minister of University and Research. The project involved 4 Italian IT companies. In more detail, KMS+ is meant to make explicit and organize organizational knowledge thus providing an integrated support for enterprise activities. Thanks to KMS+ it should be possible, for instance, to share data, documents, information, software, create work groups and so forth. All this will improve organizational productive processes and allow to manage knowledge in an integrated manner thus paving the way to its efficient reuse. KMS+ exploits ontologies to model organizational processes and knowledge related to the various domains interesting for an organization. This way it is possible to create links between Knowledge Objects (KOs) and ontology concepts. Having a formal representation of organizational knowledge will bring some advantages such as the improvement of the quality of query answering or the possibility to browse organizational knowledge in an efficient way. An interesting aspect of this project, which has been at the basis of this work, is the investigation of *Distributed Knowledge Management*.

8.2 Background and Motivation

Today’s more and more competitive business ecosystem requires Individual Knowledge Workers (IKWs) to learn from other people’s knowledge, to keep current and innovate. Knowledge is a critical factor for business activities supporting organizational strategies [88]. Therefore, managing knowledge becomes central for sustaining organizational core competency. Knowledge Management (KM) is defined as the formal management of knowledge to facilitate its creation, access and reuse by using ad-hoc technologies [127]. The creation of new knowledge occurs through a spiral process of interaction between *tacit* and *explicit* knowledge [121]. The combination of these two categories of knowledge makes it possible to conceptualize four patters: *externalization* (tacit to explicit), *combination* (explicit to explicit), *internalization* (explicit to tacit) and *socialization* (tacit to tacit) referred to as the SECI model. Under the umbrella of KM there are various sub-activities such as knowledge capturing, sharing, generation and retrieval [3]. In order to keep competitive and innovate, organizations need to provide the adequate technological support to KM activities. Early KM systems, based on the ”one size fits all” principle, adopted centralized technological architectures in which corporate knowledge is made explicit, collected, represented and organized following a uniform and superimposed schema. According to this vision, knowledge can be standardized, centralized, and controlled through a linear process that ”cleans” diversity [14, 104].

In the last years [59, 64] knowledge is being considered as the result of different perspectives and social interactions between interpretations belonging to both individuals and groups. Therefore, in this new setting, subjectivity and sociality have to become part of the knowledge creation process. These new requirements are not properly fulfilled in centralized KM architectures. Therefore, Distributed Knowledge Management (DKM) has been proposed as a new vision for KM. DKM is based on the principle that different perspectives within complex organizations should not be viewed as an obstacle to knowledge exploitation, but rather as an opportunity to foster innovation and creativity. The two core principles of DKM are [4]:

- *Autonomy*: IKWs should be granted the highest possible degree of semantic autonomy to manage their local knowledge.
- *Coordination*: the collaboration between autonomous entities is achieved through a process of semantic coordination rather than through a process of semantic homogenization.

Hence, it is crucial for organizations to support the creation of communities of workers in which knowledge can be created, organized and shared. Communities of Practice (CoP) are ”places” where knowledge can be created and exchanged [96]. A CoP includes people sharing goals and interests that collectively reflect on a problem or an idea. In a CoP, individuals can produce and learn new concepts and best practices, thus allowing the community to innovate and create new knowledge. IKWs within communities access and share knowledge interacting through synchronous (e.g., instant messaging and collaborative editing environments) and asynchronous (e.g., e-mail applications) tools. However, today an increasing number of people work outside of the traditional office for many hours a day. Current technologies do not properly support this new style of work, so it is becoming increasingly hard to exchange information in a labyrinth of network connections, firewalls, file systems, applications, etc. IKWs spend much of their time to adapt to their ever changing work environment, and limited time is left to actual productive work. What is needed is a flexible work to support the ubiquity of the IKW and enable cooperative work. The *virtual office* approach can comply with this requirement. A virtual office fulfills the roles of the traditional, centralized office although the employees collaborate for the most part electronically with sporadic physical contacts. This model is becoming more and more essential since, even in conventional offices, today many business relations are necessarily maintained across distributed environments. For instance, customers and suppliers are located at different sites, project co-workers are often located in different departments, and a CEO's speech may be listened remotely [27]. Overall, technological supports to foster the creation of CoPs and support collaborative work and cooperation become central for the success of DKM systems.

8.2.1 The Role of Peer-to-Peer Computing

The P2P paradigm naturally supports the creation of communities (e.g., workspaces, peer groups) in which content and conveyed knowledge can be created, shared, exchanged and transformed through synchronous and asynchronous collaboration. The impact of P2P computing in KM application has been thoroughly investigated in [5, 61]. In particular, emerged that in the new economy collaboration between IKWs is moving from intra organization to inter organization, that is, across organizational boundaries. Since centralized KM systems have a corporate-based infrastructure and/or proprietary networks to operate, in these environments it is infeasible to support ad-hoc and volatile collaborations. Conversely, the P2P naturally supports volatile cooperation as peers can leave and join the network at any time. Additionally peer groups, bringing together different people from different organizations, can be formed and dissolved dynamically. Hence, collaboration between IKWs (i.e., peers) can be ”naturally” extended across organizational boundaries without relying on any corporate infrastructure.

8.2.2 The Role of Semantic Technologies

Toward the design of a comprehensive DKM system to met the requirements envisioned by the KMS+ project, the use of semantic technologies becomes crucial. In particular, the use of ontologies for KM purposes recently received attention. As semantic technologies are proving their value with targeted applications, there are increasing opportunities to consider their application in KM as a support to increase organizational performance by better exploiting intellectual assets [37]. There are recent examples of applications of semantics to empower knowledge management or better support knowledge services [48, 106]. In particular, ontologies are used by communities to establish conceptual models that enable to share a precise meaning of symbols exchanged during communications [105]. We argue that KM applications can benefit from ontologies to precisely define the meaning of various symbols at organizational level (e.g., basic organizational assets and interests), community level (e.g., to model a particular aspect of the organizational knowledge domain) and individual level (e.g., to create personal perspective about a knowledge domain).

8.2.3 Combining Peer-to-Peer and Semantic Technologies

The combination of P2P (to support DKM and the virtual office model) and ontologies (to harness semantics of knowledge) allows comprehensive Distributed Ontology-based Knowledge Management Systems (DOKMS) to be built. In this chapter we cover some issues related to the design of a real-world DOKMS with special emphasis on the requirements to enable semantics driven KM as envisioned by the KMS+ project. In particular, we faced the following issues: (i) imposing a single ontology on the enterprise is difficult (if not impossible); (ii) in a DOKMS, adequate supports to enable ontology building and evolution are required; (iii) ontologies can be fruitfully exploited to perform semantics-driven content retrieval.

To cope with the first issue, our DOKMS model provides an ad-hoc ontology framework supporting different levels of knowledge management. An Upper Ontology is exploited to establish a common organizational knowledge background. A set of Workspace Ontologies are designed to manage and search knowledge within workspaces (e.g., communities) by the establishment of a contextual (i.e., related to the aim of a group) understanding. Finally, according to the autonomy principle of the DKM, Personal Ontologies support IKWs in Personal Knowledge Management [169] activities. To cope with the second issue, a mechanism based on distributed voting, enabling to build and update ontology in a democratic way is provided. This way a tradeoff between complete centralization, that does not fit with the DKM theories, and complete decentralization, that raises problems of ontology matching [56], can be achieved. Finally, semantics-driven content retrieval is achieved by exploiting a mechanism through which content can be ”annotated” to ontology concepts and subsequently retrieved by specifying ontology concepts instead of simple keywords.

The remainder of this chapter is organized as follows. Section 8.3 describes a generic architecture exploitable to design DOKMS. Section 8.4 describes the layered ontology framework supporting DOKMS. In this section the mechanism to handle ontology drift and perform semantic-based knowledge retrieval will be presented.

8.3 A Generic Architecture for Designing DOKMS

Fig. 8.1 shows the abstract architecture of the framework. The architecture is based on five layers including basic communication services, data handling services, semantic services, and workspace management services. At the higher level there are a set of tools allowing IKWs to do actual work. The layers of this architecture are briefly described in the following.

8.3.1 Core Layer

This layer defines the core services whose implementation can be based on any P2P infrastructure. The main services provided by the Core Layer, which are exploited by higher layers, are: the K-Group Service which allows to create new K-Groups (e.g., communities or workspaces); the Connection Service which allows IKWs to join

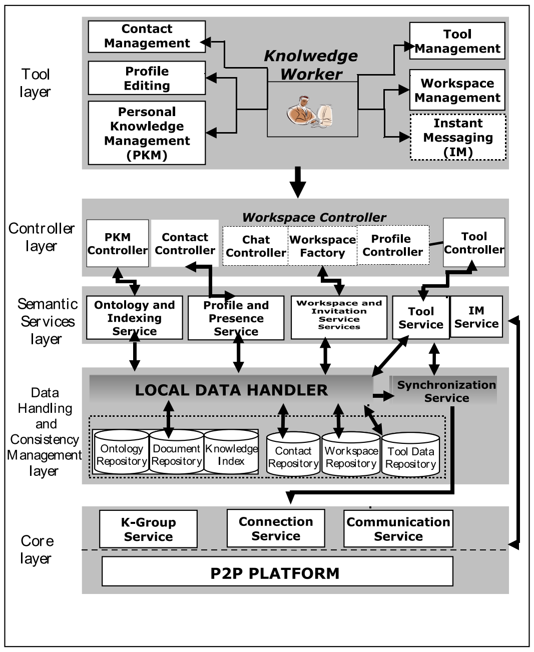


Fig. 8.1. A generic framework for designing DOKMS

P2P network, and the Communication Service that provides features used to send and receive messages.

8.3.2 Data Handling and Consistency Management Layer

To favor the autonomy of users, this framework enables to create different replicas of the same object, so that users can work on their local copies. Since several clients can concurrently work on shared objects, this raises the problem of maintaining data consistency ([27, 177]). Each IKW can perform read operations, or provisional write operations, directly on its local copy of the object, through the primitives provided by the Local Data Handler. The purpose of this layer is to ensure data persistence, consistency management and synchronization of shared objects. More details on the techniques adopted are given in Chapter 9. Finally, the Local Data Handler manages a set of local repositories to store information about contacts, workspaces and knowledge objects.

8.3.3 Semantic Services Layer

The Ontology and Indexing service deals with operations involving ontologies (creation, update). Moreover, through this service documents can be indexed for keyword based search. The Profile and Presence Service manages status check operations and enables users to create and publish their profiles within the network. Within these profiles peers can advertise their expertise on the form of a set of ontology concepts. The Workspace and Invitation Service handles the set up of workspaces and their population which is performed by sending invitation messages to peers. The Tool Service is used to add new tool instances to workspaces at run time. The Instant Message Service allows peers to communicate each other via a chat system.

8.3.4 Controller Layer

This layer contains a set of controllers that catch users operations and forward them to the underlying layers. The Workspace Controller manages workspace settings through the creation of workspace profiles that contain information about workspace topics and about the set of tools and the IKWs that are included in the workspace. The Contact Controller enables peer to discover other peers over the network and add them to a personal Contact List. The PKM Controller is delegated to manage IKWs Personal Knowledge. The Tool Controller is responsible for allowing users to handle operations (add, update, remove) on tools.

8.3.5 Tool Layer

This layer provides a basic set of tools (document sharing, shared calendar, shared address book, shared sketch pad, shared browser) that can be used within workspaces. In addition, other tools can be developed and included in the system as modular components.

The described framework can be implemented by any underlying P2P architecture, however, as Sun's JXTA [167] is widely accepted as the *de facto* standard P2P framework we used it. The implementation of this framework within the the K-link+ will be described in Chapter 10.

8.4 An Ontology Framework Supporting DOKMS

An ontology [68, 69] is an abstract representation of a knowledge domain which allows its modeling in terms of concepts, relations between concepts, class hierarchies and properties, and permits reasoning about the represented knowledge. Ontologies also offer a way for defining a set of possible instances of concepts and relations, thus providing links between the model and the modeled reality. In the latest years the knowledge management community has been considering ontologies as an adequate support for managing the semantics of information [60]. Next generation knowledge management systems will probably rely on conceptual models that go beyond classical ER models. They will exploit ontologies for defining a precise semantic meaning of a shared terminology. Recently some knowledge management systems based on ontologies have been proposed. The FRODO system [1] exploits ontologies as a mean for knowledge description in organizational memory. Comma [62] combines agent technologies for enabling ontology-based knowledge management systems. Also the problem of building ontology-based systems has been recently investigated in [159].

Here we present an ontology framework focused on distributed knowledge management in organizations. In designing this framework we faced three critical issues. First, in an organization it is not arguable to have a single and universally accepted ontology. It is preferable to provide a multilayer ontology support that allows to: (i) define a quite-static part of organizational knowledge that should be accepted by everyone; (ii) cover specific aspects of the knowledge domain faced by the organization (e.g., in an organizational commitment) that will be deepened when necessary. In such a way, the organizational background can incrementally grow up. This aspect will be detailed in Section 8.4.1. Second, ontologies in an organization need to evolve continuously [123]. This problem becomes more challenging in a distributed scenario where there are no central entities that handle ontology management operations. This aspect will be detailed in Section 8.4.2. Third, a large body of information in an organization typically exists outside the knowledge base (e.g., emails, textual documents, databases). In order to reuse this amount of information appropriate wrappers have to be provided. These should convert information into an ontological format at an affordable cost. However, this is this is not an easy task; thus it is necessary to provide a different mechanism allowing to create a fine grained layer of metadata based on ontologies [105]. This aspect will be detailed in Section 8.4.3.

8.4.1 Harnessing Organizational Knowledge through Ontologies

In order to manage semantics of information in our DKOMS architecture, we designed an ontology framework organized in two layers. This framework is shown in Fig. 8.2 with concepts represented as circles and relations as dashed lines.

First Layer: the Organizational Knowledge Background

The first layer (i.e., organization layer) contains an Upper Ontology (*UO*) and a set of Core Organizational Knowledge Entities (COKEs) represented as ontology classes. Ontologies contained in this layer aim at modeling the basic knowledge background of an organization. In particular, the *UO* represents a basic set of meta-concepts relevant for an organization, typically defined by domain experts. More formally an *UO* can be defined as:

$$UO = \langle C, \mathcal{P}, H^c, H^p, A, I \rangle$$

consisting of a set of concepts C and a set of properties $\mathcal{P}$ respectively arranged in the hierarchies H^c and H^p that associates each concept c_i with its sub-concepts $Sub(c_i)$ and each property p_i with its sub-properties $Sub(p_i)$. A is a set of axioms. I represents the extensional part of the ontology and contains instances of concepts and properties. This definition comply with the features of OWL [1] and RDF(S) [2]

[1] http://www.w3.org/2004/OWL

[2] http://www.w3.org/RDF

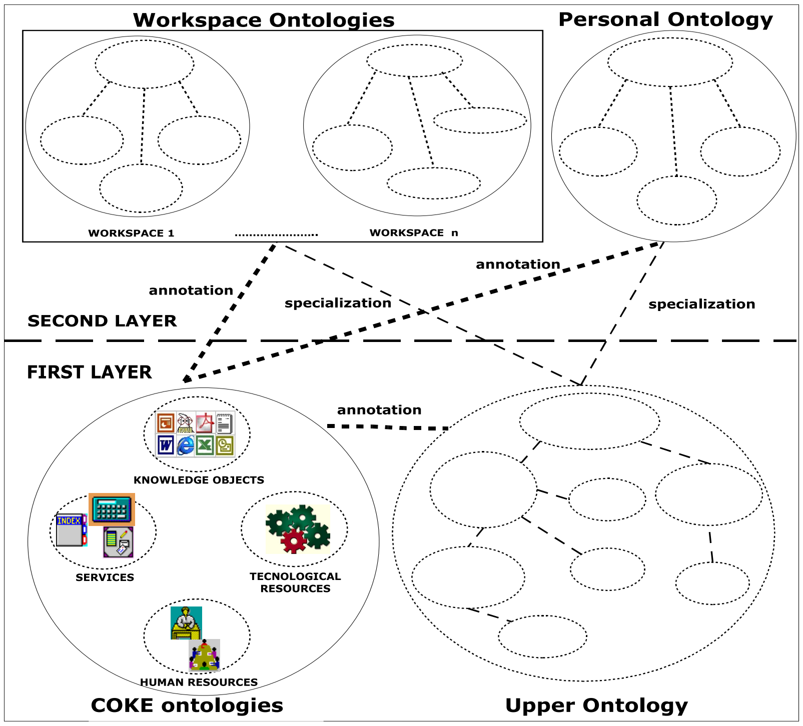


Fig. 8.2. The user view of the Ontology Framework

ontology languages which provide constructs such as *owl:Class* and *rdfs:subClassOf* to define classes and their hierarchy H^c and *rdfs:Property* and *rdfs:subPropertyOf* to define properties and their hierarchy H^p.

The *UO* can be viewed as a semantic network of concepts similar to a thesaurus. For instance, the *UO* for a health care organization will contain concepts and relations related to diseases, clinical practices, drugs, surgery, etc. COKEs aim at giving a semantic description of well-known organizational sources of knowledge. We identified four COKEs:

- The *Human Resource* COKE describes organizational groups (Community of Practices, Project Teams) and individuals. For each IKW, personal data, skills, group memberships and topics of interest are represented. A group is described through its objectives and topics and contains information about the participant IKWs.
- The *Knowledge Object* COKE describes textual documents, database elements, emails, Web pages, through common metadata (e.g., data of creation, document type, author, URI). In particular, this COKE supports ontology-based content retrieval.

- The *Technological Resource* COKE describes tools through which knowledge objects are created, acquired, stored and retrieved. For each tool, this kind of COKE provides information about version and features.
- The *Service* COKE describes services, provided by IKWs, in terms of provided features and access modalities. Example of services can be Web services or P2P services such as JXTA services [167].

Each COKE has its own definition also in terms of attributes. For instance, the COKE *Knowledge Object* (KO), which describes different types of unstructured textual documents, contains attributes such as *name*, *size*, and *author*. Instances of the same COKE share the same structure, so allowing for the management of implicit and explicit knowledge stored in structured, semi-structured or unstructured formats. As shown in Fig. 8.2, *annotation* relations can be defined between the COKEs and the *UO*. That means that COKE instances can be semantically associated to the concepts of the *UO* by following the principle of superimposed information, i.e., data or metadata ”placed over” existing information sources [107]. For instance, let us consider a human resource skilled in Java. An annotation relation can associate the corresponding COKE human resource instance to the Java concept contained in an *UO*. This annotation can be exploited when searching for human resources skilled or interested in Java, for instance, if a group must be created to carry out a particular commitment related to Java programming.

Second Layer: Extending the Organizational Knowledge Background

The second layer of the ontology framework (shown in Fig. 8.2) is composed of a set of *UO* extensions called Workspace Ontologies (WOs), and one or more Personal Ontologies for each IKW. A *Personal Ontology* (PO) is the specialization of one or more *UO* concepts and is used to deepen a particular aspect of the knowledge domain in which an IKW is interested. More formally, a *PO* can be defined as follows:

$$PO = \langle UO, UOC', UOP' \rangle$$

where the *UO* is the Upper Ontology and UOC' and UOP' are the sets of new concepts and properties added by the IKW. A *PO* operates at individual level as semantic support for personal knowledge management operations. It is defined by IKWs that use the *UO* and need to extend it for their specific goals in the organizational activities. In order to enhance social aspects of knowledge management, the framework also allows to create *WOs*. A *WO* specializes one or more *UO* concepts and is used to support cooperative work in a workspace. Even in this case, IKWs can annotate COKEs instances relevant to the workspace to *WO* concepts and retrieve them by semantic search. More formally, a *WO* can be defined as follows:

$$WO = \langle PO, WT \rangle$$

where *PO* has the same structure as the *PO* and *WT* is a set of concepts about workspace topics, on which an agreement among workspace members has been reached. The relations existing between the *UO* and the *WO* and *PO* ontologies are

”specialization” relations, since such ontologies specialize one or more *UO* concepts. In an organization it is not feasible to have a completely predefined modeling of organizational knowledge through ontologies. Therefore, we designed a distributed voting mechanism that enables ontologies to evolve in a collaborative and democratic way. The next section provides an overview of this mechanism.

8.4.2 Handling Ontology Drift

Although the structure of the defined ontology framework has been designed beforehand, static or fully predefined ontologies in a dynamic distributed environment cannot satisfy the ever-changing requirements of an organization. In our framework, IKWs are allowed to propose extensions or modifications of ontologies (i.e., the *UO* and *WO*) according to their needs. Upon acceptance of such proposals, ontologies evolve in a collaborative and emerging way. Ontology drift, i.e., the evolution of an ontology, is managed through a distributed voting mechanism [63]. In particular for each voting procedure, a voting chair is in charge of permitting or denying the voting process, collecting results and propagating them to participants. Before initiating a new voting procedure, an IKW obtains the authorization from the chair if there are no other voting procedures in progress. An update proposal related to the *UO* is accepted if, within a specified amount of time, the majority of all peers members, regardless of their workspace memberships, agree with the proposal. Similarly, to be approved, an update proposal related to a *WO* needs to be accepted by the majority of the workspace members. A voting process is divided into three phases:

1. *Set up phase*: in this phase the voting initiator contacts the voting chair which, if there are no pending voting procedures, and forwards a ”request for vote” message to all the involved IKWs. This message contains information about the update proposal along with the voting deadline.
2. *Voting phase*: IKWs vote to confirm or reject the ontology update proposal, and send their vote to the chair.
3. *Scrutiny phase*: when the deadline expires, the chair counts up the votes and sends the result to the involved IKWs. If the update proposal has been accepted, the *UO* or *WO* is modified accordingly.

When IKWs, which were previously offline, reconnect while a voting procedure is in progress, they are made aware of the voting proposal by the voting chair and can join the voting process. If they reconnect when the voting procedure has terminated, they receive from the chair a notification containing information about the updated version of the ontology.

8.4.3 Ontology Based Information Retrieval

As stated in Section 8.4.1, ontologies can be exploited to annotate COKE instances to concepts. These annotations are supposed to reflect the content of a particular instance and establish the foundation for its retrieval when requested. In general, semantics-driven information retrieval can be performed using specific tools able to retrieve specific kinds of COKE instances. In particular, we implemented the K-link+ File Sharing tool, a tool trough which instances of Knowledge Objects (KOs) can be retrieved (see Section 10.2.1). While the system allows annotating each kind of COKE instance, here, we describe the retrieval of KOs instances. Through the annotations, unstructured information constituting a KO (e.g., a textual file) can be semantically enriched and its retrieval can be performed by specifying ontology concepts instead of keywords. However, it is expected that the annotation process can be automated to decrease the burden of the IKW. For this purpose, a method based on keyword extraction, as in [136] can be adopted. Keywords extracted from the text of the KO can be viewed as descriptors of the content of the KO. Therefore annotations between such descriptors and ontology concepts can be created. In Fig. 8.3, the portion of the ontology framework exploited for annotating KOs is detailed. As

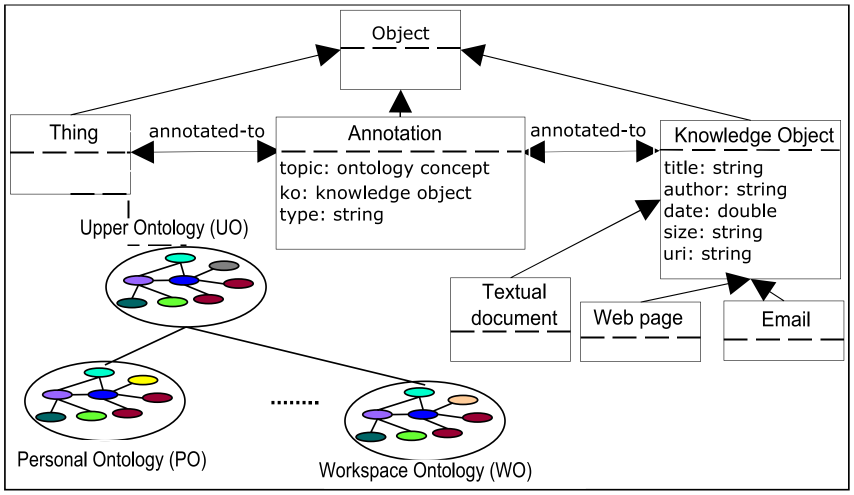


Fig. 8.3. Annotation of knowledge objects

can be noted, the annotation of KO instances to ontology concepts is handled by an *Annotation* class. This class has two properties, *topic* and *KO*, by which concepts and documents are related together. The property type is used to specify the kind of annotation (i.e., manual or automatic). Overall, the process of retrieving a KO can be summarized as follows. The user annotates its own KOs thus creating instances of the *Annotation* class. Instances of the *Annotation* class have a property *topic* which indicates the ontology concept that describes the KO. The user can retrieve KOs by choosing a concept of the ontology and sending a request to the peers.

8.5 Concluding Remarks and Pointers to Next Chapters

This chapter described a generic architecture for designing Distribute and Ontology-Based Knowledge Management Systems. The requirements of this framework have been investigated in the context of the KMS+ project. We presented a generic framework architecture which can be implemented by any P2P platform. Moreover, we described a multi-layered ontology framework allowing to model organizational knowledge and perform ontology-based knowledge retrieval.

A particular aspect introduced in our framework architecture, is the possibility to allow each IKW to work independently on its local copy of a piece of knowledge. This rises the content consistency and peer synchronization problem. In more details this problem concerns how to modify a shared piece of knowledge and keep synchronized while reconnecting after being offline. To address these problems, we devised a consistency model that will be presented in the next chapter. On the other hand, the implementation of the overall framework architecture in the K-link+ system, will be discussed and evaluated in Chapter 10.

9

Content Consistency and Peer Synchronization

In Chapter 8, we discussed a framework that allows to create flexible and collaborative Peer-to-Peer (P2P) applications for knowledge management. In this framework, peers can concurrently work on the same shared documents/files, in the following referred to as ”knowledge objects” or simply ”objects”. To foster peer autonomy, different local replicas of an object can be created, so concurrent access can affect data consistency if adequate mechanisms are not provided. Moreover, peers can join or leave the system at any time, thus introducing the synchronization issue: synchronization is required by peers that reconnect to the network and need to be informed about recent updates made on objects by other peers. Object consistency is also a fundamental reliability requirement for a P2P system. Even if it is not possible, or convenient, to guarantee that all users are provided identical object replicas all the time, mechanisms must be provided to make users work without any actual limitations [140].

This chapter describes an architecture which is designed for the management of knowledge in small/medium enterprises motivated by the KMS-Plus project (see Section 8.1). This architecture adopts a hybrid model to cope with the content consistency and peer synchronization issues and is actually adopted in the K-link+ system that will be presented in the next chapter. The proposed architecture exploits the efficiency of centralized models but at the same time includes decentralized features, which assure scalability properties when the system size increases. This is accomplished by using: (i) a unique and stable server to maintain a limited amount of metadata information about shared objects, (ii) a number of interchangeable servers that maintain and manage the *primary copies* of shared objects, and (iii) a pure decentralized mechanism that allows P2P nodes to effectively exchange up-to-date object replicas. To this aim, different roles can be assumed by K-link+ peers which will be referred to as K-link+ Nodes (KLNs) from now on. In particular, a *Rendezvous* node maintains a common view about shared objects and their state. A set of *Manager*nodes are in charge of receiving object update requests from *Worker* nodes and possibly authorizing them. Finally, *Broker* nodes are used to speed up the propagation of updated objects over the network. A redirection mechanism is exploited to reduce the number of objects handled by overloaded *Managers*. This not only improves load balancing of hosts, but makes the system able to support a larger number of shared objects.

The remainder of this chapter is organized as follows. Section 9.1 describes the hybrid model that addresses data consistency and peer synchronization. Section 9.2 discussed related work. Finally, Section 9.3 concludes. The evaluation of this approach will be discussed in Section 10.3.

9.1 Content Consistency and Peer Synchronization

In our framework, several users can work concurrently on shared objects. To favor the autonomy of users, the system allows different replicas of the same object to be created, so that users can work on their local copies. As mentioned in Section 8.3, the purpose of the D*ata Handling and Consistency Management* layer is to ensure data persistence, consistency management and peer synchronization. To cope with this issue we adopt the sequential consistency model [94], which assures that all updates performed on an object are seen in the same sequence by all the peers. The model is implemented by associating, to each object, a KLN (called *Manager*), which is responsible for authorizing object updates thus allowing the KLNs to view the updates in the same order. In particular, each object is assigned a *Version Numbe*r (VN), which is incremented after each update. In more details, K-link+ defines the following set of roles that can be assumed by workspace nodes:

- *Creator*: it is a KLN that creates a shared object and specifies the Manager List (ML), i.e. the list of KLNs that can assume the Manager role for this object. Managers are ordered on the basis of their responsibilities in managing the object.
- *Rendezvous*: For each workspace, one rendezvous node maintains metadata about all the shared objects in a *Consistency Table* (described below) and provides such information to workspace members. The Rendezvous stores up-to-date information about objects, in particular the identity of the node which is currently in charge of each object (i.e., the Current Manager) and the current VN.
- *Manager*: an object Manager is a KLN that manages the object life cycle and is contacted by KLNs when they want to propose an object update. An object can be assigned to several Managers, but at a given time only the Current Manager, i.e., the first online Manager in the ML, is actually responsible for the object. The Current Manager can decide whether or not to authorize an object update, according to the specific set of semantic rules associated to the object. KLNs are informed about the identity of the Current Manager by the Rendezvous.
- *Broker*: it is a KLN that maintains an updated copy of an object and can forward it to other KLNs. Whereas the Manager role is assigned at object creation time, the Broker role is dynamic, since it can be played by any node whenever it maintains an updated copy of an object.
- *Worker*: it is an ordinary KLN that operates on an object and possibly issues update proposals to the Current Manager. Workers can obtain an updated copy of an object either by a Broker, in a P2P fashion, or by the Current Manager of the object, with a centralized approach.

Table 9.1. An Entry of the Consistency Table

Field	Description
Object ID	A unique ID, that identifies the shared object
Version Number (VN)	Object version number, incremented at each object update
Current Manager	The first online Manager. It is responsible for a shared object
Manager List	An ordered list of nodes that can assume the Current Manager role
Creator	The node that creates the object

The Rendezvous maintains information about the state of the objects in a *Consistency Table*. Each object is permanently associated to an *Entry* of this table, whose structure is shown in Table 9.1. An object is identified by a unique *ID*, which is assigned when the object is created. Moreover, to keep trace of the object state, the Consistency Entry includes a version number *VN* (an integer value), which is incremented at each authorized object update, the *ID* of the Current Manager and the Manager List. While the Rendezvous is in charge of maintaining updated information about all the shared objects of the workspace, KLNs can maintain replicas of the Consistency Entrie describing the objects in which they are interested.

The definition of the mentioned roles enables three different kinds of interactions, as shown in Fig. 9.1. Different kinds of arrows are used to show the different models of interaction among KLNs. In particular, a static centralized approach is adopted when workers interact with the unique Rendezvous of the workspace. The presence of a single Rendezvous is appropriate in a small/medium network, as it is generally possible to assign this role to a node with high reliability features. Note, however, that the load of this node is moderate, as it only deals with small size metadata information, as it will be better discussed in Section 10.3. In fact, the aim of the Rendezvous is to provide reliable and updated information about objects, but the actual management of each single shared object is delegated to the corresponding Current Manager. This enables a dynamic centralized paradigm because the role of Current Manager, if and when needed, can be switched from one Manager to another that is included in the ML of the object. This way, several issues can be tackled: (i) the presence of a central bottleneck, which would be originated if all objects were managed by a single node, is avoided; (ii) it is possible to cope with the volatile nature of P2P networks, in which peers with Manager responsibilities can leave the network at any time; (iii) a Current Manager switch can be performed for an object also to better balance the load among different Managers.

On the other hand, a decentralized approach is exploited by Brokers that provide updated copies of objects to workers in a P2P fashion. The combined use of these three paradigms can represent an efficient trade-off among different ways to face distributed object management. In the following different scenarios, in which the above-mentioned types of interactions occur, are described. These scenarios are: (a) the creation of a new shared object, (b) the update of an existing shared object, (c) the synchronization of a peer and (d) the Manager switch, performed either after a Manager disconnection or to achieve a fairer load balancing of Managers.

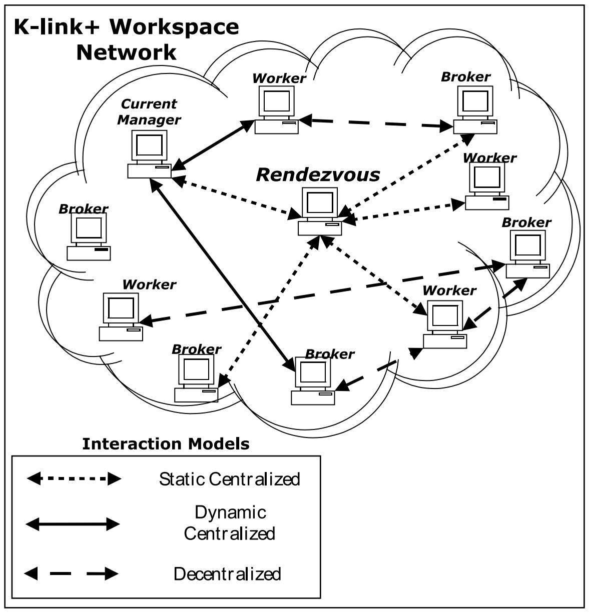


Fig. 9.1. The K-link+ approach to content consistency

Table 9.2. Messages received by the Rendezvous node

Message	Sender	Description
create object	Creator	Inform the Rendezvous about the new object
object information request	Worker	Check the current version number of an object
manager leave	Current Manager	Inform the Rendezvous that the Current Manager is leaving the network
version update	Current Manager	Inform the Rendezvous about the new version number of an object

Table 9.3. Messages received by Current Manager nodes

Message	Sender	Description
online update request	Worker	Propose an object update while online
offline update request	Worker	Propose an object update after reconnecting

Tables 9.3 to 9.5 list the various types of messages used in these scenarios. They are grouped by target node, as this will be useful for the evaluation of the computation load discussed in Section 10.3.

Table 9.4. Messages received by Worker nodes

Message	**Sender**	**Description**
object copy reply	Broker	Send an updated copy of an object
version check	Broker	Ask a worker to check the version number of an object
version information reply	Workspace Network	Inform the worker about the current version number of an object
update reply	Current Manager	Accept/decline update proposals
object information reply	Rendezvous	Give information about the current Manager and current *VN* of an object
object copy request	Worker	Request an updated copy of an object

Table 9.5. Messages received by all workspace nodes

Message	**Sender**	**Description**
object forward	Creator	Forward a copy of a new object along with its metadata to the interested peers
version information request	Worker	Check if an updated copy of an object is available
manager alive	Rendezvous	Inform the network about the identity of a new Current Manager

9.1.1 Creation of a New Shared Object. *Scenario A*

The creation of a new shared object, detailed in the sequence diagram depicted in Fig. 9.2, is performed as follows. After creating a new shared object, a KLN (i.e., the Creator) informs the Rendezvous by sending it a *create object* message which contains metadata describing the new object (i.e., a new Consistency Entry), which will be stored in the Consistency Table. Moreover, the Creator defines the Manager List (ML): the first online Manager specified in the ML automatically assumes the role of Current Manager. The Creator forwards the new Consistency Entry, along with a copy of the new object, to the KLNs that can be interested in this object, by sending *object forward* messages. The KLNs store the received copy of the object in the local object repository through the Local Data Handler, while the Consistency Entry is stored into the local Consistency Table. When a KLN receives a new object it becomes a Broker, since it owns an object whose version number is the same as that maintained by the Rendezvous. A Broker can forward the new object to other KLNs in a P2P fashion, thus making object propagation faster.

9.1.2 Object Update. *Scenario B*

A worker can perform read operations, or provisional write operations, directly on the local copy of an object, through the Local Data Handler. However, every attempt to permanently modify the state of a shared object must be forwarded, through the

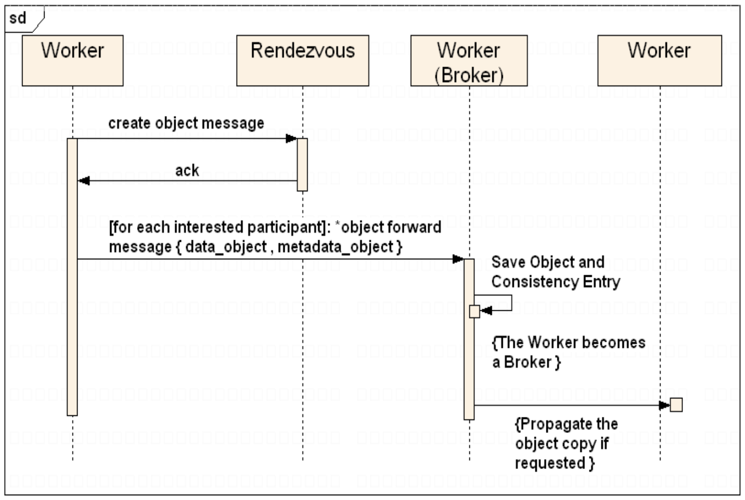

sd
Worker
Rendezvous
Worker (Broker)
Worker
create object message
ack
[for each interested participant]: *object forward message { data_object , metadata_object }
Save Object and Consistency Entry
{The Worker becomes a Broker }
{Propagate the object copy if requested }

Fig. 9.2. Creation of a new shared object

Synchronization Service, to the Current Manager of the object, by sending it an *on-line update request* message. The Current Manager accepts modifications if these do not conflict with the current object state, according to the specific set of semantic rules associated to the object. If a modification is authorized, the Current Manager increments the object VN and sends back an *update reply* message to the requesting worker. Whenever an object update proposal is accepted, the updated copy of the object, along with information about the new VN, is sent from the requester to the involved workspace members, in a P2P fashion, through *object forward* messages, whereas the updated Consistency Entry is sent by the Current Manager to the Rendezvous through a *version update* message. This procedure is described in Fig. 9.3. Note that the propagation of the updated object is initiated by the requester instead of the Current Manager, thus avoiding to overload the latter. The KLNs that receive an updated object copy of an object assume the role of Broker for this object. To foster object propagation, a Broker may contact a set of workers by sending them a *version check* message containing the current object VN. If the worker notes that this VN is higher than that maintained locally, it replies to the Broker with an *object copy request* message and will receive the updated object copy through an *object copy reply* message. If the Current Manager is not available when an update request is issued, a Manager Switch procedure is required, as detailed in Scenario D.

9.1.3 Peer Synchronization. *Scenario C*

A synchronization procedure is performed when a KLN reconnects to the workspace network after being offline. Its purpose is: (i) to provide the reconnecting KLN with updated information about the objects of interest; (ii) to enable the KLN to propose possible object updates made on the local copy while offline. In the first step, the

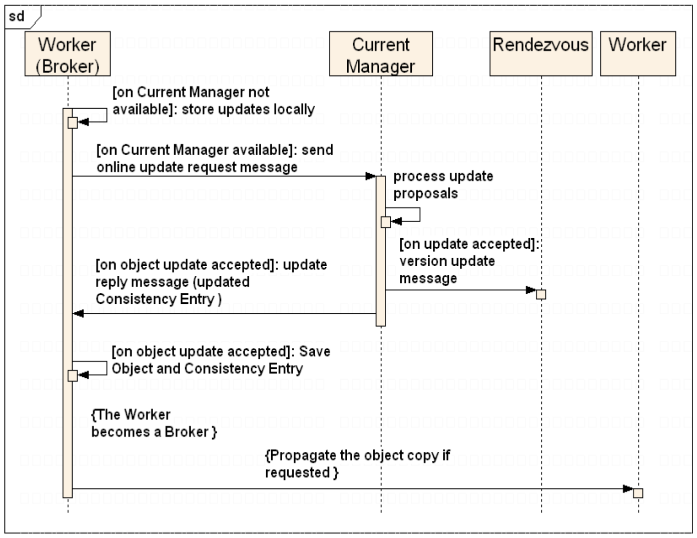

sd
Worker (Broker)
Current Manager
Rendezvous
Worker
[on Current Manager not available]: store updates locally
[on Current Manager available]: send online update request message
process update proposals
[on update accepted]: version update message
[on object update accepted]: update reply message (updated Consistency Entry)
[on object update accepted]: Save Object and Consistency Entry
{The Worker becomes a Broker }
{Propagate the object copy if requested }

Fig. 9.3. Update of a shared object

KLN node uses the Synchronization Service to contact the Rendezvous and get information about current VNs and Current Managers of the objects of interest. This information is obtained by exchanging *object information request/reply* messages. Subsequently, two different procedures are followed by a KLN depending on whether or not it has performed any object update while offline. If no updates have been made, the decentralized approach can be exploited, since the KLN can obtain the latest object version from a workspace Broker. Specifically, the KLN checks whether the object VN received by the Rendezvous is higher than the VN stored locally, which would mean that the object has been updated. In this case, the KLN issues a *version information request* message to the workspace network and receives *version information reply* messages from workspace Brokers. Afterwards, the KLN chooses a Broker from which it can obtain the updated object in a P2P fashion, by using *object copy request/reply* messages. A different procedure is followed if the KLN has made offline updates. In this case, the dynamic centralized approach must be adopted, since the KLN has to submit its update proposals to the Current Manager by sending to it *offline update request* and receiving by it *update reply* messages, following the same procedure described in Scenario B (Object Update). In the case in which the Current Manager is not available when the KLN reconnects, a Manager Switch procedure is required, as detailed in Scenario D. In this case, the KLN keeps its update proposals stored in a local buffer until it is informed by the Rendezvous about the presence of an available Current Manager. In the meantime, the KLN can obtain an updated copy of the object from a Broker. The process is depicted in Fig. 9.4.

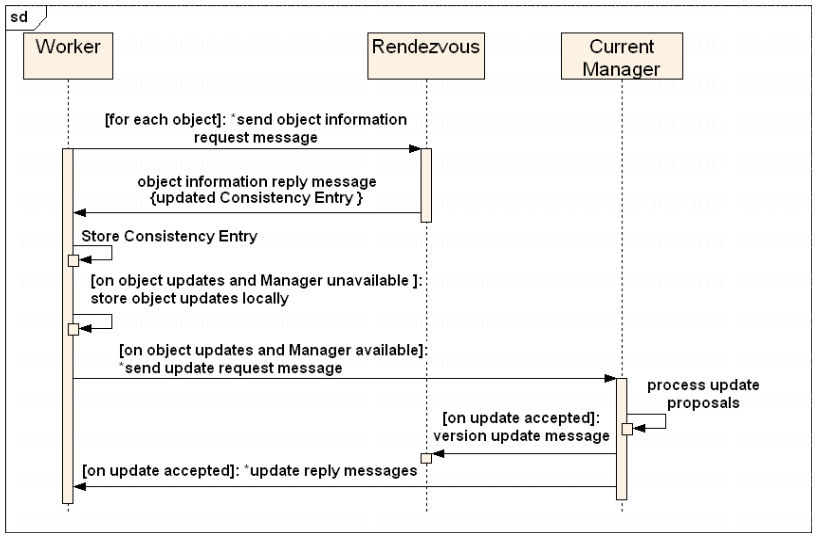

sd
Worker
Rendezvous
Current Manager
[for each object]: *send object information request message
object information reply message {updated Consistency Entry}
Store Consistency Entry
[on object updates and Manager unavailable]: store object updates locally
[on object updates and Manager available]: *send update request message
process update proposals
[on update accepted]: version update message
[on update accepted]: *update reply messages

Fig. 9.4. Object synchronization

9.1.4 Manager Switch. *Scenario D*

In the above-mentioned scenarios, it is assumed that the Current Manager is online and available. If this condition does not hold, a Manager Switch procedure is required. By default the Current Manager is the first online KLN contained in the Manager List. When a new Manager becomes Current Manager, the Rendezvous informs the workspace network through a *manager alive* message. This way workspace members can store information about the new Current Manager (by updating the local Consistency Entry of the object) and will submit to it future update proposals. However, in a P2P scenario, the Current Manager can leave the network at any time either in a safe or unsafe way. In the first case, it sends a *manager leave* message to the Rendezvous. The latter searches for the next online Manager contained in the Manager List and informs the workspace network through a *manager alive* message. If the Current Manager leaves the network abruptly (i.e., without informing the Rendezvous), a different approach is adopted. The Rendezvous is informed about the Current Manager failure directly by a worker. This can happen either when a worker reconnects (Scenario C) or when it receives no reply after an online update request (Scenario B). In both cases, the worker sends an *object information request* message to the Rendezvous. Before responding with an *object information reply* message, the Rendezvous always checks the availability of the Current Manager. If the Current Manager who is in charge of the object has left the system and another Current Manager can be elected, the Rendezvous operates the switch and informs both the requesting worker and the workspace network through a *manager alive* message. A Manager switch can be performed not only due to a peer disconnection, but also to better balance the load carried by different Managers. In this case, a redirection mechanism is exploited: as a Manager experiences a load which exceeds a defined threshold, it asks the Rendezvous to perform a Manager switch for some of the objects that it manages, so that its load can be alleviated. The effect of a Manager switch will be evaluated in Section 10.3.

9.2 Related Work

Replication of content is an important issue in P2P systems, especially if these are devoted to collaborative knowledge management [26, 28, 31, 66]. Replication mechanisms are usually classified into reactive and proactive mechanisms [140]. In reactive replication, as objects are transferred from the home node to the requesting peer, intermediate nodes through which the data flows, determine independently whether or not to cache the content. Some researchers propose to cache pointers instead of real objects in order to yield better query search performance. In DiCAS [175], queries are forwarded to peers of a predefined group which passively cache the pointers in an unstructured P2P network. However, a large overhead is necessary to update the pointers when the object is moved or deleted, since the updated location information has to be flooded to the whole overlay network. In proactive replication, content is pushed to selected peers by the node that stores the primary copy, in order to obtain better performance in terms of query latency, load balance etc. However, the cost of replicating objects to a large number of peers can be cumbersome in both terms of disk space and bandwidth, particularly for systems that support applications with large objects (e.g., audio, video, software distribution). A replication strategy based on object popularity in unstructured P2P networks is explored in [31]. Nevertheless, this strategy does not reduce the worst-case search latency for all the objects. The strategy adopted in this paper borrows characteristics of both reactive and proactive approaches. A push-based mechanism is initiated by a peer when it generates or receives an updated version of an object, since it forwards this object to other workers, in a P2P fashion. This approach assures a quick dissemination of objects to the members of a community but, owing to its decentralized and unstructured nature, cannot guarantee that every worker is given the updated version of every shared object all the time. However, the updated version of an object is always maintained by the related Manager node. Therefore, whenever a worker cannot obtain the updated version of an object through the P2P mechanism, it can always request this object, with a pull modality, to the Manager. An issue strictly related to replication is content consistency, which is, in fact, a fundamental reliability requirement for a P2P system. Current approaches differ according to the scale of P2P systems. In a large-scale and dynamic system, it is complex and cumbersome to guarantee full consistency among replicas, so researchers have designed algorithms to support consistency in a best-effort way. In [35], a hybrid push/pull algorithm is used to propagate updates, where flooding is substituted by rumor spreading to reduce communication overhead. SCOPE [3] is a P2P system that supports consistency among a large number of replicas, at the cost of maintaining a sophisticated data structure. By building a replica-partition-tree (RPT) for each key, SCOPE keeps track of the locations of replicas and then propagates update notifications. Conversely, in a small- or medium-scale system, it is possible to adopt centralized schemes to guarantee a strong consistency model, which is often the sequential model [94]. In [177], an algorithm for file consistency maintenance through virtual servers in unstructured and decentralized P2P systems is proposed. Consistency of each dynamic file is maintained by a virtual server (VS). A file update can only be accepted through the VS to ensure the one-copy serializability. The hybrid architecture described in this chapter, and adopted in the K-link+ system, is specifically designed for knowledge management in small/medium enterprises. Its main purpose is to combine the efficiency of centralized models and the scalability and fault-tolerance characteristics of decentralized systems.

9.3 Conclusions

In this chapter we described a consistency and peer synchronization model for the framework architecture described in Chapter 8. This model is based on a hybrid architecture which involves different kinds of peers. Several usage scenarios have been described. This model has been implemented in the K-link+ system that will be described in the next chapter. In the same chapter a detailed evaluation of the presented consistency model will be discussed.

10 The K-link+ System

The framework architecture motivated by the KMS-Plus project described in Chapters 8 and 9 has been concretely implemented in K-link+. In this chapter we present K-link+ and evaluate it in terms of semantic search and content consistency.

10.1 A Brief Background on JXTA

K-link+ has been implemented by exploiting the Sun's JXTA Peer-to-Peer (P2P) platform. JXTA has been designed keeping in mind three main principles: (i) interoperability; (ii) platform independence and; (iii) ubiquity. Interoperability attempts to give peers a common language to talk to each other. Platform independence is related to the fact that JXTA is platform and language independent. Ubiquity should ensure each device with a digital ”heartbeat” to take part to a JXTA network. To fulfill these ambitious requirements, JXTA's designers proposed a basic set of concepts and protocols. Therefore, we have the notion of peer, peergroup, advertisement, pipe and so forth. Among the others, an advertisement is an XML document that can be exploited to describe and advertise on the network a resource (e.g., peer, peergroup). A pipe is a channel of communication between two endpoints and is used to exchange messages between peers. Concerning protocols, JXTA provides a set of basic protocols to enable resource discovery, communication between peers and so forth. The JXTA architecture is summarized in Fig. 10.1.

10.2 Implementation of K-link+

To implement K-link+ we exploited the JXTA's Java binding. In the current implementation, K-link+ features a basic set of tools to enable Individual Knowledge Workers (IKWs) to cooperate in a virtual office environment. Here we focus on two particular tool which also served as basis to evaluate the system. The first tool is the *File Sharing* tool and the second one is the *Consistency Management* tool.

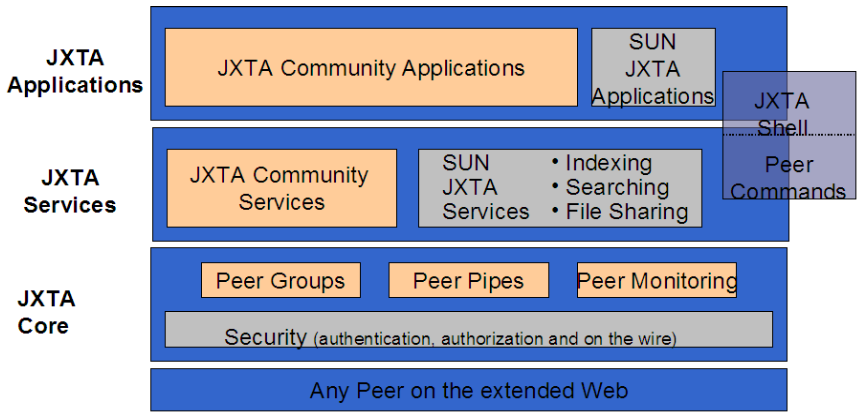


Fig. 10.1. The JXTA architecture

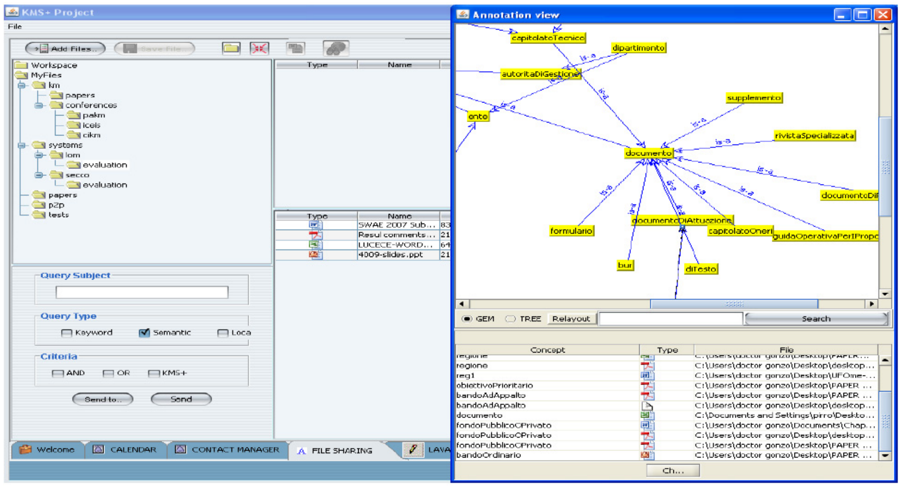
10.2.1 The *File Sharing* Tool

The *File Sharing* tool allows IKWs to annotate documents to ontology concepts and features two kinds of search. The first, based on keywords, exploits local indexes of peer documents created through the Lucene search engine library [1] and allow to search document not only on the basis of their name but also their content. The second kind exploit ontologies. In particular a peer can pick a concept from its own ontology and send the request over the network to receive documents that are relevant to that concept. Fig. 10.2 shows the annotation perspective of the *File Sharing* tool while the search perspective is shown in Fig. 10.3.

10.2.2 Evaluating Semantic Search

In this section we provide a preliminary evaluation of the distributed search mechanism implemented by K-link+. This evaluation has been performed in the real scenario of a small-medium enterprise where a few tens of people work together. The K-link+ system is based on a Super Peer (SP) [182] architecture in which peers of the same group (workspace in our case) are connected to one SP (a Rendezvous in our case), that is, are its clients. Moreover, each SP is aware of a set of other SPs (called neighbors SPs). SPs of different groups can communicate each other in a pure P2P fashion. Each SP is responsible for handling requests (i.e., queries) coming from the peers of its group and forwarding these to all the peers within the group and neighbors SPs. Note that the dimension of the network in which K-link+ has to run is that of a small-medium organizational network in which the number of peers is limited. We evaluated the query response time (QRT) as a function of the peer group size and the query rate, that is, the average rate at which a single peer generates queries. Table 10.1 summarizes the parameters of the evaluation.

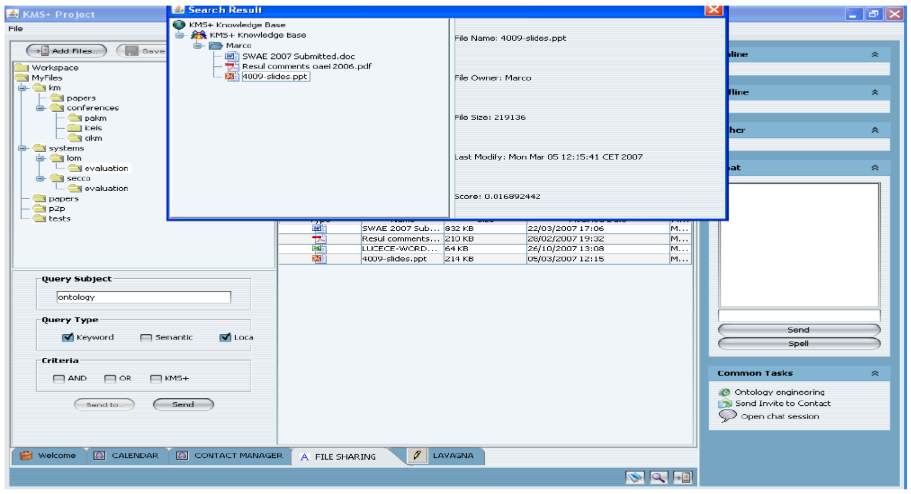
[1] `http://lucene.apache.org`



Type	Name	
	SWAE 2007 Sub...	83
	Resul comments...	21
	LUCECE-WORD...	64
	4009-slides.ppt	21

Concept	Type	File
regione		C:\Users\doctor gonzo\Desktop\PAPER...
regione		C:\Users\doctor gonzo\Desktop\desktop...
reg1		C:\Users\doctor gonzo\Desktop\UFOme-...
obiettivoPrioritario		C:\Users\doctor gonzo\Desktop\PAPER ...
bandoAdAppalto		C:\Users\doctor gonzo\Desktop\PAPER ...
bandoAdAppalto		C:\Users\doctor gonzo\Desktop\desktop...
documento		C:\Documents and Settings\pirro\Deskto...
fondoPubblicoOPrivato		C:\Users\doctor gonzo\Documents\Chap...
fondoPubblicoOPrivato		C:\Users\doctor gonzo\Desktop\desktop...
fondoPubblicoOPrivato		C:\Users\doctor gonzo\Desktop\PAPER ...
bandoOrdinario		C:\Users\doctor gonzo\Desktop\PAPER ...

Fig. 10.2. The annotation GUI



Query Subject: ontology

Query Type: [✓ Keyword] Semantic [✓ Loca]

File Name: 4009-slides.ppt

File Owner: Marco

File Size: 219136

Last Modify: Mon Mar 05 12:15:41 CET 2007

Score: 0.016892442

	SWAE 2007 Sub...	832 KB	22/03/2007 17:06	M...
	Resul comments...	210 KB	28/02/2007 19:32	M...
	LUCECE-WORD...	64 KB	26/10/2007 13:08	M...
	4009-slides.ppt	214 KB	05/03/2007 12:15	M...

Fig. 10.3. The search GUI

Fig. 10.4 shows that, with an increasing query rate, the QRT increases too. This is mainly due to the fact that the larger is the peer group size the larger is the time that a SP needs to forward queries. With a larger peer group size the SP is more inclined to be overloaded for processing queries coming from its peer group. Besides, higher query rates generate more queries thus increasing the QRT. We purposely chose to adopt an unrealistic query rate in order to stress the system and evaluate it in a worst case scenario. In fact it is unreal that a peer generates a query each 6 secs. Note that in a network of 27 peers with each peer generating a query each 6 secs., the QRT

Table 10.1. Parameter values adopted in evaluating semantic search

Parameter	Value
Number of peers	3 to 27
Query rate	1 each 6 secs. to 1 each 24 secs.

increases up to 35 secs. However, even this scenario has been adopted to stress the system. In fact, it is highly improbable that all the peers of a group generate queries at the same time. It is likely that only a few of them do this. Therefore, we decide to

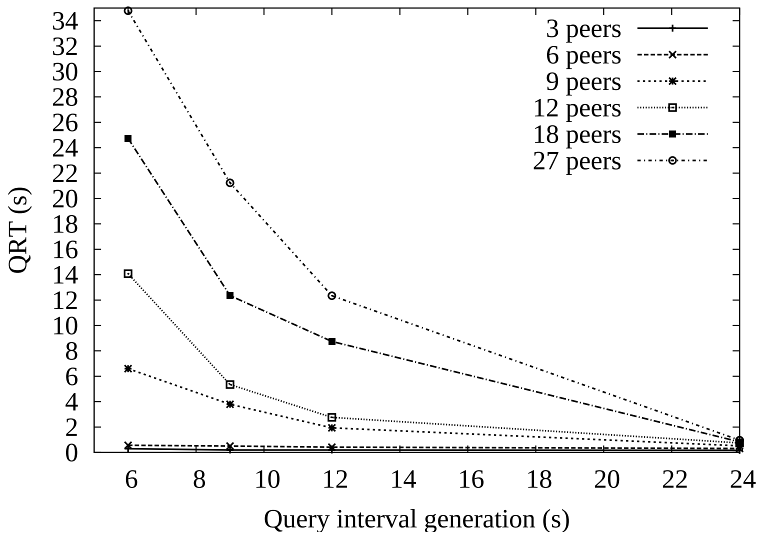


Fig. 10.4. QRT considering all the peers sending queries

further evaluate our system in two more realistic settings: first by considering only 50٪ and then 25٪ of peers as ”active” peers, that is, peers that generate queries. However, in these settings all peers answer to queries. Fig. 10.5 reports results obtained in the case that 50٪ of peers are active peers. In this setting, QRT noticeably decreases. In fact, it never exceeds 7 secs. Note that this is a more realistic setting. In fact in a given peer group, only a subset of peers generates queries simultaneously. As one can imagine, the QRT further decreases if we consider 25٪ of peers as active peers. Fig. 10.6 reports evaluation considering 25٪ of peers active. In this case the QRT never exceeds 5 secs. In Fig. 10.7, the QRT considering 18 and 27 peers is compared by considering different percentages of active peers (i.e., peers sending queries). It can be noted that only if all peers are active the QRT is very high; in real scenarios (less that 50٪of peers active in sending queries concurrently on the network) QRT is

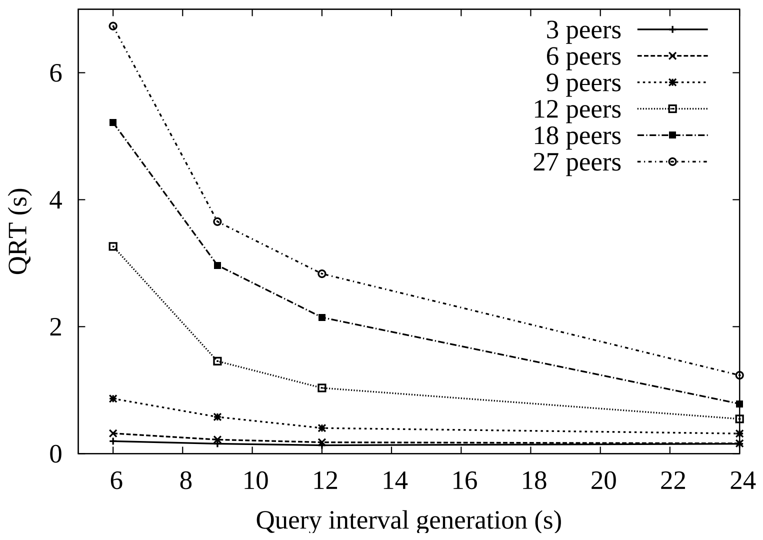


Fig. 10.5. QRT considering 50% of peers sending queries

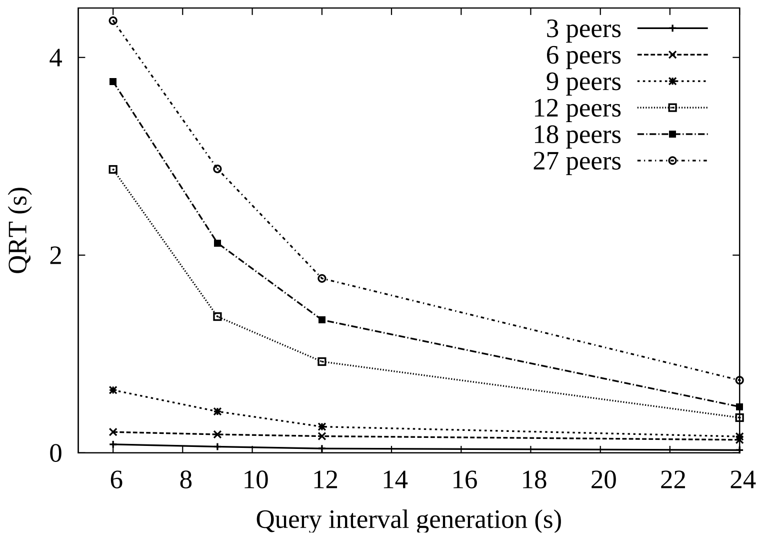


Fig. 10.6. QRT considering 25% of peers sending queries

limited. Overall, we can conclude that our approach performs well when considering a realistic interval of query generation and peers that generate queries.

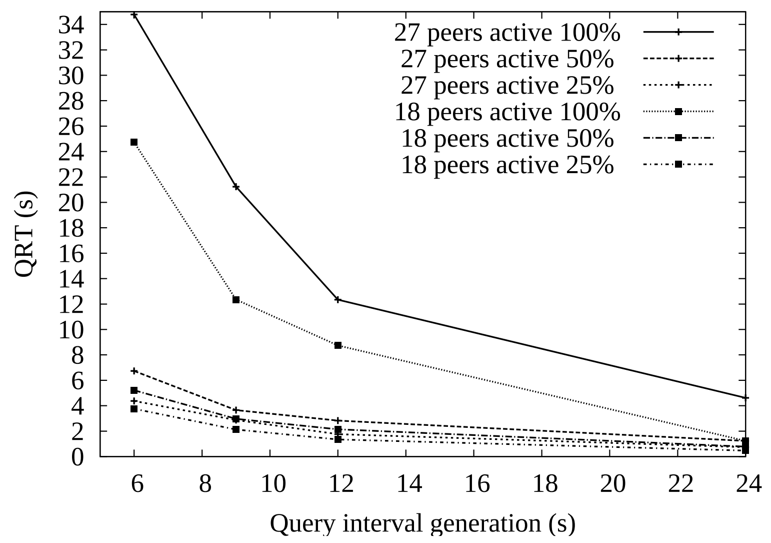


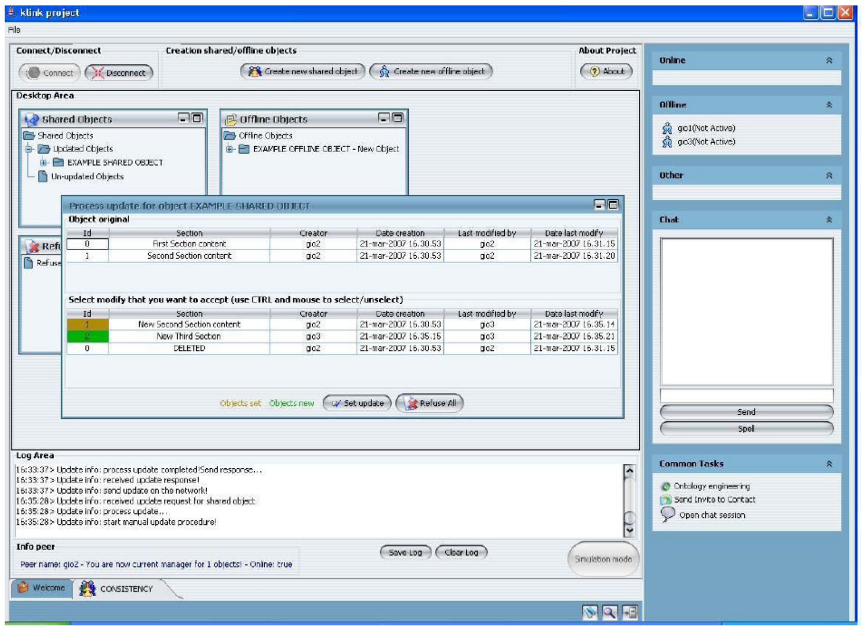
Fig. 10.7. QRT considering a variable number of peers sending queries

10.3 The *Consistency Management* Tool

The current implementation of the *Consistency Management* allows creating textual documents that can be shared by peers. Fig. 10.8 shown the main interface of the tool. Each document is composed by a set of sections described by some metadata (e.g., the section name, the peer that created the section). When a new object is created, the peer that creates it, according to what defined by the consistency model described in Chapter 9, chooses a list of peers that are responsible for the object (i.e., the Manager List). The first online peer contained in the list assumes the role of Current Manager (CM). The peers request update on a shared object directly to the CM. For each shared object the CM can accept or reject the update proposals. For instance, in the central part of Fig. 10.8, some update requests received by the CM are shown. In particular, it has to be decided whether or not to (i) allow modification of the content of object with ID 1 (first row); (ii) add a new section to object with ID 2 (second row); (iii) delete object with ID 0 (third row). When an update on a shared object is performed, an updated copy of the object is sent to the peers that are aware of it.

10.3.1 Evaluation

In this section we present an evaluation of the model for data consistency and peer synchronization described in Chapter 9. The main purpose of our performance analysis is to evaluate the load of Manager and Rendezvous nodes. Analysis is made through a mathematical model based on the queuing theory and often adopted for the performance evaluation of computer systems [81]. Parameters adopted in the evaluation were experimentally determined during the actual operation of the K-link+



Process update for object EXAMPLE SHARED OBJECT

Object original

Id	Section	Creator	Date creation	Last modified by	Date last modify
0	First Section content	gio2	21-mar-2007 16.30.53	gio2	21-mar-2007 16.31.15
1	Second Section content	gio2	21-mar-2007 16.30.53	gio2	21-mar-2007 16.31.20

Select modify that you want to accept (use CTRL and mouse to select/unselect)

Id	Section	Creator	Date creation	Last modified by	Date last modify
1	New Second Section content	gio2	21-mar-2007 16.30.53	gio3	21-mar-2007 16.35.14
2	New Third Section	gio3	21-mar-2007 16.35.15	gio3	21-mar-2007 16.35.21
0	DELETED	gio2	21-mar-2007 16.30.53	gio2	21-mar-2007 16.31.15

Log Area

```
16:33:37> Update info: process update completed!Send response...
16:33:37> Update info: received update response!
16:33:37> Update info: send update on the network!
16:35:28> Update info: received update request for shared object
16:35:28> Update info: process update...
16:35:28> Update info: start manual update procedure!
```

Info peer

Peer name: gio2 - You are now current manager for 1 objects! - Online: true

Fig. 10.8. The Consistency Management tool GUI

platform in our departmental network. In particular, these parameters concern the size and frequency of client requests and the corresponding service times experienced on the Rendezvous and the Managers. The arrival of messages and their processing is modeled through M/G/1 queues [73]. An M/G/1 queue consists of a FIFO buffer with requests arriving randomly according to a Poisson process at rate λ and a processor, called a server, which retrieves requests from the queue and serves them on a first-come-first-serve (FCFS) order, with a generic (G) distribution of service time. In fact, while the assumption of Poisson arrival process is generally considered realistic for Internet traffic, the service time of requests is heavy-tailed in nature [34, 35]. In particular, the task size is often modeled with a Bounded Pareto distribution. According to this distribution, a high percentage of tasks require a short processing time, while a low percentage require long processing time. As opposed to the Pareto distribution, the Bounded Pareto distribution allows for the definition of minimum and maximum task sizes. This prevents the possibility of generating very long or very short tasks, which are not realistic. The probability density function for the Bounded Pareto B(k,p,a) is reported in equation 10.1.

$$f(x) = \frac{\alpha k^{\alpha}}{1-(k/p)^{\alpha}} x^{-\alpha-1}. \qquad (10.1)$$

In this equation, α represents the task size variation, k is the smallest task size and p is the largest task size. This function is defined for $k <= x <= p$ and expresses the distribution probability of the service time. Values of k and p were set according to measurements taken during the actual usage of K-link+ at the GridLab of the University of Calabria and at the ICAR-CNR institute. The parameter α must be included in the range $\langle 0, 2\rangle$ (a lower value accounts for higher variability), and is set to 1 for our analysis. The theory of $M/G/1$ queues enables the calculation of several interesting indices [128], that is, the average load on the server, the average processing time, needed to process a request at the server, the average waiting time of requests in the queue and the overall service time, which is the sum of the waiting time and the processing time at the server. In particular, the average load, ρ, can be calculated as $\frac{\lambda}{\mu}$, where λ is the average frequency of request arrivals at the server and μ is the inverse of the average processing time E(X), which can be calculated as the first moment of the Bounded Pareto service time distribution. The expected waiting time of a request in the queue, E(w), can be obtained by using the Pollaczek-Khintchine (PK) equation and the Little's law [73]. This results in equation 10.2, in which $E(X^2)$ is the second momentum of the Bounded Pareto distribution.

$$E(w) = \frac{\lambda E(X)^2}{2(1-\rho)}. \tag{10.2}$$

The overall service time $E(T)$ is simply obtained by adding to equation 10.2 the average processing time $E(X) = \frac{1}{\mu}$. The service time is only defined in the case that the average load is lower than 1, that is, if λ is lower than μ, otherwise the queue will grow indefinitely. Actually, the average load can be interpreted as the average CPU utilization needed to cope with the incoming messages. A value greater than 1 indicates that the node is overloaded and more servers are necessary to cope with the flow of requests. In the next subsections, the performance of the most critical categories of nodes in the proposed content consistency and peer synchronization architecture are separately evaluated, that is, the Manager and the Rendezvous. To obtain λ, the arrival rates of the different types of requests/messages that are delivered to the Rendezvous and to the Managers are calculated and, according to the composition property of Poisson processes, these arrival rates are then summed.

10.3.2 Evaluating the Manager Load

To estimate the load of a Manager networks composed of up to 100 nodes and containing a number of shared objects ranging from 100 to 2000 are considered. Those values correspond to the objects on which clients are actually working. It means that there can be other shared objects but they do not concur to the system load if users are not working on them. In this sense, the maximum number of objects (2000) corresponds to an average of 20 objects on which each client is actually working. Table 10.2 summarizes the parameters and related values that have been adopted for our analysis. In particular, the size of content that must processed by a Manager, when it evaluates an update request for an object, is comprised between 200 bytes and 100 Kbytes, which are the values experienced during K-link+ operation. The corresponding service times vary from 20 ms to 10 secs.: these were set as the values of parameters k and p in the Bounded Pareto distribution, reported in equation 10.1. The

Table 10.2. Parameter values adopted in evaluating the Manager load

Parameter	**Value**
Number of workers, N	100
Average fraction of online and offline nodes, Fon and $Foff$	0.5 and 0.5
Overall number of shared objects, $Nobj$	100 to 2000
Number of Manager nodes, Nmg	1 to 24
Average rate of operations that a worker performs on a shared object while online, Ron	1 each 6000 s
Average rate of operations that a worker performs on a shared object while offline, $Roff$	1 each 12000 s
Minimum task size	200 bytes
Maximum task size	100 Kbytes
Average time required by a Manager to process an update request , $\frac{1}{\mu}$	450 ms

average service time, $\frac{1}{\mu}$, is obtained analytically, as the first moment of the Bounded Pareto distribution. The load of a Manager node is computed as the contribution of two types of messages (see Table 9.3): *online update* requests incoming from online nodes, and *offline update* requests that are received from nodes that reconnect to the network. Actually, several offline requests can be sent by a node when reconnecting, so possibly generating a burst of requests. However, since these bursts come from different nodes at different times, their impact was found to be insignificant, so only the average arrival rates can be considered. The arrival rates corresponding to online and offline update requests, respectively named λon and λoff, are calculated as follows:

$$\lambda on = NN_{obj}R_{on}F_{on}. \tag{10.3}$$

$$\lambda off = NN_{obj}R_{off}F_{off}. \tag{10.4}$$

In the hypothesis that all the Managers receive comparable number of requests, the average arrival rate at a Manager, λ, is computed by dividing the sum of these 2 contributions by the number of Managers:

$$\lambda = \frac{\lambda on + \lambda off}{N_{mg}}. \tag{10.5}$$

From λ, performance indices can now be calculated as described in the previous subsection.

Fig. 10.9 depicts the Manager load ρ in a network with 100, 500, 1000 and 2000 objects and different numbers of available Managers, in the hypothesis that the load is fairly shared among the Managers. The figure shows that the Manager load decreases with the number of Managers, with a negative exponential trend. It can also be noted that, in the presence of a single Manager, the load is sustainable in the case of 100 shared objects, while the presence of more objects leads to a load greater than 1. For example, the load is about 1.65 if there are 500 shared objects, and is even higher with 1000 or 2000 shared objects. In these cases, a multiple Manager configuration is necessary, and the proper number of Managers can be chosen according to the number of objects. For instance, Fig. 10.9 shows that at least 7 Managers are needed if the number of shared objects is 2000, since the load is always larger than 1 if fewer than 7 Managers are available. Fig. 10.10 shows the average overall service

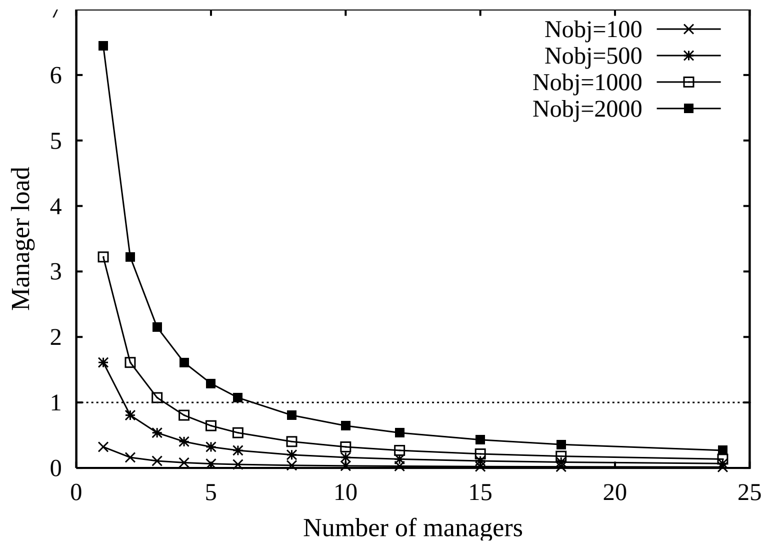


Fig. 10.9. Manager load vs. the number of Managers

time E(T) (that is, the waiting time in the queue plus the actual processing time at the server), which is defined only when the corresponding Manager load (Fig. 10.9) is lower than 1. When the corresponding average load is greater than 1 (see Fig. 10.9) the service time is undefined because the system is overloaded and the requests cannot be served. Moreover, it can be noted that the overall service time tends to be very high as the corresponding value of the load approaches the value of 1, for example, in configurations with 1000 objects and 4 Managers or with 2000 objects and 8 Managers. As the number of Managers increases beyond these values, the service time decreases and becomes acceptable. So far, it was assumed that the load is equally shared among the Managers. In a more realistic scenario, each Manager sustains a different load, either because the objects are unfairly distributed, or because different numbers of update requests are issued for different objects. In this case, a redirection mechanism is devised: as a Manager experiences a load that is approaching the value of 1 (i.e., if the load exceeds a defined threshold), this Manager can decide to ask the Rendezvous to perform a Manager switch procedure (see Section 9.1.4) for some of the managed objects, in order to alleviate its load. This can be effective if there are other Managers that experience a lower load. The effect of request redirection in the case of a load imbalance among Managers was evaluated. Specifically, we con-

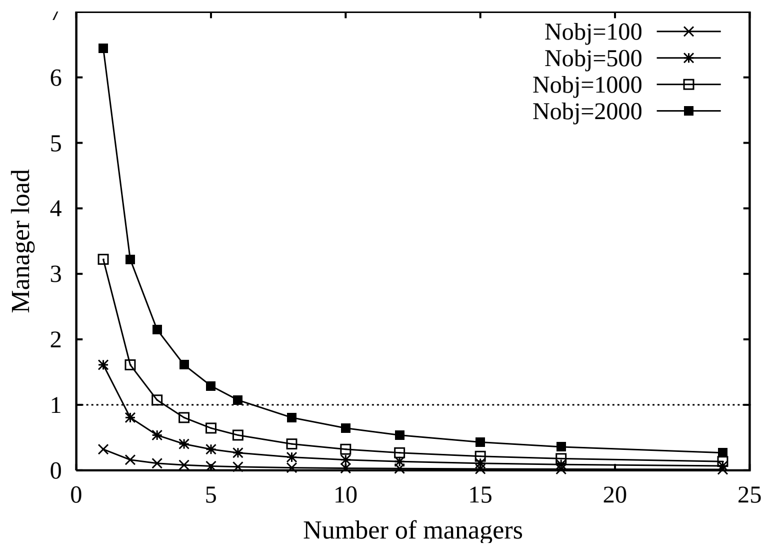


Fig. 10.10. Overall service time of requests vs. the number of Managers

sidered the case in which 6 Managers are available, and 2 of these equally share the management of half the number of objects, was considered. The other 4 Managers share the management of the remaining objects. As the load carried by one of the 2 high-loaded Managers approaches the value of 1 (precisely, as the value of ρ exceeds a threshold set to 0.85), this Manager asks the Rendezvous to perform a Manager switch for a percentage of the objects it is currently handling. This percentage was set to 10% and 20% in two different evaluation tests. Results are reported in Figs. 10.11 and 10.12 that show, respectively, the average load and the overall response time. Indices related to high and low loaded Managers are marked with tags *Hi* and *Lo* in these figures. Results are reported for a number of objects ranging from 600 to 1600, in order to better highlight the effect of redirection, which is clearly visible in this range. Fig. 10.11 shows that the redirection becomes necessary as the number of shared objects approaches 1000, since the corresponding load of ”high-loaded” Managers exceeds the threshold of 0.85. The effect of object redirection is pointed out by means of dashed lines. Note that redirection allows the load of these Managers to be alleviated, though at the cost of increasing the load of the rest of Managers. Redirection permits a better load balancing to be achieved among Managers and, more important, makes the system able to sustain a larger computation load as a whole. Fig. 10.12 shows the effect on the overall service time, which is defined only when the corresponding load is lower than 1. This figure clearly highlights that the redirection mechanism makes the system able to cope with a larger number of objects. Specifically, without redirection, Managers can sustain the load of at most 1200 objects: beyond this value, the load of high loaded Managers exceeds 1 and the service time becomes undefined. Conversely, with values of the redirection percentage set to

10٪ and 20٪ the number of manageable objects can be increased up to about 1350 and 1500 objects, respectively.

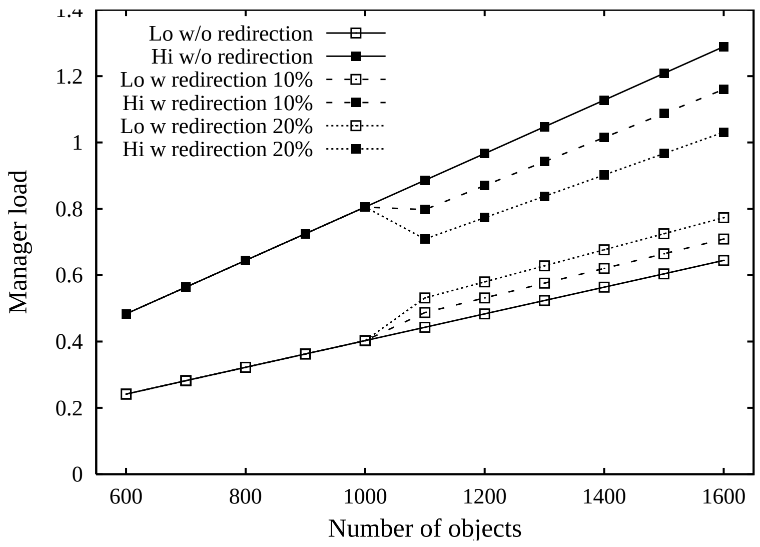


Fig. 10.11. Manager load vs. the number of shared objects

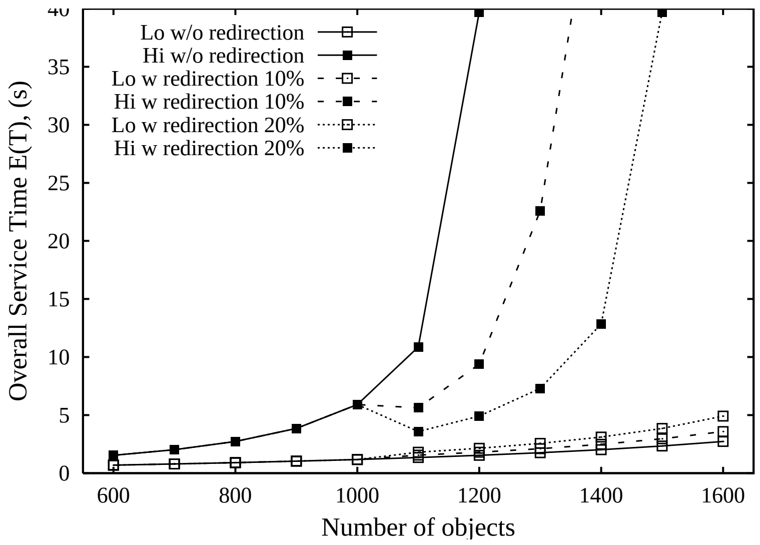


Fig. 10.12. Overall service time vs. the number of shared objects

10.3.3 Evaluating the Rendezvous Load

As described in Section 9.1, K-link+ relies upon a hybrid paradigm with the simultaneous use of centralized and decentralized communication mechanisms. While the presence of several Managers allows for sharing the processing load pertaining to the management of objects, and brokers are exploited to disseminate objects in a P2P fashion, some high level functionalities are kept centralized. In particular, the maintenance of the Consistency Table and the dynamic assignment of Current Managers to objects is consigned to the Rendezvous. This choice was made to exploit the efficiency and security of the centralized paradigm at least for such important operations as the two mentioned above. However, the centralized approach can also have two drawbacks: (i) the fault tolerance management and (ii) a possible high load on the server. In order to cope with the first issue, the K-link+ application manages a possible Rendezvous fault by maintaining a back up Rendezvous that can substitute the current one at each time (this feature is similar to that adopted by JXTA). The second issue is tackled by assigning the Rendezvous only operations that require few computing resources. Indeed, a Rendezvous only copes with metadata documents, which are small and easily manageable, whereas more cumbersome operations, which pertain to the management and update of actual knowledge objects, are distributed among multiple Managers. To verify the last point, the Rendezvous load was evaluated. It is computed as the contribution of three types of messages (see Table 9.1): *version update* messages and *manager leave* messages, which are sent by Managers, and object information request messages issued by workers when they reconnect. The contribution of *create object* messages is not considered, since it is negligible with respect to others. The average rates of these three types of messages are computed as described in the following:

- The average rate of *version update* messages is obtained as follows: (i) the contributions of online and offline requests issued by a single worker, for all their objects (see Table 9.2), are summed; (ii) each time a worker request is accepted by the corresponding Manager, which is assumed to happen 50% of times, a *version update message* is sent by this Manager to the Rendezvous: therefore the event rate computed at the first step is multiplied by 0.5; (iii) finally, the obtained rate is multiplied by the number of workers.
- The average rate of *manager leave* messages is obtained by assuming an average connection time of Managers equal to 5 hours. The corresponding rate, equal to 1 message each 18000 seconds, is then multiplied by the number of Managers.
- The average rate of *object information* request messages is obtained by assuming an average connection time of a worker equal to 3 hours. This rate is then multiplied by the number of workers.

The average time intervals required to process these types of message were estimated on the running K-link+ application. They are equal to about 50 milliseconds (ms) for processing a *version update* message, and 100 ms for processing a *manager leave* or an *object information request* message. Note that these values are much lower than the processing values experienced by the Manager nodes, since the Rendezvous only deals with metadata information, while the Managers deal with actual knowledge objects.

Actually, the load related to the *version update* messages, which depends on the number of shared objects, gives the largest contributions if compared with the load of the other two terms, which do not depend on the number of objects, but on the number of nodes and connection times of workers and Managers. Fig. 10.13 reports the Rendezvous load and shows that it increases with the number of nodes N and the number of shared objects. In this scenario, the CPU utilization of the Rendezvous remains below 65% in all cases. This behavior indicates that to handle up to 100 nodes it is not necessary to adopt a multiple Rendezvous architecture. Fig. 10.14 shows that the overall service time is also acceptable, since it is always lower than 160 milliseconds.

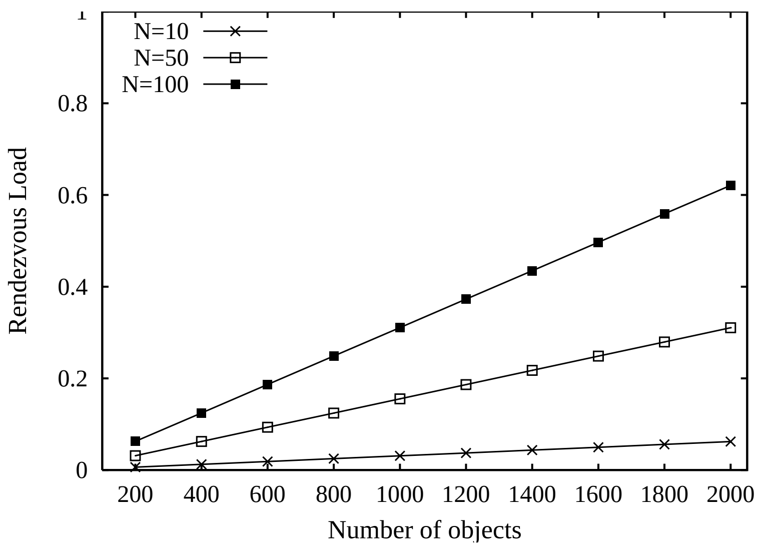


Fig. 10.13. Load of a Rendezvous node

10.4 Related Work

Recently some collaborative systems implementing the virtual office paradigm have been developed, e.g., Zoho [2], ThinkFree [3]. However, most of them are based on a client/server approach and do not include semantic features. The system closer to K-link+ is Groove Virtual Office [4]. Groove is an integrated environment for creating

[2] `http://www.zoho.com`
[3] `http://www.thinkfree.com`
[4] `http://www.groove.net`

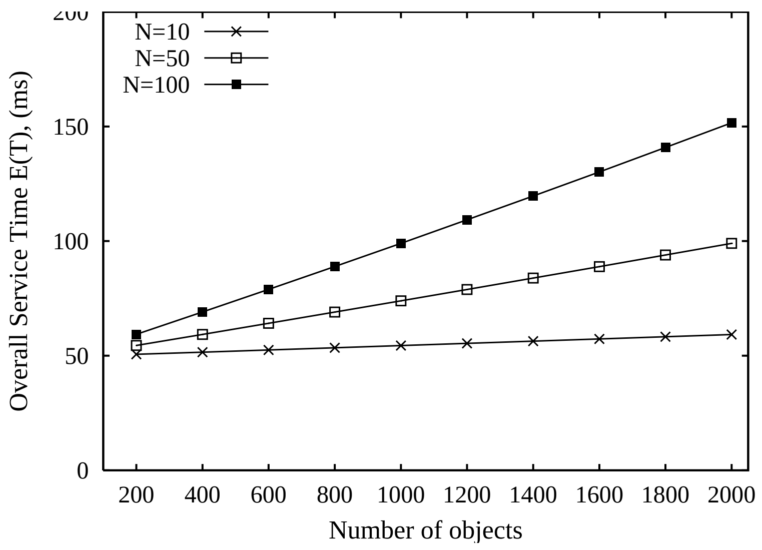


Fig. 10.14. Overall service time of requests on the Rendezvous node

distributed virtual offices. Collaboration activities in Groove take place in a shared application space, which is accessed from an application client called *transceiver*. A shared space, including tools and persistent data, is replicated on every member's computer. Data within a shared space is encrypted, both on disk and over the network, to assure confidentiality and integrity. Both data and commands are transformed, stored and transmitted as XML documents. Every modification made in a shared space is propagated to the other peers. Though its approach is very promising, Groove, differently from K-link+, does not feature semantic functionalities nor exploit ontology mechanisms to cope with knowledge. There are also some systems implementing the DKM paradigm and support semantics through the use of ontologies.

KEEx [14] is a P2P architecture that aims to combine both semantic and P2P technologies. This system implemented in JXTA allows a set of K-nodes to exchange information on a semantic basis. Semantics in KEEx is supported through the notion of context, which represents the peer's personal conceptualization of the world. In KEEx, users are able to autonomously create a context from scratch in order to organize their personal knowledge. The system relies on an automatic mapping algorithm to find correspondences between concepts present in contexts created by different users. KEEx leaves the user completely free about context creation without providing him/her with any organizational background. This principle may result in a weakness for its usage in structured organizations.

SWAP (Semantic Web and Peer-to-Peer) [48] aims at combining ontologies and P2P for knowledge management purposes. SWAP enables local knowledge management through a component called LR (Local node Repository), which gathers knowledge from several sources and represents it in RDF-Schema. SWAP allows searching for knowledge by using a query language called SeRQL, which is an evolution of RQL. Castano et al. [24] proposed a general framework, called Helios, for ontology-based KM in P2P systems. Ontology matching (by the H-match system [21]) could be dynamically performed at different levels of accuracy by exploiting ontology belonging to peers. They also proposed to add a communication infrastructure called Hermes thus creating the H3 framework. This approach allows peers to dynamically joint community of interests and to share their knowledge.

10.5 Discussion and Lesson Learned

This chapter focused on the concrete implementation of the framework architecture described in chapters 8 and 9 in the K-link+ system. This way we can have a practical outcome of the research findings in the context of the KMS-Plus project which originally motivated the design of the framework itself. In particular, here we described and evaluated two important aspects of K-link+. The first is related to semantic search. In this case we constructed a real network of peers and performed several tests of different complexity. The second is related to content consistency and peer synchronization. In this case an analytical performance evaluation, based on the theory of queue networks, confirmed the suitability of the approach.

11

The ERGOT System

11.1 Background and Motivation

The Grid has been conceived as a distributed platform for resource sharing and problem solving in which participants, possibly from different organizations, choose to cooperate by dynamically forming virtual organizations. The first generation of Grids was middleware-centric in the sense that it provided a set of software components and protocols definitions to form a toolkit. More recently, the attention has shifted toward the application layer and, in particular, the concept of service orientation as a way to virtualize and unify resources, services and information has been introduced. As a matter of fact, be either middleware-centric or service-oriented, the Grid requires adequate mechanisms to allow the resource orchestra to coordinate and play the same tune. In this chapter we are concerned in investigating and addressing the service discovery problem. This problem can be characterized in the general context of the Service Oriented Architecture (SOA) model from which the Open Grid Service Architecture (OGSA) has been conceived. The SOA model has been widely recognized as a promising form of distributed computing on the Internet. In this architecture three main actors can be recognized: (i) a service provider, which advertises information about services it wants to make accessible; (ii) a service registry, which stores information about available services; (iii) a service requester, which queries the registry to look for services satisfying some requirements. However, this formulation suffers from some limitations:

- It provides an approach to service discovery based on centralized registries (i.e., UDDI). Such an architecture is unlikely to go through the soaring rate of incoming requests and in case of registry crash jeopardizes the whole service discovery mechanism.
- The lack of semantically-rich service descriptions and complex query mechanisms to perform service matchmaking makes it harder and harder to find services that fit one's needs.

In order to mitigate these issues, two profitable research strands are Peer-to-Peer (P2P) and the Semantic Web (SW).

P2P architectures guarantee decentralization, scalability and fault tolerance. There is a variety of P2P network models ranging from unstructured (a la Gnutella) or hybrid based on Super Peer (SP), to structured based on distributed hash tables (DHTs). In particular in the Grid context, several research strands have investigated how P2P architectures can be exploited for efficient resource (e.g., service) discovery. [164, 78, 168].

On the other hand, SW technologies allow for semantic characterization of resources through the use of ontologies that provide shared and formally defined terminologies describing knowledge domains [68]. In the Grid, a major initiative in this strand of research is the Semantic Open Grid Service Architecture (S-OGSA) [32] which extends OGSA with some services to manage the semantics of Grid resources. In a more general context of Web services, some initiatives such as OWL-S [1], SAWSDL [2] and WSMO [3] have recently been proposed to semantically characterize Web service description.

In this chapter, we investigate how P2P models and SW technologies can be exploited to perform service discovery in open environments (e.g., the Grid). In particular our investigation has been performed in the context of thew CoreGRId Network of Excellence which will be presented in Section 11.1.1. P2P and SW technologies, separately or together, gave birth to several approaches to service discovery. In [12, 99, 42] the semantic-based service discovery issue has been addressed but not that of centralization. Other approaches such as [7, 152] do just the opposite. More comprehensive systems combine P2P and SW technologies in different fashions. The Hypercube [151] exploits ontologies to give positions to peers in the network. The SPiDer system [147] combines ontologies and a SP-based DHT. WSPDS [86] exploits a Semantic Overlay Network [33] and WSDL-S to semantically describe services. Generally speaking, decentralized and semantic-based approaches exploit either SONs (e.g., WSPDS) or a structured architectures such as DHTs in which services are semantically characterized through ontologies (e.g., SPiDer). Both SONs and DHTs have their pros and cons. In a SON, peers choose their neighbors according to a criterion of semantic similarity between services they provide. Here, service discovery is not based on ”exact” matching. Conversely, in a DHT peers are assigned neighbors algorithmically (in a semantic-free way) and services can only be discovered through ”exact” matching. This allows DHTs to obtain maximum precision intended as the fraction of results that match a given key. However, since the definition of discovering [4] itself suggests that one does not exactly know in advance what it is discovering, DHTs do not guarantee maximum recall intended as the fraction of results that are relevant to a request. In this respect, SONs can perform better since they go beyond exact matching. Finally, current approaches do not feature ef-

[1] `http://www.w3.org/Submission/OWL-S`
[2] `http://www.w3.org/2002/ws/sawsdl`
[3] `http://www.wsmo.org`
[4] Discover: to find information, a place or an object, especially for the first time. Cambridge dictionary online, `http://dictionary.cambridge.org`

fective service matchmaking techniques as those devised in centralized initiatives (e.g., [12, 99, 42]).

In this chapter we present the ERGOT (Efficient Routing Grounded On Taxonomy) system combining DHTs and SONs to perform semantic-based service discovery and featuring a service matchmaking mechanism based on an ad-hoc semantic similarity metric. The contributions of this chapter can be summarized as follows:

- ERGOT combines DHTs and SONs to perform semantic-based service discovery. We argue that these two models can benefit from each other in the sense that SONs can be constructed by exploiting DHTs mechanisms and hence the former can help to light the way to the semantics-free content publishing and retrieval approach of the latter.
- ERGOT allows peers to build semantic links, and then a SON, during their normal activities in a DHT (e.g., service advertising). As we will show, semantic links can also be viewed as semantic shortcuts on the DHT.
- ERGOT features different and flexible service discovery mechanisms that can exploit the SON and/or the DHT enhanced with semantic shortcuts.
- ERGOT performs service matchmaking by an ad-hoc similarity metric that compares operation names with related inputs and outputs in a service request and profile. Results are given as numeric values. This approach differs from the state of the art techniques (e.g., [87]) which give as output semantic relations (e.g., exact, subsume) obtained through time-expensive reasoning operations.

The rest of this chapter is organized as follows. In Section 11.2 we provide a brief background on DHTs and SONs and discuss how these two P2P models can be profitably combined. In Section 11.3, we present the semantic service discovery model exploited in ERGOT. Moreover here we introduce the service matchmaker based on semantic similarity. In Section 11.4, we present the architecture of ERGOT and its functioning principles. In particular, we throughly analyze the provider and requester perspectives. In Section 11.5, we review related work and compare ERGOT w.r.t the state of the art.

11.1.1 The CoreGRID Network of Excellence

The CoreGRID Network of Excellence (NoE) [5] aims at strengthening and advancing scientific and technological excellence in the area of Grid and P2P technologies. To achieve this objective, the Network brings together a critical mass of well-established researchers from forty-one institutions who have constructed an ambitious joint program of activities. This joint programme of activity is structured around six complementary research areas that have been selected on the basis of their strategic importance, their research challenges and the recognized European expertise to develop next generation Grid middleware, namely:

- Knowledge & data management.
- Programming models.

[5] `http://www.coregrid.net`

- Architectural issues: scalability, dependability, adaptability.
- Grid information, resource and workflow monitoring services.
- Resource management and scheduling.
- Grid systems, tools and environments.

The work presented in this chapter has been addressed in the area of Knowledge & Data Management.

11.2 On Combining DHTs and SONs

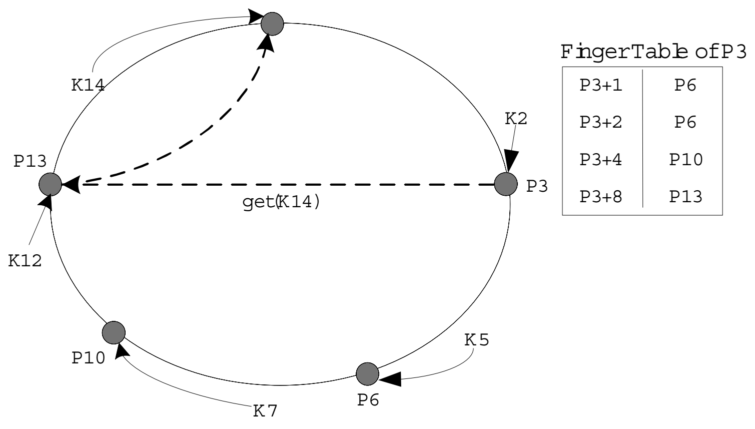
This section provides a brief background on DHTs and SONs and some insights on why it is useful to combine these two P2P architectures in the context of service discovery.

11.2.1 Distributed Hash Tables (DHTs)

DHTs have been recognized as a prominent network paradigm due to their scalability properties and efficiency in retrieving content. In this chapter we focus on the Chord DHT, tough any other DHT can be used instead. Chord [160] organizes peers into an m-bit identifiers ring, in the interval $[0, 2^{(m-1)}]$, which is the basis for routing and locating objects. Both peers and objects are assigned m-bit keys by exploiting consistent hashing which guarantees that the addition or removal of one component in the ring does not significantly affect the network organization. In particular, an object (or a reference to it) is stored in the peer that follows it in the ring. This peer is called the *successor*.

Fig. 11.1 shows a simple 4-bit Chord network. For instance, the key with *id* 2 (i.e., *K2*) is assigned to its successor, that is, peer P3 which is the peer following the *id* 2 in the Chord ring proceeding in clockwise direction. Note that with 4 bits it is possible to create up to 16 keys which will be assigned to peers and objects. In order to perform efficient routing each peer maintains a *finger table* which contains the Chord *ids* of its neighbors. The number of neighbors of a peer is *O(logN)* where *N* is the number of peers in the network. Neighbors are peers located on the ring at exponentially increasing distance from a given peer. For instance, the finger table of *P3* maintains information about *P3*'s neighbors, that is, *P6*, *P10* and *P13*.

The state of the network is maintained by a *stabilization protocol* which refreshes information in the peers' finger tables. Chord is a dynamic system where peers can join and leave the network at their will. When a peer joins the network it is assigned an identifier (e.g., obtained by hashing its IP address) to which it sends a request through an existing node in the ring. This way, the peer can reach its successor from which it obtains the keys it is responsible for. At this point the involved finger tables are updated. In case of departure, keys a peer is responsible for are assigned to its successor. Chord features two basic primitives: *put(key, value)* which is used to publish content in the network and *get(key)* which given a key finds the value associated to it. Chord performs exact look up meaning that a certain content can be retrieved



Finger Table of P3

P3+1	P6
P3+2	P6
P3+4	P10
P3+8	P13

Fig. 11.1. A 4-bits Chord network

only if its *id*, and then the associated key, is exactly known. In fact, a slight difference in terms of *id* (e.g., a file name) can generate completely different keys which can possibly be located in different nodes. In more detail, a request lookup requires at most *O(logN)* hops [160]. Chord routing is clockwise greedy meaning that at each hop a request is forwarded to the peer (in the finger table) whose *id* most immediately precedes the destination point proceeding clockwise. In Fig. 11.1, the request posed by *P3* for the key with *id* 14 (i.e., *K14*) is routed at first hop to the peer in *P3*'s finger table closest to (but not higher than) 14, that is, *P13* which on its turn, by looking at its finger table can easily find the peer responsible for *K14* (i.e., *P1*).

11.2.2 Semantic Overlay Networks (SONs)

In early P2P systems, nodes were typically connected to each other in a ”blind” way (e.g., by selecting a random number of neighbors) and queries were propagated along these links or were collected by a central server and then propagated to the peers in broadcast. Such approaches do not consider at all content stored by peers which actually can be the discriminating factor in building ”intelligent” strategies for clustering peers and routing queries. Crespo and Garcia-Molina proposed the concept of Semantic Overlay Networks (SON) as a new paradigm for organizing peers and enhancing content search [33]. This approach is based on the idea that peers with similar interests, deducted by content they hold, have to be clustered together for speeding up query routing and providing better recall to their information needs. The concept of SON introduces some challenges due to the design of mechanisms to perform peer clustering, classification of content (e.g., documents) and choice of the proper SONs to which a peer has to join. As factor for partitioning an unstructured P2P network in SONs, authors proposed to exploit classification hierarchies. These furnish the semantic underpinning for classifying content, peers and queries. Authors showed the suitability of this approach both in terms of number of messages sent over the network and recall as compared to the Gnutella flooding-based approach. However, in some phases this architecture relies on flooding mechanisms as, for instance, when a peer has to choose the proper SONs to join or when a query reaches a SON (in this phase the query is broadcast within the SON). SON connections among peers are ”logical”, that is, constructed according to a criterion of semantic similarity. Conversely, physical connections do not take into account semantic aspects.

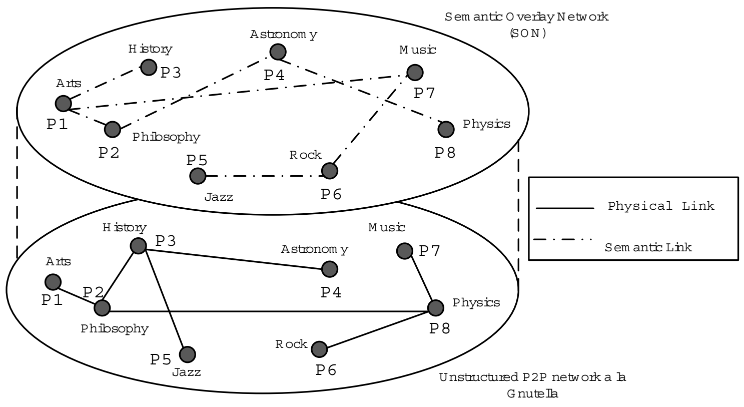


Fig. 11.2. An example of SON

11.2.3 Why combining DHTs and SONs?

In this section, we provide a comparison of the main features of these two models. Table 11.1 reports the results of this analysis which are useful to motivate the design of the ERGOT system that combines DHTs and SONs.

As can be noted in Table 11.1, DHTs are very scalable and guarantee efficient lookup at the cost that the ”identity” of what one is looking for has to be exactly known. SONs are more flexible as intrinsically perform semantic-based query answering and then go beyond exact matching. However, their performance heavily depends on how semantic links are created and the cost to create the semantic links (i.e., how to find neighbors) has also to be taken into account.

DHTs and SONs have been (separately) exploited in recent service discovery initiatives (e.g., [147, 173, 86]). We claim that these two models can profitably be combined in the context of service discovery by observing that:

- Peers, during their normal interaction in the DHT (e.g., service advertising) can discover peers with similar content with which establishing semantic links. This way the SON can be constructed without additional costs. However, the definition of some shared semantic artifact on which the notion of ”similar content” can be defined is required.

Table 11.1. Comparison between DHTs and SONs

	DHTs	**SONs**
Content Placing	Fixed, based on hashing	Each peer is responsible for its content
Content Retrieval	Exact lookup	Flexible
Network Structure	Fixed	Variable
Lookup cost	At most O(logN)	Variable
Topology Management	Stabilization protocol	Peers need to rearrange semantic links
Semantics	None	Exploit some semantic artifact (e.g., ontologies)

- Content lookup can be performed by considering both the exact lookup techniques of the DHT and semantic-similarity based of SONs. In particular, when searching for a particular service, a peer can look at its semantic neighbors to see if they have something which is *similar*, in semantic sense, to the requested service. That complies with the notion of discovery thus going beyond exact lookup.
- The construction of additional links (i.e., semantic links) to those provided by the DHT topology can also improve the DHT's exact lookup. In fact, a peer when searching for a service can consult its semantic links to see if one of the semantic neighbors is responsible (or it is closer than a traditional DHT neighbor) to the key it is looking for. These additional links can be viewed as semantic shortcuts in the DHT.

In the light of these intuitions we devised the ERGOT system that exploits ontologies as semantic artifacts and an ad-hoc service matchmaker to numerically quantify the similarity between service requests and profiles.

11.3 A Service Matchmaker Based on Semantic Similarity

As discussed in Section 11.1, a number of pitfalls can be recognized in the semantic-free Web service description and discovery mechanism exploited in the SOA. In order to overcome these issues and support distributed service discovery, we devised a semantic-based approach to characterize services that exploits two kinds of ontological knowledge: one to describe service functionalities and the other to annotate service operations with related inputs and outputs. Moreover, an ad-hoc service matchmaker based on semantic similarity supports numeric ranking of results related to a request. In this section we elaborate on these aspects.

11.3.1 Category and Domain Ontology Annotations

In ERGOT, services are advertised by providers in the form of semantically-enhanced profiles. In particular, a provider can perform two ”levels” of annotations. A higher level is exploited to associate a service with one or more concepts belonging to a Category Ontology (CO). This approach resembles the service categorization mechanism provided by UDDI in which some standard taxonomies (e.g., UNSPSC, NAICS) are exploited. The aim of this kind of annotations is to ”summarize” service functionalities. Finer-grained annotations are exploited to annotate operations with related inputs and outputs to concepts belonging to a Domain Ontology (DO) with the aim to provide a more detailed characterization of Web services. DO annotations result particularly useful to distinguish between services belonging to the same category.

Fig. 11.3 shows and excerpt of a CO in the bioinformatics domain extracted from the ^{my}Grid ontology [181]. According to our model, a service can be annotated to one or more CO concepts which provide a semantic summarization of the tasks carried out by the service. In Fig. 11.4 it is shown an excerpt of DO, even in this

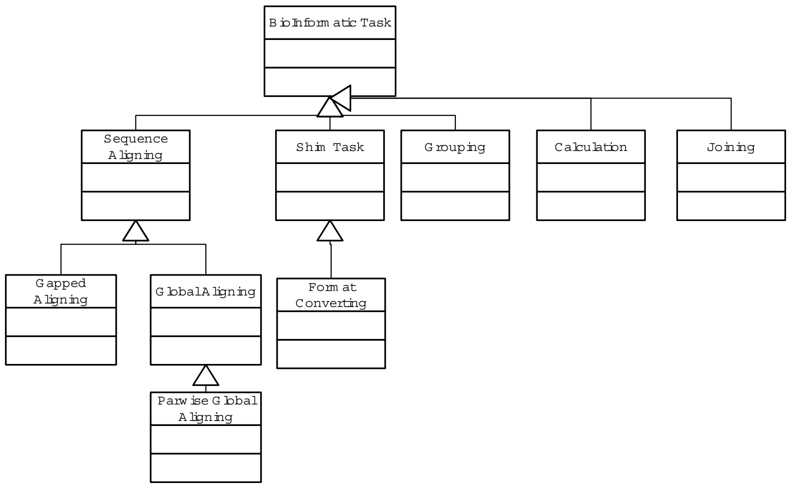


Fig. 11.3. An excerpt of the ^{my}Grid ontology used as Category Ontology

case extracted from the ^{my}Grid ontology. As can be noted, concepts in DO are more specific than those in the CO and can profitably be exploited to annotate service operations with related inputs and outputs.

Fig. 11.5 shows an example of annotated service. This service whose name is *nucleotide alignment* has been annotated to the CO concept *global alignment*. The service features an operation named *nucleodite alignment* that takes as input parameter a *nucleotide alignment request* which has been annotated to the DO concept *fasta format* and has as output parameter a *nucleotide alignment return* which has been annotated to the concept *multiple sequence alignment report*. Service discovery is performed by exploiting both CO and DO concepts. In particular, CO concepts are useful to identify a possible set of candidate matching services as they are used to summarize service functionalities. On the other hand, DO concepts are exploited to

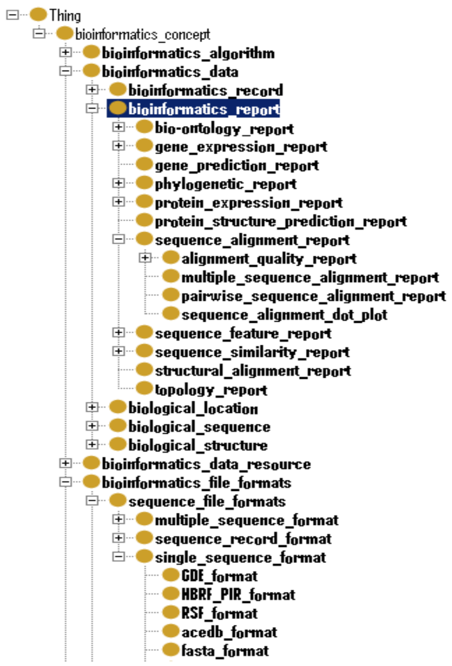


Fig. 11.4. An excerpt of the ^{my}Grid ontology used as Domain Ontology

perform finer-grained analysis of results to find out the most relevant through the mechanism described in Section 11.3.5. Finally, to represent annotations we exploit the lightweight approach of SAWSDL (`http://www.w3.org/2002/ws/sawsdl`).

11.3.2 Preliminary Definitions

After introducing the annotation mechanism, in this section we provide some formal definitions adopted throughout the chapter.

A service profile $P = \langle sn, \mathcal{O}p \rangle$ is defined by a service name sn and a set of operations $\mathcal{O}p$. An operation $Op \in \mathcal{O}p$ has a name n and a set $\mathcal{I}$ and $\mathcal{O}$ of inputs and outputs, respectively:

$$Op = \langle n, \mathcal{I}, \mathcal{O} \rangle$$

Here sn, n, and each $I \in \mathcal{I}$, $O \in \mathcal{O}$ can be annotated with ontology concepts. We write $ann(x)$ to denote the concept that annotates a generic element x.

A service profile forms a hierarchical structure, and we will use a dot notation to refer to its elements. For example, $P.sn$ is the service name, $P.\mathcal{O}p_i.\mathcal{I}_j$ is the j-th input of the i-th operation, and so forth. A service request R has the following structure: $R = \langle \mathcal{C}, \mathcal{O}p \rangle$ where $\mathcal{O}p$ is optional. It has a structure similar to that of a service profile apart for the fact that instead of having a single service name it has a set of concepts (i.e., $\mathcal{C}$) belonging to the Category Ontology (CO) presented in Section 11.3.1.

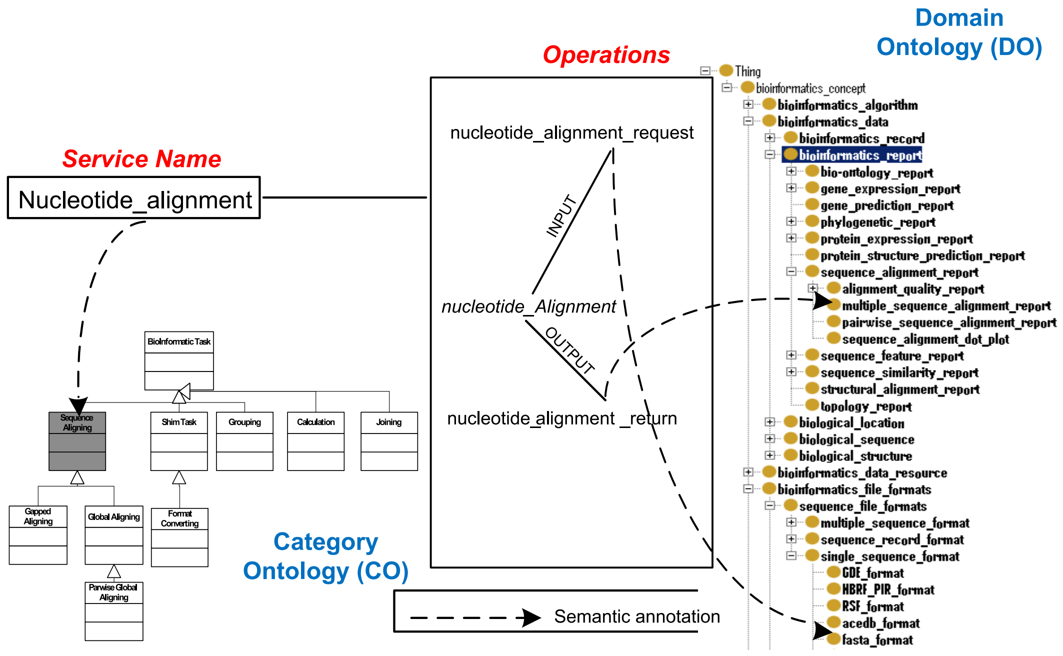


Fig. 11.5. An example of service annotation

11.3.3 On Semantic Service Matchmaking

The task of comparing a service request with a service profile is generally referred to as service matchmaking. UDDI adopts a simple approach based on string comparison. In order to improve the poor accuracy of this technique, several approaches have been proposed. These range from pure logic-based approaches [87] exploiting ontologies to non logic-based that can exploit different techniques such as Information Retrieval [42] or Rough Set Theory [99] just to cite a few. Logic-based initiatives, through reasoning operations identify logic relations, such as *exact*, *plugin*, *subsume*, *fail*, between a request and a service profile. Conversely, non-logic-based approaches provide numeric assessments. However, as also observed by [12], it is difficult to numerically quantify some semantic relations such as *subsume* or *plugin*. Therefore, ranking and interpretation of the relevance of results becomes more challenging. More specifically, in distributed contexts only a few initiatives ([173, 147]) have addressed the problem of result ranking but only taking into account Quality of Service (QoS) indicators and not the semantics associated to operation names with related inputs and outputs. In ERGOT, we devised an ad-hoc mechanism for service matchmaking based on semantic similarity that will be described in the next section.

11.3.4 A Service Matchmaker based on Semantic Similarity

The notion of semantic similarity has been widely recognized as important in many research areas ranging from Artificial Intelligence to Cognitive Sciences. In particular, it aims at quantifying the similarity between two terms by exploiting one or more information sources that can be for example a well-defined ontology (e.g., WordNet [112]). We aim at exploiting semantic similarity for service matchmaking. In this case, the sources of knowledge are the CO and the DO which are exploited for service annotation and discovery. In the literature several approaches have been proposed to assess similarity between concepts in the same or different ontologies. In this work we adopt the approach described in Chapter 7. In particular, the semantic similarity between two concepts $Csim(c_1, c_2)$ is computed as follows:

$$Csim(c_1, c_2) = \begin{cases} 3 \cdot IC(msca(c_1, c_2)) - IC(c_1) - IC(c_2) & \text{if } c_1 \neq c_2 \\ 1 & \text{otherwise} \end{cases}$$

where $msca$ is the most specific common abstraction between c_1 and c_2 and IC represents the information content of a given concept which quantifies the information a concept expresses in terms of the number of hyponyms (i.e., subconcepts) it has in the ontology. In particular the more hyponyms a concept has the less information it expresses. For further details refer to Chapter 7. Our service matchmaker is based on this metric.

11.3.5 Measuring Semantic Similarity between Request and Profile

The aim of our matchmaker is to perform finer-grained comparison between a service request and a service profile thus allowing to numerically quantify in what extent a request R fits with a service profile P. The similarity function $RPsim(R, P)$ between P and R is defined inductively on their common structure, as follows. Initially, we consider the semantic similarity between two concepts above presented. Next, we consider an operation $Op^P \in P.Op$ defined as part of a profile P and an operation $Op^R \in R.Op$ in a request R, and let $\mathcal{I}^P = Op^P.\mathcal{I}$ and $\mathcal{I}^R = Op^R$ be their respective annotated sets of inputs.

The similarity $Isim(\mathcal{I}^P, \mathcal{I}^R)$ between the two input sets is obtained by comparing each concept associated to an input in the request, $I_i^R \in \mathcal{I}^R$, with each concept associated to an input in the profile, $I_j^P \in \mathcal{I}^P$. The similarity is the sum of the best matches, normalized by the number of inputs in the request. Formally:

$$Isim(\mathcal{I}^P, \mathcal{I}^R) = \frac{\sum_{I_i^R \in \mathcal{I}^R} \max_{I_j^P \in \mathcal{I}^P} Csim(ann(I_i^R), ann(I_j^P))}{|\mathcal{I}^R|}$$

The normalization factor $|\mathcal{I}^R|$ is a measure of specificity of the request. To see why this is important, consider two requests R_1 and R_2, where R_1 is vague and contains a strict subset $\mathcal{I}^{R_1} \subset \mathcal{I}^{R_2}$ of the input concepts of R_2. Suppose that both requests are matched with a profile P, and that the similarity between input terms in $\mathcal{I}^{R_2} \setminus \mathcal{I}^{R_1}$ and the inputs of P is very low. In this case, it is reasonable to expect the vaguer request R_1 to have a better match with P than R_2, because P does not offer any of the additional inputs requested by R_2.

The similarity $Osim(\mathcal{O}^P, \mathcal{O}^R)$ between the annotated sets of outputs for an operation is defined similarly to that of the inputs:

$$Osim(\mathcal{O}^P, \mathcal{O}^R) = \frac{\sum_{O_i^R \in \mathcal{O}^R} \max_{O_j^P \in \mathcal{O}^P} Csim(ann(O_i^R), ann(O_j^P))}{|\mathcal{O}^R|}$$

We also define the similarity between two annotated operations names $n^P = Op^P.n$ and $n^R = Op^R.n$:

$$Nsim(n^P, n^R) = \begin{cases} Csim(ann(n^P), ann(n^R)) & \text{if } n^P \neq \perp \text{ and } n^R \neq \perp \\ 0 \text{ otherwise} \end{cases}$$

We can now proceed to define the similarity between a service operation $Op^R \in R.\mathcal{O}p$ and an operation request $Op^P \in P.\mathcal{O}p$, as follows:

$$\begin{aligned} OPsim(Op^R, Op^P) = \; & \alpha \; Nsim(Op^R.n, Op^P.n) + \\ & \beta \; Isim(Op^R.\mathcal{I}, Op^P.\mathcal{I}) + \\ & \gamma \; Osim(Op^R.\mathcal{O}, Op^P.\mathcal{O}) \end{aligned}$$

The weights α, β, and γ are defined by the provider of the annotations for profile P, and account for the different importance that the provider associates to the various annotations of the profile structure.

Finally, the similarity function $RPsim(R, P)$ between a request and a profile is computed by matching each operation request $Op^R \in R.\mathcal{O}p$ with all profile operations $Op^P \in P.\mathcal{O}p$ and adding up the best matches. As we have done with input and output similarity, the sum is normalized by the number of operations specified in the request:

$$RPsim(R, P) = \frac{\sum_{Op^R \in R.\mathcal{O}p} \max_{Op^P \in P.\mathcal{O}p} OPsim(Op^R, Op^P)}{|R.\mathcal{O}p|}$$

11.4 The ERGOT Architecture

In this section we present the ERGOT system that brings together the discussion done in Section 11.2 on combining DHTs and SONs with the service discovery model and matchmaker presented in Section 11.3. In describing ERGOT, we distinguish between the service provider and requester perspective.

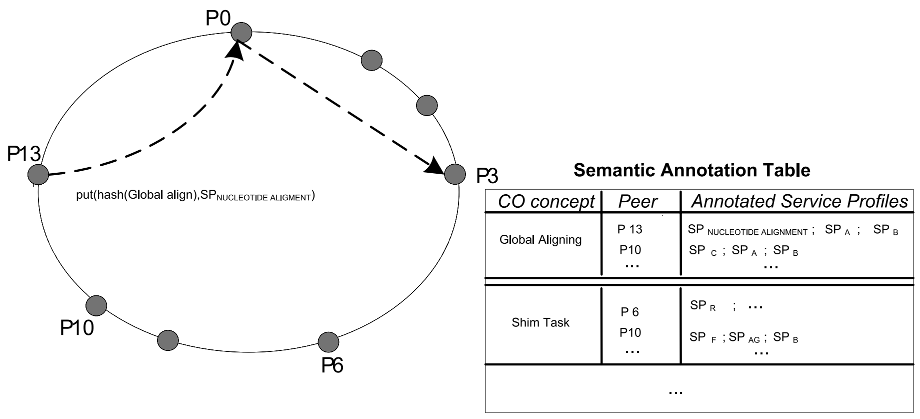
11.4.1 Publishing Semantic Service Profiles

A service provider is allowed to publish service profiles annotated as described in Section 11.3.1. Service profiles are at the basis of the construction of semantic links among peers with similar interests. In this section we elaborate more on these aspects and in particular how to publish service profiles and how to build semantic links.

On Publishing Service Profiles

In order to provide a deeper insight on the ERGOT service publishing mechanism we suppose the knowledge domain to be bioinformatics and services to be annotated by exploiting the ^{my}Grid ontology [181] although any other ontology can be used. The provider annotates services by exploiting two shared ontologies as described in Section 11.3. Briefly, a Category Ontology (CO) is used to summarize service functionalities, whereas a Domain Ontology (DO) is exploited to annotate operations with related inputs and outputs. We assume a peer receives both the CO and DO from the peer it contacts in its join phase in the DHT. We also assume CO concepts to be distributed among the participants to the DHT. In practice, each peer will be responsible for a subset of keys obtained by hashing CO concept identifiers (e.g., their path from the root). Note that concepts that are ”close” in the ontology will be possibly dispersed in different nodes as the DHT hashing mechanism does not preserve locality.

Service profiles are published by exploiting the DHT's *put(key, value)* primitive where each *key* is assigned a *value*. In our context, the *key* is one CO concept. Note that a service can be described by more than one CO concept and therefore it will be published several times. It is interesting also observing that CO concepts are used to publish service profiles whereas DO concepts are used in a second phase to perform finer-grained service matchmaking (see Section 11.3.4). In order this publishing mechanism to work properly a slightly modification to the DHT storing mechanism is required. In fact, we need to keep trace of multiple services annotated to the same concept along with peers that annotated them. In Fig. 11.6 it is shown an example of publication of the *nucleotide alignment* service profile (see Fig. 11.5) belonging to the peer P13.


P0
P13
P3
put(hash(Global align),SP_NUCLEOTIDE ALIGMENT)
P10
P6
Semantic Annotation Table

CO concept	*Peer*	*Annotated Service Profiles*
Global Aligning	P 13 P10 ...	$SP_{NUCLEOTIDE\ ALIGNMENT}$; SP_A ; SP_B SP_C ; SP_A ; SP_B ...
Shim Task	P 6 P10 ...	SP_R ; ... SP_F ; SP_{AG} ; SP_B ...
...		

Fig. 11.6. An example of service publication

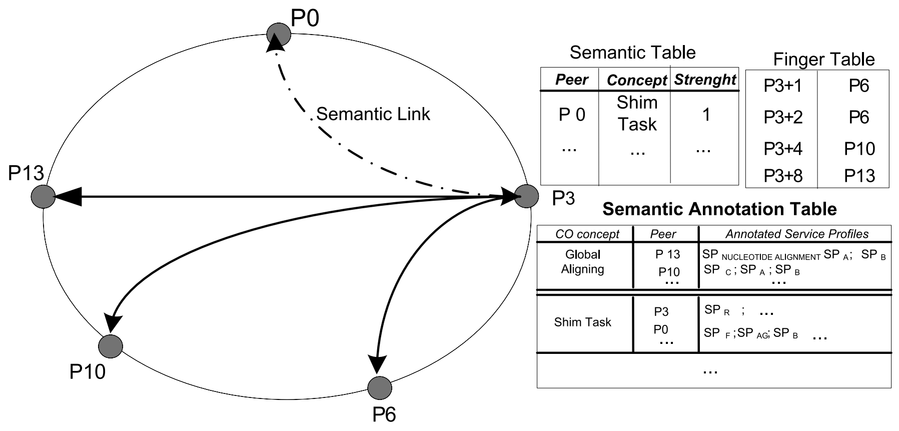
As can be noted, the peer responsible for the CO concept *Global Aligning* to which the service is annotated is *P3*. Hence, *P3* is responsible for keeping track, through the *Semantic Annotation Table*, of all the services annotated by peers to that concept. The second column of the *Semantic Annotation Table* keeps track of the peers that have services annotated to a given concept. Information in the *Semantic Annotation Table* will provide support to the creation of create semantic links as will be discussed in the next section.

On Building Semantic Links

In order to allow building SONs grouping peers with similar content and enhancing the DHT with meaningful semantic links ERGOT exploits the information stored in the *Semantic Annotation Table*. CO concepts, used to publish services in the DHT, can also be viewed as high level semantic descriptions of peer content in terms of services. For instance, in the *Semantic Annotation Table* in Fig. 11.6 both *P13* and *P10* have services annotated to the *Global Aligning* CO concept. As this concept is given a well-defined semantic meaning, one can infer that *P13* and *P10* are semantically-similar to some extent. In ERGOT peers act as rendezvous points for the CO concepts they are responsible for and enable peers with similar interests (in terms of CO concepts) to get in touch with each other. This consideration is at the basis of the construction of semantic links.

In more detail, the process of semantic links creation can be summarized as follows. A peer *p* when publishing semantic service profiles can get, from the peers responsible for the CO concepts to which these profiles are annotated, a list *Ls* of candidate semantic neighbors. This list will be constructed by exploiting the information in the *Semantic Annotation Table*. The number of possible candidates depends on the strategy adopted. A conservative strategy would highlight as possible semantic neighbor each peer that has at least one service annotated to a CO concept to which the peer is annotating its services. A less conservative strategy can fix a threshold in terms of minimum numbers of services so that a peer can be pointed out as a possible semantic neighbor. Again, another strategy could take into account not the number of services but their similarity obtained by exploiting the mechanism defined in Section 11.3.5. The process of finding a set of candidate semantic neighbors can also be performed in the discovering phase. In this case, a peer by scrutinizing the results of a discovery process can establish semantic links with those peers that provide interesting results.

The peer *p*, sends to each peer *pi* in *Ls* a request for establishing semantic links that are constructed by comparing service profiles of *p* with those of *pi* through the matchmaker described in Section 11.3.5. The result of this comparison is a number representing the strength of the semantic link. Moreover, a peer *pi* can suggest to *p* a set of other relevant peers with which the latter has already established semantic links. Note that a shallower approach to semantic link creation could not involve the matchmaker and consider as semantic neighbors all the peers in *Ls* or compare them only in terms of CO concepts to which they have annotated their services. Besides, the frequency of semantic neighbor discovery can be set by the user and the number of neighbors can be fixed apriori. At this point each peer in addition to the links imposed by the DHT topology (in our case Chord) will have an additional set of links, stored in a *Semantic Link Table* as shown in Fig. 11.7. Peers that are related by



Semantic Table

Peer	Concept	Strenght
P 0	Shim Task	1
...	...	...

Finger Table

P3+1	P6
P3+2	P6
P3+4	P10
P3+8	P13

Semantic Annotation Table

CO concept	Peer	Annotated Service Profiles
Global Aligning	P 13 P10 ...	SP $_{\text{NUCLEOTIDE ALIGNMENT}}$ SP $_A$; SP $_B$ SP $_C$; SP $_A$; SP $_B$ ...
Shim Task	P3 P0 ...	SP $_R$; ... SP $_F$;SP $_{AG}$; SP $_B$ ...
...		

Fig. 11.7. Construction of semantic links

semantic links form a SON. For instance, in Fig. 11.7, *P3* and *P0* are not neighbors in the DHT but they are semantic neighbors since they have services published under the CO concept *Shim Task*. To summarize, the DHT coupled with the CO is the basis for building semantic links that allow the building of SONs. The process of service discovery will be detailed in the next section.

11.4.2 Exploiting Semantic Links

ERGOT offers different types of service discovery with different level of flexibility: (i) semantic based discovery (ii) category based discovery; (iii) combined discovery. In the rest of this section we briefly describe them in more detail.

11.4.3 Semantic Based Service Discovery

Semantic-based service discovery involves the peers belonging to SONs to which the requester belongs. A peer poses a service discovery request by choosing one or more CO concepts and possibly specifying operations with related inputs and outputs. The request is then forwarded to semantic neighbor peers that are relevant to the subject (in terms of CO concepts) of the request. The relevance of a request with a semantic neighbor can be computed in different ways. For instance, one selection criterion could be the strength of the semantic links whereas another one could measure the semantic similarity, through the metrics described in Section 11.3, between the request and concepts in the *Semantic Link Table*.

When a peer receives a request it tries to locally fulfill it and then looks in its local *Semantic Link Table* for peers whose interests, in terms of CO concepts, are semantically close to the concepts in the request. The request is then forwarded to the relevant peers. Note that the search in the SON is not ”exact” as in the case of DHT lookup but involves the similarity metric and possibly the matchmaker described in Section 11.3 (depending on how detailed is the request).

11.4.4 Category Based Discovery

It could be that a peer that poses a service request has no, or not enough, semantically related peers in its *Semantic Link Table* to send the request to. This could for example happen when the peer does not share any service profile on the network, resulting in no semantic links creation. When this happens, the request has to be resolved exclusively on the DHT. Even in this case the request comprises one or more CO concepts and possibly some operations with related inputs and outputs. The cost of routing such kind of request in terms of hops is $O(k \cdot logN)$ where k is the number of CO concepts in the request. Note that this approach relies on ”exact” matching as it involves only the DHT. An approach similar to this has been adopted by the SPiDer system [147]. However, differently from SPiDer, ERGOT offers a more flexible mechanism since it can possibly distinguish relevant services by performing, on the set of results retrieved through exact lookup, finer-grained matchmaking.

Combined Discovery

A peer can also pose a combined search discovery request. In this case, the query will involve both SONs and the DHT thus combining the flexibility of SONs with exact and efficient lookup of the DHT.

11.4.5 Enhancing DHTs with Semantic Shortcuts

In ERGOT, semantic links can be viewed as (possibly) additional links to those imposed by the DHT topology (Chord in our case). In particular, semantic neighbors are also peers in the DHT terminology each of which is given an *id*. As an example let’s consider the scenario depicted in Fig. 11.8.

P7 stores in the finger table Chord neighbors and in the *Semantic Link Table* its semantic neighbors discovered as detailed in Section 11.4.1. Here can be noted that *P7* in addition to its Chord neighbors has additional semantic neighbors, that is, *P30* and *P24*. To show what benefits can bring semantic links to the traditional DHT routing, suppose *P7* poses a search on the DHT for a key with *id* 31. According to the Chord protocol, the successor of this key is *P0*. If we consider the standard DHT routing algorithm, this request from *P7* to reach *P0* will involve the following hops: *P21*, *P29*, *P30*, *P0*. Now, by noting *P30* is a semantic neighbor of *P7* we can save some hops. In fact, in this case the request will be routed though the path *P30*, *P0*. Therefore, when performing exact lookup (i.e., looking for a particular key) in the DHT it is possible that a semantic neighbor is given an *id* which is closer to the searched key (in the DHT terminology) than every node in the finger table. This can help to reduce the number of hops needed to reach the node responsible of a given

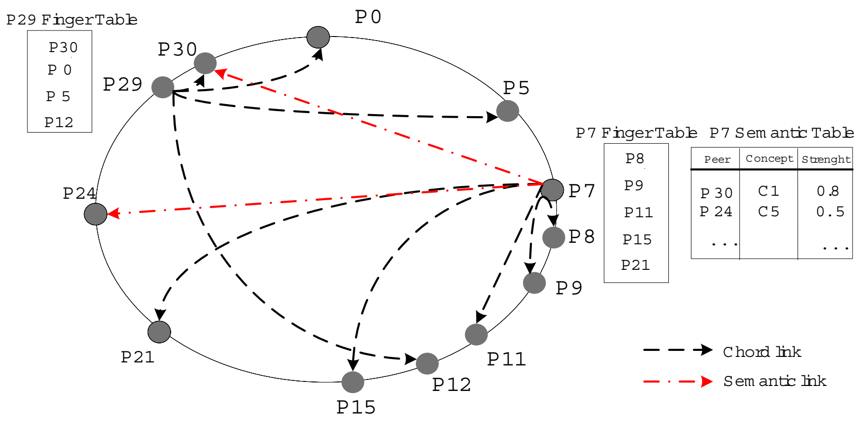


Fig. 11.8. A semantically-enhanced DHT

key. In addition, as done in semantic-based discover, transitivity can also be allowed between semantic links, that is, it would be possible to look at semantic neighbors of a peer's semantic neighbors.

11.5 Related Work and Discussion

The problem of service discovery has been addressed from different, not necessarily disjoint, perspectives thus giving birth to several strands of research. Some of these focus on mitigating centralization by adopting decentralized architectures (e.g., structured, unstructured, hybrid). Others focus on defining semantically-rich data models such as OWL-S, SAWSDL and WSMO to describe and discover services. Again, others focus on performing efficient discovery by adopting, for instance, Information Retrieval to exploit as much as possible information encoded in service descriptions. We investigated the state of the art of service discovery initiatives and identified a list of desiderata a service discovery mechanism should fulfill:

- Decentralization: as the number of services grows up, decentralization becomes a mandatory requirement to avoid bottlenecks of centralized service repositories such as UDDI.
- Semantic rich representation: the more a service description is semantically characterized, the more it is possible by a "machine" to understand if it satisfies a user need. This way, more complex tasks such as service composition can be meaningfully carried out.
- Semantic based discovery: similar to service representation, semantic-based service discovery allows better expressing requests and giving them a precise meaning. In particular, the similarity between a request and a service profile can be performed on a semantic basis.
- Ranking mechanism: ranking is important to present results to a user and allows her to distinguish among a multitude of services all claiming to fulfill her needs.

In the following table, on the basis of the list of desiderata we identified, a set of service discovery approaches are compared. In particular in our subsequent analysis we employ as distinctive factor the network architecture they adopt.

Table 11.2. Comparison among service discovery architectures

	Network Architecture	**Semantic Support**	**Ranking Mechanism**	**Specific techniques**
	Centralized			
ROSSE	Registry-based	No	Yes (numerical)	Rough set theory
Woogle	Registry-based	No	Yes	Ad-hoc clustering technique
COMPAT	Registry-based	Yes	Yes (numerical)	Semantic matching
Matchmaker	Registry-based	Yes	Yes (logic relations)	Reasoning
	Decentralized			
DUDE	DHT	No	No	Prefix Match of service names
Schmidts et al.	DHT	No	No	Hilbert Space Filling curves
Meteor-S	JXTA	Yes	-	-
Hypercube	Ad-hoc topology	Yes	-	-
SPiDeR	DHT-Super peer based	Yes	Yes (based on Qos)	Behavior-based search
Vu et al.	DHT	Yes	Yes (based on Qos)	Ontology partitioning. Bloom filters
ATLAS	DHT	RDF	No	RDF-based
WSPDS	SON	Yes	No	Similarity between peers
ERGOT	DHT+SON	Yes	Yes (numerical)	Combines DHTs and SONs

11.5.1 Centralized Approaches

In this section, we examine centralized architecture for Web service discovery. The ROSSE system [99] exploits Rough Set Theory to perform service discovery in the Grid. In particular, ROSSE identifies some dependences among service properties and is able to compute the Lower and Upper approximation of Grid services that match a user request. It also quantifies the similarity between relevant properties by converting them to numeric values on the basis of a predefined set of possible semantic relations. ROSSE also takes into account QoS by defining a set of heuristics each of which will be weighted toward a final similarity ranking. The system also supports subsumption reasoning by exploiting an ad-hoc reasoner.

Woogle [42] is a system meant to perform efficient service discovery by exploiting textual information encoded in Web service descriptions. In particular, authors devised a novel clustering algorithm which groups parameter names into semantically meaningful concepts and performs similarity computations by an Information Retrieval approach where service descriptions are viewed as vectors of terms. The COMPAT system [12] exploits an ad-hoc ontology framework based on Description Logics and a thesaurus to enable semantic-similarity based service discovery. In particular, service profiles are defined by exploiting both a service ontology, which allows to group services with similar features, and a domain ontology used to annotate operation names, input and output. The system support ranking of results by computing, through the thesaurus, the semantic similarity between a request and a service profile.

The Matchmaker [87] has been one of the first semantic matchmakers for Web services. It allows, through reasoning, to define different semantic relations (e.g., exact, subsume, plug-in) between a user request and a service profile.

ERGOT differs from ROSSE and Woogle since it exploits ontologies to annotate services. It shares some semantic features with COMPACT as it performs semantic based service matchmaking. However, differently from all these approaches it is based on a fully decentralized architecture.

11.5.2 Decentralized Approaches

To cope with the pitfalls of centralized service discovery architectures several decentralized initiatives have been proposed. However, while being decentralized these can adopt different approaches to reach decentralization.

The DUDE system [7] extends the UDDI centralized service discovery mechanism by allowing multiple registries to form a federation with a DHT as a rendezvous point. Here, service information (on the basis of the service name) is distributed among the participants; however the DHT querying mechanism limits the scope of the queries only to relevant registries. To support prefix based querying a service name is hashed different times, one for each prefix, and published on the DHT.

In [152] a DHT based Web service discovery system is proposed. Here a service description is viewed as a set of points in a multidimensional space identified by the possible keywords found in service descriptions. In order to map the multidimensional space to DHT keys, authors exploit Hilbert Space Filling Curves (HSFCs) which ensure that the locality in the multidimensional space will be preserved after the reduction. However, that jeopardizes the hashing mechanism of the original DHT thus leading to load imbalance. In practice, with the dimensional reduction, data elements are not uniformly distributed in the index space, i.e., certain keywords can be more popular and hence the associated index subspace can be more populated. To cope with this issue authors propose two load balancing mechanisms: Load Balancing at Node Join and Load Balancing at Runtime. The system supports wildcard queries and partial keyword queries.

Both DUDE and the system described in [152], differently from ERGOT, while coping with the scalability and fault tolerance issues, neither provide semantic characterization of Web services nor result ranking. Besides, the approach described in [152] uses Hilbert Space Filling Curves as a mechanism to ensure locality preserving hashing. This mechanism, on one side ensures that service descriptions which are similar in the space of keywords describing them will be mapped into similar keys to be stored in the DHT. But, on the other side destroys the nice properties of consistent hashing thus not ensuring that keys will be distributed evenly among nodes. To cope with this issue an ad-hoc load balancing technique has been devised by authors as discussed in [152]. ERGOT avoids this problem by combining DHTs and SONs, that is, efficient lookup and semantic based service discovery.

The Meteor-S system [92] support semantic based organization of Web services in a federation of registries. This system, developed in JXTA, is based on an unstructured P2P network and is mainly meant to organize service publications by identifying the most suitable registry to host a service description.

The WSPDS system [86] aims at constructing an overlay network of peers (here called servents, that is, server and client at the same time) by comparing their data content (Web service descriptions). In particular, nodes create links by comparing the inputs and the outputs of their services by exploiting the matchmaker described in [87]. Besides, the similarity between a query and the peers to whom forward it is computed by the same matchmaker. Each WSDL description has associated a WSIL which contains a pointer to a WSDL-S exploited to semantic annotate a service. The matching between requests and services is enhanced by annotating both with globally shared concepts.

The Hypercube system [151] adopts an ad-hoc network topology. Here, a globally known ontology is exploited to determine the organization of peers.

The Spider system [147] organizes participants into a super peer (SP) based P2P structured network in order to take into account the different computational power of nodes. Each peer is connected to a SP with which it interacts for performing operations of service advertising and discovery. In particular, service discovery is performed by using three different techniques. A keyword based approach, a category-based approach and a behavioral approach. The system supports a reputation component to perform QoS ratings of Web services.

The system proposed in [173] combines ontologies, DHTs and a reputation mechanism based on trusted agents to perform service discovery. One of its key features is the partitioning of a shared ontology, with which describing and querying for services, in concept groups. Each concept group is summarized by a Bloom filter to enable quick concept-membership checking. Hence, a service is described as a set of unordered concepts each of which is represented by the Bloom filter to which each concept belongs. To map service descriptions in the underlying DHT a special hash function is applied to the concatenation of all the keys the description. A QoS component allows rating the quality of a Web service and is used to rank results. Even in this case, the use of a particular hash function that does not guarantee consistent hashing causes that keys will not be distributed evenly among nodes.

The ATLAS system [85] has been designed as a decentralized mechanism for resource discovery in S-OGSA [32]. ATLAS adopts a DHT-based architecture to publish and discovery information about Grid resources on the form of RDF triples. The system allows to pose two types of queries: (i) on-time queries which will be resolved on the network on-the-fly and; (ii) publish/subscribe queries which are continuous queries in the sense that if a resource specified in the query does not exist, the request will be stored in the network and when the resource is published the initial requester is notified. ATLAS allows resolving conjunctive queries expressed in a logic language based on triple patterns.

ERGOT shares some characteristics with the above-mentioned systems. In particular, it provides semantic characterization of services through ontologies. ERGOT, differently from other systems offering a similar feature (e.g., [173, 147, 86]) provides two levels of annotation. A category ontology is used to categorize services and guiding their publishing on the DHT, while a domain ontology is used to semantically characterize operation names with related inputs and outputs. Overall, the main differences w.r.t these systems can be summarized as follows: (i) ERGOT combines DTHs and SONs. To the best of our knowledge ERGOT is the only system combining these two architectures for the purpose of service discovery; (ii) ERGOT adopts a ranking mechanism based on semantic similarity. In particular, after locating services that match a user need by scrutinizing their categorization, it performs fine-grainer matchmaking by computing the semantic similarity between operation names, inputs and outputs in a request and those in a service description. This provides the user with a more immediate interpretation of results since most of current approaches either do not perform ranking of results or only provide ranking based on QoS (e.g., [173]).

11.6 Concluding Remarks

This chapter discussed the application of Semantic Web technologies to the process of discovering semantic services. We devised the ERGOT system which combine the efficiency of DHTs with semantic capabilities of SONs. In particular, a service annotation mechanism has been introduced and the SON is built in a serendipitous way by exploiting normal activities a peer performs in the DHT (e.g., service publishing) and allows to establish semantic links between peers on the basis of their content. ERGOT exploits semantic links to contact peers that are more likely to answer an information need (a service request in our case). The system is in phase of evaluation. However, preliminary results confirmed the suitability of this approach.

Part V

Conclusions and Future Trends

12

Conclusions and Future Trends

The content of thesis concerns two main strands of research. The first investigates the problem of discovering mappings between ontologies. The second one has a more practical outcome and investigates some applications of ontologies in distributed systems. In the rest of this chapter we will summarize the content of this work, remark the main contributions and briefly discuss future trends in the considered fields of research.

12.1 Content Summary

This section recaps the content of this work thus giving an overview in retrospective. An overview of the content of this thesis along with the road map have been presented in the introductory chapter (Chapter 1). One goal of this research was to investigate the ontology mapping problem from different perspectives and provide both methodologies and practical implementations thereof. Discovering ontology mappings concerns the discovery of *semantic links* between distributed ontologies. Mappings are crucial for several semantic applications such as semantic search, semantic query routing just to cite a few. The second objective of this work was to investigate applications of ontologies in distributed systems in two use cases emerging from research projects.

The motivations of this thesis have been laid out in Chapter 2 by investigating some possible applications of ontologies and ontology mapping. In Chapter 2 (Part II), formal definitions for ontology and ontology mapping have been provided along with practical examples. Moreover, a detailed investigation of current ontology mapping techniques has been provided by classifying them on the basis of their peculiar features. This investigation has been useful to identify missing requirements in current mapping solutions.

Part III represents the original contribution in the field of ontology mapping. In particular, in Chapter 4 the Lucene Ontology Matcher (LOM) has been presented. The tenet of this system is to exploit all the sources of linguistic information present in an ontology. These encompass entities' names, comments, labels, instances and other annotation properties. We classified this system as an *offline* system in the sense that it does not address specific aspects of the mapping problem apart from mapping discovery. The second contribution, that is, the User-Friendly Ontology Mapping Environment (UFOme) has been described in Chapter 5. UFOme has been designed to give particular emphasis to the supports that should be provided to a user to facilitate the process of mapping two ontologies. In particular, it provides both GUIs to assist the user in each phase of a mapping task execution and a *strategy predictor* module to help designing the most appropriate mapping strategy and automatically adjust the different ”knobs” (e.g, weights, threshold) involved. In Chapter 6 the SEmantiC COordinator (SECCO) has been described which faces the ontology mapping problem in open and dynamic environments. SECCO includes a new contextual matcher the tenet of which has to be found in the contextual theory of meaning advanced by Miller and Charles [113]. The contextual matcher assesses contextual similarity by computing how concepts can be substituted in each other’s context. The context of a concept has been defined as the set including its properties and neighbors concepts. Finally, Chapter 7 discussed a new similarity metric combining features and intrinsic information content. The innovative aspect of this metric consists in the fact that it projects the Tversky’s feature-based theory of similarity [170] into the information theoretical domain. The suitability of this metric has been tested by correlating its performance with a dataset of human ratings we collected by conducting an online similarity experiment. The results of this experiment have been useful also to establish a possible upper bound we can expect from a computational method in computing semantic similarity.

Part IV presented some applications of ontologies in distributed environments in the context of two research projects. In particular chapters from 8 to 10 discussed K-link+, a P2P system for organizational knowledge management. An abstract definition of the architecture of this system is presented in Chapter 8. Here, particular emphasis has been given to how ontologies can help in designing a knowledge management solution. An ontology framework supporting this architecture has also been motivated and discussed. Chapter 9 focused on the problem of content consistency and peer synchronization which arises in a distributed environment where peers can concurrently work on the local copy of a shared object. A hybrid model combining centralized and decentralized features has been introduced. Chapter 10 discussed the implementation and evaluation of K-link+.

Chapter 11 introduced the ERGOT system to enable semantic-based and scalable service discovery on the Grid. This system has been devised in the context of the CoreGRID NoE through a collaboration with University of Manchester’s Information Management Group. ERGOT combines Distributed Hash Tables (DHTs) and Semantic Overlay Networks (SONs) to perform semantic based service discovery and uses a semantic similarity metric to perform both semantic query routing and results ranking.

12.2 Contributions

The research done in thesis has been motivated by identifying a set of issues and requirements emerging from some use cases covering a wide range of applications (see Chapter 2). In the following we report on how these requirements have been addressed.

- *Mapping Discovery*: this issue emerged in several contexts when it is necessary to deal with distributed and independently designed ontologies. In this thesis we addressed this problem from different perspectives as the facets of a mapping solution can be different depending on the application context. As general approach to ontology mapping we devised a matcher called LOM aimed at exploiting linguistic information encoded in ontology. The motivating factor has been that most of the existing solutions only focus on entitie's names to discover mappings thus underestimating other important sources of linguistic information such as labels, comments and other annotation properties. LOM has been shown to be effective as compared to current mapping strategies (e.g., string matching). Our investigation on ontology mapping then focused on another important aspect underestimated by many mapping solutions, that is, user supports. We designed the UFOme system which can facilitate and make more user-friendly the process of ontology mapping. A striking feature of UFOme is the strategy predictor module which suggests the mapping component to be included and parameter values of a mapping strategy. The effectiveness of this module as compared to manual parameters configuration has been shown through experimental evaluation. Finally, as this thesis devoted special attention to distributed systems we investigated ontology mapping in open environments through the SECCO algorithm. SECCO exploits a novel approach to compute similarity between ontology entities based on the notion of context and has a theoretical underpinning in the Miller and Charles's [113] theory of similarity.
- *Application of ontologies for knowledge management*: this requirement has been motivated by the prominent role that ontologies can have in describing, organizing and managing knowledge. We investigated ontology-based knowledge management in the context of an Italian research project named KMS-Plus. Our investigation led to the design and implementation of K-link+, a P2P system supporting collaborative work and knowledge management in an organization. K-link+ features different tools and a content consistency and peer synchronization mechanism to allow individual knowledge workers to concurrently work on the same piece of knowledge (e.g., document). The system has been fully implemented and tested in a real network.
- *Computing semantic similarity*: this requirements emerged in several contexts ranging from mapping discovery to semantic based query routing. We studied the problem of determining semantic similarity between ontology concepts and devised a new similarity metric which yields results above the state of the art.
- *Semantic Web service discovery*: in this respect we investigated the combination of P2P and Semantic Web technologies to perform scalable service discovery. The outcome of this research has been the ERGOT system which combines

DHTs and SONs and exploits a mechanism based on semantic similarity to route queries and rank results. The system is currently in phase of evaluation.

12.3 Future Trends

Semantic technologies have been placed in the list of top ten disruptive technologies by market analysts such as the Gartner group [1] in the next five years. This section will discuss some interesting topics around a subset of semantic technologies, that is, ontology mapping and ontologies.

12.3.1 Ontology Mapping

The ontology mapping problem has been a core issue in recent research on ontology. To date, many diverse solutions addressing this problem have been proposed as discussed in Chapter 3. Despite the huge amount of research produced around this topic, from the work presented in this thesis, one can see that it is still possible to consider extensions both in research and concrete applications. This section will report on some open areas of research that will influence or will be influenced by ontology mapping.

- *User supports*: in current research on ontology mapping there is no integrated solution that is a clear success, which is robust enough to be the basis for future development, and which is usable by non expert users. Even if the contribution presented in Chapter 5 is one of the first attempt toward this goal, it is still far from being a stable and complete mapping solution. Therefore, in this respect, there are some interesting research issues that need to be addressed. As an example, it should be valuable to investigate the quality of mapping when presenting them to the user. Moreover, as a mapping solution usually includes many individual matchers it is possible that it performs well in some cases and not so well in some other cases. This makes the issues of (i) matcher selection, (ii) matcher combination and (iii) matcher tuning of prime importance. These issues have been addressed by the UFOme system, but much work remains to be done.
- *Performance*: the performance issue is of paramount importance to evaluate the scalability of mapping solutions. Moreover, in a off-the-shelf tool it is not reasonable that a user should wait too long for the system to respond. This issue has been addressed by some system (e.g., [45]) even if it is still far from being solved. Moreover, from a recent ontology evaluation initiative (i.e, the OAEI), emerged that several mapping system present problems of memory usage as they ran out of memory in some tests involving large ontologies.
- *Theoretical Underpinning*: the mapping problem has been addressed from different perspectives. However, its theoretical foundations have not been completely addressed. Some initiatives such as [16] worked on giving an overview of mapping with its theoretical foundations. Therefore, shedding more light on the notion of ontology mapping is ongoing work.

[1] `http://www.gartner.com/it/page.jsp?id=681107`

- *Evaluation*: to date, there are some efforts concerned to establish general evaluation guidelines for ontology mapping [52]. However, more efforts are required, for instance, on evaluating complex alignments.
- *Reasoning*: the process of ontology mapping is tied to the particular application that should use mappings. Once mapping have been found it is necessary to reason about them for several purposes such as finding missing mappings, strengthening, evaluating or repairing them. Some initiatives in this direction are discussed in [110] and [50].
- *Uncertainty*: As a mapping solution usually combines different individual matching strategies, the problem of managing uncertainty arises. A way of modeling ontology mapping as an uncertain process is to use similarity matrices as a measure of certainty. A matcher then is measured by the fit of its estimation of a certainty of a correspondence to the real world. Some mapping initiatives taking into count uncertainty are discussed in [119] and [176].

12.3.2 Applications of Semantic Web Technologies

As general trend, semantics is expected to be a key element for the transformation of information to knowledge. In particular, the massive use of ontologies will enable far more effective machine to machine communication thus enabling a better manipulation of data they hold and intelligent actions based upon that data. The ability of semantic technology to describe complex systems and apply computing power to analyze them goes far beyond traditional approaches to knowledge management. Semantic technology provides standards (OWL, RDF, etc.) and structure that allow information to be described in a way that captures what it is, what it means in a machine-readable form. As more systems become interconnected both in the enterprise and across the Web and as new composite applications and services emerge, having consistent models to evaluate information fit becomes critical to accurate analysis. Business Managers are frustrated that the data they require to help them make critical decisions is often scattered across many different systems and difficult to get to and understand. Therefore, more and more firms started to consider investments in semantic technologies. The desiderata are that semantic technologies will help knowledge workers by bring them the right information understood in context to help them make critical decisions.

As for semantics in P2P systems, whilst the Semantic Web and ontologies provide us with a mechanism for facilitating semantic information management and processing, they focus more on local and static situations, rather than a distributed and dynamic environment. Because they are innately decentralized, P2P systems can help exploit the full potential of the Semantic Web's capabilities. In other words, P2P systems can act as a fundamental platform for the searching and sharing of distributed information by using the Semantic Web technology in a near future.

Finally, Semantic Web services are gaining momentum as they enable a precise definition of service capabilities thus facilitating some tasks such as discovery and composition. Despite numerous initiatives in this field (e.g., OWL-S, WSMF, WSDL-S, METEOR-S), there are no universally-accepted frameworks covering all the phases of a Web service lifecycle from publishing to discovery. Besides, the success of Web services is in some sense related to ontology mapping as in the process of discovery it is necessary to identify which services ”match” a particular request. Therefore, Semantic Web services are an area with a lot of research going on.

References

1. A. Abecker, A. Bernardi, K. Hinkelmann, O. Kuhn, and M. Sintek. Toward a Technology for Organizational Memories. *IEEE Intelligent Systems and Their Applications*, 13(3):40–48, May–June 1998.
2. K. Aberer, P. Cudré-Mauroux, M. Hauswirth, and T. V. Pelt. Gridvine: Building Internet-Scale Semantic Overlay Networks. In *Proc. of ISWC*, pages 107–121, 2004.
3. M. Alavi and D. Leidner. Review: Knowledge Management and Knowledge Management Systems: Conceptual Foundations and Research Issues. *MIS Quarterly*, 25:107–136, 2001.
4. A. Aldea, R. Baares-alcntara, J. Bocio, J. Gramajo, and D. Isern. An Ontology-Based Knowledge Management Platform. In *Proc. of IIWeb-03*, pages 177–182, 2003.
5. C. Axton, R. Gear, N. Macehiter, and E. Woods. Peer-to-Peer Computing: Applications and Infrastructure. Technical report, Ovum, 2002.
6. R. A. Baeza-Yates and B. A. Ribeiro-Neto. *Modern Information Retrieval*. ACM Press / Addison-Wesley, 1999.
7. S. Banerjee, S. Basu, S. Garg, S. Garg, S.-J. Lee, P. Mullan, and P. Sharma. Scalable Grid Service Discovery based on UDDI. In *Proc. of MGC '05*, pages 1–6, New York, NY, USA, 2005. ACM.
8. S. Berkovsky, Y. Eytani, and A. Gal. Measuring the Relative Performance of Schema Matchers. In *Proc. of WI '05*, pages 366–371, Washington, DC, USA, 2005. IEEE Computer Society.
9. T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web. *Scientific American*, 284(5):34–43, 2001.
10. P. A. Bernstein and S. Melnik. Model Management 2.0: Manipulating Richer Mappings. In *Proc. of SIGMOD Conference*, pages 1–12, 2007.
11. W. L. Bethea, C. Fink, and J. Beecher-Deighan. JHU/APL Onto-Mapology Results for OAEI 2006. In *Ontology Matching*, 2006.
12. D. Bianchini, V. Antonellis, and M. Melchiori. Flexible Semantic-Based Service Matchmaking and Discovery. *World Wide Web*, 11(2):227–251, 2008.
13. V. D. Blondel, A. Gajardo, M. Heymans, P. Senellart, and P. V. Dooren. A Measure of Similarity between Graph Vertices. *CoRR*, cs.IR/0407061, 2004.
14. M. Bonifacio, P. Bouquet, G. Mameli, and M. Nori. KEx: A Peer-to-Peer Solution for Distributed Knowledge Management. In *Proc. of PAKM '02*, pages 490–500, London, UK, 2002. Springer-Verlag.

15. M. Bonifacio, P. Bouquet, G. Mameli, and M. Nori. Peer-Mediated Distributed Knowledge Management. In *Proc. of AMKM*, pages 31–47, 2003.
16. P. Bouquet, J. Euzenat, E. Franconi, L. Serafini, G. Stamou, and S. Tessaris. Specification of a Common Framework for Characterzing Alignment. Technical report, Knowledge Web, 2004.
17. P. Bouquet, L. Serafini, and S. Zanobini. Semantic Coordination: A New Approach and an Application. In *Proc. of ISWC*, pages 130–145, 2003.
18. H. G. Budanitsky A. Semantic Distance in WordNet: an Experimental Application Oriented Evaluation of Five Measures. In *Proc. of NACCL 2001*, pages 29–34, 2001.
19. J. Cardoso and A. P. Sheth. *Semantic Web Services, Processes and Applications (Semantic Web and Beyond: Computing for Human Experience)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.
20. S. Castano, A. Ferrara, and G. Messa. Results of the H-Match Ontology Matchmaker in OAEI 2006. In *Proc. of Ontology Matching*, 2006.
21. S. Castano, A. Ferrara, and S. Montanelli. H-Match an Algorithm for Dynamically Matching Ontologies in Peer-based Systems. In *Proc. of SWDB*, pages 231–250, 2003.
22. S. Castano, A. Ferrara, and S. Montanelli. Matching Ontologies in Open Networked Systems: Techniques and Applications. *Journal on Data Semantics*, pages 25–63, 2006.
23. S. Castano, A. Ferrara, S. Montanelli, and G. Racca. From Surface to Intensive Matching of Semantic Web Ontologies. In *Proc. of WEBS Workshop*, pages 140–144, 2004.
24. S. Castano, A. Ferrara, S. Montanelli, and D. Zucchelli. HELIOS: a General Framework for Ontology-based Knowledge Sharing and Evolution in P2P Systems. In *Proc. of DEXA '03*, page 597, Washington, DC, USA, 2003. IEEE Computer Society.
25. P. Castells, M. Fernández, and D. Vallet. An Adaptation of the Vector-Space Model for Ontology-Based Information Retrieval. *IEEE TKDE*, 19(2):261–272, 2007.
26. T. Chang and M. Ahamad. Improving Service Performance through Object Replication in Middleware: A Peer-to-Peer Approach. In *Proc. of P2P*, pages 245–252, 2005.
27. X. Chen, S. Ren, H. Wang, and X. zhang. SCOPE: Scalable Consistency Maintenance in Structured P2P Systems. In *Proc. of INFOCOM 2005*, volume 3, pages 1502–1513, 13–17 March 2005.
28. Y. Chen, R. H. Katz, and J. Kubiatowicz. Dynamic Replica Placement for Scalable Content Delivery. In *Proc. of IPTPS '01*, pages 306–318, London, UK, 2002. Springer-Verlag.
29. N. Choi, I.-Y. Song, and H. Han. A Survey on Ontology Mapping. *SIGMOD Record*, 35(3):34–41, 2006.
30. R. L. Cilibrasi and P. M. B. Vitanyi. The Google Similarity Distance. *IEEE TKDE*, 19(3):370–383, 2007.
31. E. Cohen and S. Shenker. Replication Strategies in Unstructured Peer-to-Peer Networks. *SIGCOMM Comput. Commun. Rev.*, 32(4):177–190, 2002.
32. O. Corcho, P. Alper, I. Kotsiopoulos, P. Missier, S. Bechhofer, and C. A. Goble. An Overview of S-OGSA: A Reference Semantic Grid Architecture. *Journal of Web Semantics*, 4(2):102–115, 2006.
33. A. Crespo and H. Garcia-Molina. Semantic Overlay Networks for P2P Systems. In *Proc. of AP2PC*, pages 1–13, 2004.
34. M. E. Crovella and A. Bestavros. Self-similarity in World Wide Web Traffic: Evidence and Possible Causes. *IEEE/ACM Transactions on Networking*, 5:835–846, 1997.
35. M. E. Crovella, M. S. Taqqu, and A. Bestavros. *Heavy-tailed Probability Distributions in the World Wide Web*, chapter A Practical Guide To Heavy Tails, pages 3–26. Chapman & Hall, 1998.

36. B. Danushka, M. Yutaka, and I. Mitsuru. Measuring Semantic Similarity between Words Using Web Search Engines. In *Proc. of WWW2007*, pages 757–766, 2007.
37. J. Davies, M. D. Lytras, and A. P. Sheth. Guest Editors' Introduction: Semantic-Web-Based Knowledge Management. *IEEE Internet Computing*, 11(5):14–16, 2007.
38. J. Devore. *Probability and Statistics for Engineering and the Sciences*. International Thomson Publishing Company, 1999.
39. H. H. Do, S. Melnik, and E. Rahm. Comparison of Schema Matching Evaluations. In *Web, Web-Services, and Database Systems*, pages 221–237, 2002.
40. H. H. Do and E. Rahm. COMA - A System for Flexible Combination of Schema Matching Approaches. In *Proc. of VLDB*, pages 610–621, 2002.
41. A. Doan, J. Madhavan, R. Dhamankar, P. Domingos, and A. Y. Halevy. Learning to match ontologies on the Semantic Web. *VLDB Journal*, 12(4):303–319, 2003.
42. X. Dong, A. Y. Halevy, J. Madhavan, E. Nemes, and J. Zhang. Simlarity Search for Web Services. In *Proc. of VLDB*, pages 372–383, 2004.
43. P. F. Drucker. *A Post Capitalist Society*. HarperCollins, New York, 1993.
44. M. Ehrig. *Ontology Alignment: Bridging the Semantic Gap (Semantic Web and Beyond)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.
45. M. Ehrig and S. Staab. QOM - Quick Ontology Mapping. In *Proc. of ISWC*, pages 683–697, 2004.
46. M. Ehrig and Y. Sure. Ontology Mapping - An Integrated Approach. In *Proc. of ESWS*, pages 76–91, 2004.
47. M. Ehrig and Y. Sure. FOAM- A Framework for Ontology Alignment and Mapping. In *Proc. of ISWC*, 2005. Demo.
48. M. Ehrig, C. Tempich, J. Broekstra, F. van Harmelen, M. Sabou, R. Siebes, S. Staab, and H. Stuckenschmidt. SWAP - ontology-based knowledge management with peer-to-peer technology, 2003.
49. J. Euzenat. An API for Ontology Alignment. In *Proc. of ISWC*, pages 698–712, 2004.
50. J. Euzenat. Semantic Precision and Recall for Ontology Alignment Evaluation. In *Proc. of IJCAI*, pages 348–353, 2007.
51. J. Euzenat, T. L. Bach, J. Barrasa, P. Bouquet, J. D. Bo, Rose, Dieng-Kuntz, M. Ehrig, M. Hauswirth, M. Jarrar, R. Lara, D. Maynard, A. Napoli, G. Stamou, H. Stuckenschmidt, P. Shvaiko, S. Tessaris, S. V. Acker, and I. Zaihrayeu. State of the Art on Ontology Alignment. Technical report, Knowledge Web NoE, 2004.
52. J. Euzenat, M. Ehrig, and R. G. . Castro. Specification of a Benchmarking Methodology for Alignment Techniques. Technical report, Knowledge Web NoE, 2005.
53. J. Euzenat, P. Guégan, and P. Valtchev. OLA in the OAEI 2005 Alignment Contest. In *Proc. of Integrating Ontologies*, 2005.
54. J. Euzenat, M. Mochhol, P. Shvaiko, H. Stuckenschmidt, O. Svatek, W. R. Van Hage, and M. Yatskevich. Results of the Ontology Alignment Evaluation Initiative. In *Proc. of International Workshop on Ontology Matching*, 2006.
55. J. Euzenat and P. Valtchev. Similarity-Based Ontology Alignment in OWL-Lite. In *Proc. of ECAI*, pages 333–337, 2004.
56. P. Euzenat, J. Shvaiko. *Ontology Matching*. Springer, 2007.
57. S. M. Falconer, N. Noy, and M.-A. D. Storey. Towards Understanding the Needs of Cognitive Support for Ontology Mapping. In *Ontology Matching*, 2006.
58. S. M. Falconer and M.-A. D. Storey. A Cognitive Support Framework for Ontology Mapping. In *Proc of ISWC/ASWC*, pages 114–127, 2007.
59. G. Fauconnier and M. Turner. Conceptual Integration Networks. *Cognitive Science*, 22(2):133–187, 1998.

60. D. Fensel. *Ontologies: A Silver Bullet for Knowledge Management and Electronic Commerce*. Springer, 2004.
61. J. Frazer. A Representation of Context for Computer Supported Collaborative Design. *Automation in Construction*, 10:715–729(15), 2001.
62. F. Gandon, R. Dieng-Kuntz, O. Corby, and A. Giboin. Semantic Web and Multi-Agents Approach to Corporate Memory Management. In *Intelligent Information Processing*, pages 103–115, 2002.
63. Y. Ge, Y. Yu, X. Zhu, S. Huang, and M. Xu. OntoVote: a Scalable Distributed Vote-collecting Mechanism for Ontology Drift on a P2P Platform. *Knowl. Eng. Rev.*, 18(3):257–263, 2003.
64. F. Giunchiglia and C. Ghidini. Local Models Semantics, or Contextual Reasoning = Locality + Compatibility. *Artificial Intelligence*, 127:2001, 2001.
65. F. Giunchiglia, P. Shvaiko, and M. Yatskevich. S-Match: an Algorithm and an Implementation of Semantic Matching. In *Proc. of ESWS*, pages 61–75, 2004.
66. V. Gopalakrishnan, B. Silaghi, B. Bhattacharjee, and P. Keleher. Adaptive Replication in Peer-to-Peer Systems. In *Proc of ICDCS*, pages 360–369, 2004.
67. T. Gruber. Ontology. In *Encyclopedia of Database Systems*. Springer, 2008.
68. T. R. Gruber. A Translation Approach to Portable Ontologies. *Knowledge Acquisition*, 5(2):199–220, 1993.
69. N. Guarino and G. P. Ontologies and Knowledge Bases: Towards a Terminological Clarification. *Towards Very Large Knowledge Bases*, pages 25–32, 1995.
70. R. V. Guha, R. McCool, and E. Miller. Semantic Search. In *Proc. of WWW*, pages 700–709, 2003.
71. C. Hai, J. Hanhua. SemreX: Efficient Search in Semantic Overlay for Literature Retrieval. *Future Gener. Comput. Syst.*, 24(6):475–488, 2008.
72. A. Y. Halevy, Z. G. Ives, J. Madhavan, P. Mork, D. Suciu, and I. Tatarinov. The Piazza Peer Data Management System. *IEEE TKDE*, 16(7):787–798, July 2004.
73. B. Haverkort. *Performance of Computer Communication Systems*. John Wiley & Sons, 1998.
74. G. Hirst and D. St-Onge. *WordNet: An Electronic Lexical Database*, chapter Lexical Chains as Representations of Context for the Detection and Correction of Malapropisms. MIT Press, 1998.
75. A. Hliaoutakis, G. Varelas, E. Voutsakis, E. G. M. Petrakis, and E. E. Milios. Information Retrieval by Semantic Similarity. *Int. J. SWIS*, 2(3):55–73, 2006.
76. I. Horrocks and P. F. Patel-Schneider. A Proposal for an OWL Rules Language. In *Proc of WWW*, pages 723–731, 2004.
77. W. Hu, G. Cheng, D. Zheng, X. Zhong, and Y. Qu. The Results of Falcon-AO in the OAEI 2006 Campaign. In *Ontology Matching*, 2006.
78. A. Iamnitchi and D. Talia. P2P Computing and Interaction with Grids. *Future Generation Comp. Syst.*, 21(3):331–332, 2005.
79. R. Ichise. Machine Learning Approach for Ontology Mapping Using Multiple Concept Similarity Measures. In *Proc. of ICIS '08*, pages 340–346, Los Alamitos, CA, USA, 2008. IEEE Computer Society.
80. A. Isaac, L. van der Meij, S. Schlobach, and S. Wang. An Empirical Study of Instance-Based Ontology Matching. In *Proc. of ISWC/ASWC*, pages 253–266, 2007.
81. R. Jain. *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*. Wiley- Interscience, 1991.
82. K. Janowicz. Semantic Similarity Blog, http://www.similarity-blog.de/.
83. N. Jian, W. Hu, G. Cheng, and Y. Qu. Falcon-AO: Aligning Ontologies with Falcon. In *Proc. of K-CAP Workshop on Integrating Ontologies*, pages 87–93, 2005.

84. J. Jiang and D. Conrath. Semantic Similarity Based on Corpus Statistics and Lexical Taxonomy. In *Proc. of ROCLING X*, 1997.
85. Z. Kaoudi, M. Koubarakis, K. Kyzirakos, M. Magiridou, I. Miliaraki, and A. Papadakis-Pesaresi. Publishing, Discovering and Updating Semantic Grid Resources using DHTs. In *Proc. of CoreGRID Workshop on Grid Programming Model, Grid and P2P Systems Architecture, Grid Systems, Tools and Environments.*, 2007.
86. F. B. Kashani, C.-C. Chen, and C. Shahabi. WSPDS: Web Services Peer-to-Peer Discovery Service. In *Proc. of International Conference on Internet Computing*, pages 733–743, 2004.
87. T. Kawamura, J.-A. D. Blasio, T. Hasegawa, M. Paolucci, and K. P. Sycara. Public Deployment of Semantic Service Matchmaker with UDDI Business Registry. In *Proc. of ISWC*, pages 752–766, 2004.
88. L. Kerschberg. Knowledge Management in Heterogeneous Data Warehouse Environments. In *Proc. of DaWaK*, pages 1–10. Springer, 2001.
89. T. Kirsten, A. Thor, and E. Rahm. Instance-Based Matching of Large Life Science Ontologies. In *Proc. of DILS*, pages 172–187, 2007.
90. K. Kotis, A. G. Valarakos, and G. A. Vouros. AUTOMS: Automated Ontology Mapping through Synthesis of Methods. In *Ontology Matching*, 2006.
91. K. Kotis, G. A. Vouros, and K. Stergiou. Towards Automatic Merging of Domain Ontologies: The HCONE-merge Approach. *Journal of Web Semantics*, 4(1):60–79, 2006.
92. V. Kunal, K. Sivashanmugam, A. Sheth, A. Patil, S. Oundhakar, and J. Miller. METEOR-S WSDI: A Scalable P2P Infrastructure of Registries for Semantic Publication and Discovery of Web Services. *Inf. Technol. and Management*, 6(1):17–39, 2005.
93. P. Lambrix and H. Tan. A Tool for Evaluating Ontology Alignment Strategies. *Journal on Data Semantics*, 8:182–202, 2007.
94. L. Lamport. How to Make a Multiprocessor Computer That Correctly Executes Multiprocess Programs. *IEEE Trans. Computers*, 28(9):690–691, 1979.
95. M. Lauer. *Designing Statistical Language Learners: Experiments on Noun Compounds*. PhD thesis, Macquarie University, 1996.
96. J. Lave and E. Wenger. *Situated Learning. Legitimate Peripheral Participation*. University of Cambridge Press, 1991.
97. J. Lee, M. Kim, and Y. Lee. Information Retrieval Based on Conceptual Distance in IS-A Hierarchies. *Journal of Documentation*, 49:188–207, 1993.
98. V. Levenshtein. Binary Codes Capable of Correcting Deletions, Insertions, and Reversals. *Soviet Physics Doklady*, pages 707–710, 1966.
99. M. Li, B. Yu, O. Rana, and Z. Wang. Grid Service Discovery with Rough Sets. *IEEE TKDE*, 20(6):851–862, 2008.
100. Y. Li, A. Bandar, and D. McLean. An Approach for Measuring Semantic Similarity between Words Using Multiple Information Sources. *IEEE TKDE*, 15(4):871–882, 2003.
101. Y. Li, D. McLean, Z. Bandar, J. O'Shea, and K. Crockett. Sentence Similarity based on Semantic Nets and Corpus Statistics. *IEEE TKDE*, 18(8):1138–1150, 2006.
102. Y. Li, J. Tang, D. Zhang, and J. Li. Toward Strategy Selection for Ontology Alignment. In *Proc. of ESWC (Poster)*, 2007.
103. D. Lin. An Information-Theoretic Definition of Similarity. In *Proc. of Conf. on Machine Learning*, pages 296–304, 1998.
104. F.-R. Lin, K.-L. Tsai, and P.-C. Sun. The Design and Evaluation of P2P Transactive Memory System. In *Proc. of EEE '05*, pages 640–645, Washington, DC, USA, 2005. IEEE Computer Society.
105. A. Maedche, B. Motik, L. Stojanovic, R. Studer, and R. Volz. Ontologies for enterprise knowledge management. *IEEE Intelligent Systems*, 18(2):26–33, Mar–Apr 2003.

106. S. S. Maedche, A. KAON: The Karlsruhe Ontology and Semantic Web Meta Project. *Knstliche Intelligenz*, 3, 2003.
107. D. Maier and L. M. L. Delcambre. Superimposed Information for the Internet. In *Proc. of WebDB (Informal Proceedings)*, pages 1–9, 1999.
108. D. L. Martin, M. Paolucci, S. A. McIlraith, M. H. Burstein, D. V. McDermott, D. L. McGuinness, B. Parsia, T. R. Payne, M. Sabou, M. Solanki, N. Srinivasan, and K. P. Sycara. Bringing Semantics to Web Services: The OWL-S Approach. In *Proc. of SWSWPC*, pages 26–42, 2004.
109. J. McCarthy. Circumscription - A Form of Non-Monotonic Reasoning. *Artificial Intelligence*, 13(1-2):27–39, 1980.
110. C. Meilicke, H. Stuckenschmidt, and A. Tamilin. Repairing Ontology Mappings. In *Proc. of AAAI 2007*, pages 1408–1413, 2007.
111. S. Melnik, H. Garcia-Molina, and E. Rahm. Similarity Flooding: A Versatile Graph Matching Algorithm and Its Application to Schema Matching. In *Proc. of ICDE*, pages 117–128, 2002.
112. G. Miller. WordNet An On-line Lexical Database. *International Journal of Lexicography*, 3(4):235–312, 1990.
113. G. Miller and W. Charles. Contextual Correlates of Semantic Similarity. *Language and Cognitive Processes*, 6:1–28, 1991.
114. G. A. Miller. WordNet: A Lexical Database for English. *Commun. ACM*, 38(11):39–41, 1995.
115. G. Mitra, P. & Wiederhold. Resolving Terminological Heterogeneity in Ontologies. In *Proc. of Workshop on Ontologies and Semantic Interoperability*, 2002.
116. P. Mitra, N. F. Noy, and A. R. Jaiswal. OMEN: A Probabilistic Ontology Mapping Tool. In *Proc. of ISWC*, pages 537–547, 2005.
117. P. Mitra, G. Wiederhold, and J. Jannink. Semi-automatic Integration of Knowledge sources. In *Proc. of FUSION99*, 1999.
118. S. Montanelli and S. Castano. Semantically Routing Queries in Peer-based Systems: The H-link Approach. *Knowl. Eng. Rev.*, 23(1):51–72, 2008.
119. M. Nagy, M. Vargas-Vera, and E. Motta. DSSim-ontology Mapping with Uncertainty. In *Ontology Matching*, 2006.
120. R. Neches, R. Fikes, T. W. Finin, T. R. Gruber, R. S. Patil, T. E. Senator, and W. R. Swartout. Enabling Technology for Knowledge Sharing. *AI Magazine*, 12(3):36–56, 1991.
121. I. Nonaka. A Dynamic Theory of Organizational Knowledge Creation. *Organization Science*, 5(1):14–37, 1994.
122. N. F. Noy. Semantic Integration: A Survey Of Ontology-Based Approaches. *SIGMOD Record*, 33(4):65–70, 2004.
123. N. F. Noy, A. Chugh, W. Liu, and M. A. Musen. A Framework for Ontology Evolution in Collaborative Environments. In *Proc. of ISWC*, pages 544–558, 2006.
124. N. F. Noy and M. A. Musen. Anchor-PROMPT: Using non-local Context for Semantic Matching. In *Proc. of Ontologies and Information Sharing*, pages 63–70, 2001.
125. N. F. Noy and M. A. Musen. Evaluating Ontology-mapping Tools: Requirements and Experience. In *Proc. of OntoWeb-SIG3 Workshop at the 13th International Conference on Knowledge Engineering and Knowledge Management*, pages 1–14, 2002.
126. N. F. Noy and M. A. Musen. The Prompt Suite: Interactive Tools for Ontology Merging and Mapping. *International Journal of Human-Computer Studies*, 59:2003, 2003.
127. D. E. O'Leary. Using AI in Knowledge Management: Knowledge Bases and Ontologies. *IEEE Intelligent Systems*, 13(3):34–39, 1998.

128. A.-M. K. Pathan, J. A. Broberg, K. Bubendorfer, K. H. Kim, and R. Buyya. An Architecture for Virtual Organization (VO)-based Effective Peering of Content Delivery Networks. In *Proc. of UPGRADE '07*, pages 29–38, New York, NY, USA, 2007. ACM.
129. A. A. Patil, S. A. Oundhakar, A. P. Sheth, and K. Verma. Meteor-S Web Service Annotation Framework. In *Proc. of WWW '04*, pages 553–562, New York, NY, USA, 2004. ACM Press.
130. P. T. Patwardhan S. Using WordNet-based Context vectors to Estimate the Semantic Relatedness of Concepts. In *Proc. of EACL 2006*, pages 1–8, 2006.
131. T. Pedersen. Wordnet-similarity, http://talisker.d.umn.edu/cgi-bin/similarity/similarity.cgi.
132. T. Pedersen, S. V. S. Pakhomov, S. Patwardhan, and C. G. Chute. Measures of semantic similarity and relatedness in the biomedical domain. *Journal of Biomedical Informatics*, 40(3):288–299, 2007.
133. G. Pirr and D. Talia. Lom: a linguistic ontology matcher based on information retrieval. *Journal of Information Science*, 2008. to appear.
134. G. Pirro', M. Ruffolo, and D. Talia. SECCO: On Building Semantic Links in Peer to Peer Networks. *Journal on Data Semantics*, XII:to appear, 2008.
135. A. Poggi, D. Lembo, D. Calvanese, G. D. Giacomo, M. Lenzerini, and R. Rosati. Linking Data to Ontologies. *Journal on Data Semantics*, 10:133–173, 2008.
136. B. Popov, A. Kiryakov, A. Kirilov, D. Manov, D. Ognyanoff, and M. Goranov. KIM - Semantic Annotation Platform. In *Proc. of ISWC*, pages 834–849, 2003.
137. Y. Qu, W. Hu, and G. Cheng. Constructing virtual documents for ontology matching. In *Proc. of WWW*, pages 23–31, New York, NY, USA, 2006. ACM.
138. R. Rada, H. Mili, and M. Bicknell, E. andBlettner. Development and Application of a Metric on Semantic Nets. *IEEE Transactions on Systems, Man, and Cybernetics*, 19:17–30, 1989.
139. E. Rahm and P. A. Bernstein. A Survey of Approaches to Automatic Schema Matching. *VLDB Journal*, 10(4):334–350, 2001.
140. K. Ramamritham and P. J. Shenoy. Guest Editors' Introduction: Dynamic Information Dissemination. *IEEE Internet Computing*, 11(4):14–15, 2007.
141. S. Ravi and M. Rada. Unsupervised Graph-based Word Sense Disambiguation Using Measures of Word Semantic Similarity. In *Proc. of ICSC 2007*, 2007.
142. P. Resnik. Using Information Content to Evaluate Semantic Similarity in a Taxonomy. In *Proc. of IJCAI 1995*, pages 448–453, 1995.
143. E. L. Rissland. AI and Similarity. *IEEE Intelligent Systems*, 21:39–49, 2006.
144. J. L. Rodgers and W. A. Nicewander. Thirteen Ways to Look at the Correlation Coefficient. *The American Statistician*, 42(1):59–66, 1988.
145. M. Rodriguez and M. Egenhofer. Determining semantic similarity among entity classes from different ontologies. *IEEE TKDE*, 15(2):442–456, 2003.
146. H. Rubenstein and J. B. Goodenough. Contextual Correlates of Synonymy. *Commun. ACM*, 8(10):627–633, 1965.
147. O. D. Sahin, C. E. Gerede, D. Agrawal, A. E. Abbadi, O. H. Ibarra, and J. Su. SPiDeR: P2P-Based Web Service Discovery. In *Proc. of ICSOC*, pages 157–169, 2005.
148. G. Salton and M. McGill. *Introduction to Modern Information Retrieval*. Mc-Graw Hill, 1983.
149. G. Salton, A. Wong, and C. S. Yang. A Vector Space Model for Automatic Indexing. *Commun. ACM*, 18(11):613–620, 1975.
150. B. Schaeffer and R. Wallace. Semantic Similarity and the Comparison of Word Meanings. *J. Experiential Psychology*, 82:343–346, 1969.

151. M. Schlosser, S. Michael, D. Stefan, and N. Wolfgang. A Scalable and Ontology-Based P2P Infrastructure for Semantic Web Services. In *Proc of P2P '02*, page 104, Washington, DC, USA, 2002. IEEE Computer Society.
152. C. Schmidt and M. Parashar. A Peer-to-Peer Approach to Web Service Discovery. *World Wide Web*, 7(2):211–229, 2004.
153. A. Schwering. Hybrid Model for Semantic Similarity Measurement. In *Proc. of ODBASE05*, pages 1449–1465, 2005.
154. N. Seco. Computational Models of Similarity in Lexical Ontologies. Master's thesis, University College Dubin, 2005.
155. N. Seco, T. Veale, and J. Hayes. An Intrinsic Information Content Metric for Semantic Similarity in WordNet. In *Proc. of ECAI 2004*, pages 1089–1090, 2004.
156. G. Shafer. *A Matematical Theory of Evidence*. Princeton University Press, 1976.
157. C. Shannon. A Mathematical Theory of Communication. *Bell System Technical Journal*, 27:379–423, 1948.
158. R. N. Shepard. Toward a Universal Law of Generalization for Psychological Science. *Science*, 37(4280):1317–1323, 1987.
159. S. Staab, R. Studer, H.-P. Schnurr, and Y. Sure. Knowledge Processes and Ontologies. *IEEE Intelligent Systems*, 16(1):26–34, 2001.
160. I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications. In *Proc. of the 2001 conference on applications, technologies, architectures, and protocols for computer communications*, pages 149–160. ACM Press, 2001.
161. G. Stoilos, G. B. Stamou, and S. D. Kollias. A String Metric for Ontology Alignment. In *Proc. of ISWC*, pages 624–637, 2005.
162. G. Stumme, M. Ehrig, S. Handschuh, A. Hotho, A. Maedche, B. Motik, D. Oberle, C. Schmitz, S. Staab, L. Stojanovic, N. Stojanovic, R. Studer, Y. Sure, R. Volz, and V. Zacharias. The Karlsruhe View on Ontologies. Technical report, University of Karlsruhe, 2003.
163. X. Su and J. A. Gulla. An Information Retrieval Approach to Ontology Mapping. *Data Knowl. Eng.*, 58(1):47–69, 2006.
164. D. Talia and P. Trunfio. Toward a Synergy between P2P and Grids. *IEEE Internet Computing*, 7(4):94–95, 2003.
165. J. Tang, J. Li, B. Liang, X. Huang, Y. Li, and K. Wang. Using Bayesian Decision for Ontology Mapping. *Journal of Web Semantics*, 4(4):243–262, 2006.
166. R. Tous and J. Delgado. A Vector Space Model for Semantic Similarity Calculation and OWL Ontology Alignment. In *Proc of DEXA*, pages 307–316, 2006.
167. B. Traversat, A. M., and E. Pouyol. Project JXTA: A Loosely-Consistent DHT Rendezvous Walker, March 2003. `http://www.jxta.org`.
168. P. Trunfio, D. Talia, H. Papadakis, P. Fragopoulou, M. Mordacchini, M. Pennanen, K. Popov, V. Vlassov, and S. Haridi. Peer-to-Peer resource discovery in Grids: Models and systems. *Future Generation Comp. Syst.*, 23(7):864–878, 2007.
169. E. Tsui. Technologies for Personal and Peer to Peer Knowledge Management. Technical report, CSC Leading Edge Forum Technology, 2002.
170. A. Tversky. Features of Similarity. *Psychological Review*, 84(2):327–352, 1977.
171. A. Updegrove. The Semantic Web: An Interview with Tim Berners-Lee. *Consortium Standard Bullettin*, 4(6), 2005.
172. G. Varelas, E. Voutsakis, P. Raftopoulou, E. G. Petrakis, and E. E. Milios. Semantic Similarity Methods in WordNet and their Application to Information Retrieval on the Web. In *Proc. of WIDM '05*, pages 10–16, New York, NY, USA, 2005. ACM.

173. L.-H. Vu, M. Hauswirth, and K. Aberer. Towards P2P-based Semantic Web Service Discovery with QoS Support. In *Proc. of BPS 2005*, 2005.
174. H. Wache, T. Vögele, U. Visser, H. Stuckenschmidt, G. Schuster, H. Neumann, and S. Hübner. Ontology-based Integration of Information a Survey of Existing Approaches. In H. Stuckenschmidt, editor, *Ontologies and Information Sharing*, pages 108–117, 2001.
175. C. Wang, L. Xiao, L. Yunhao, and P. Zheng. DiCAS: An Efficient Distributed Caching Mechanism for P2P Systems. *IEEE Trans. Parallel Distrib. Syst.*, 17(10):1097–1109, 2006. Student Member-Chen Wang and Member-Li Xiao and Member-Yunhao Liu and Member-Pei Zheng.
176. Y. Wang, W. Liu, and D. A. Bell. Combining Uncertain Outputs from Multiple Ontology Matchers. In *Proc. of SUM*, pages 201–214, 2007.
177. Z. Wang, M. Kumar, S. K. Das, and H. Shen. File Consistency Maintenance Through Virtual Servers in P2P Systems. In *Proc. of ISCC '06*, pages 435–441, Washington, DC, USA, 2006. IEEE Computer Society.
178. Z. Wang, Y. Wang, S. Zhang, G. Shen, and T. Du. Matching Large Scale Ontology Effectively. In *Proc. of ASWC*, pages 99–105, 2006.
179. H. Wei, N. Jian, Y. Qu, and Y. Wang. GMO: A Graph Matching for Ontologies. In *Proc. of K-Cap 2005 Workshop on Integrating Ontologies 2005*, pages 43–50, 2005.
180. W. E. Winkler. The State of Record Linkage and Current Rsearch Problems. Technical report, Statistical Research Division, U.S. Census Bureau, 1999.
181. K. Wolstencroft, P. Alper, D. Hull, C. Wroe, P. Lord, R. Stevens, and C. Goble. The (my)Grid Ontology: Bioinformatics Service Oiscovery. *Int J Bioinform Res Appl.*, 3(3):303–325, 2007.
182. B. Yang and H. Garcia-Molina. Designing a Super-Peer Network. In *Proc. of ICDE*, pages 49–60, 2003.
183. L. Yi, J.-Z. Li, D. Zhang, and J. Tang. Result of Ontology Alignment with RiMOM at OAEI'06. In *Proc. of Ontology Matching*, 2006.
184. Z. Yingwu and H. Yiming. Efficient Semantic Search on DHT overlays. *J. Parallel Distrib. Comput.*, 67(5):604–616, 2007.
185. A. Zavaracky. Glossary-Based Semantic Similarity in the WordNet Ontology. Master's thesis, University College Dublin, 2003.