IMT Institute for Advanced Studies, Lucca

Lucca, Italy

Ontology-Driven Knowledge Discovery

PhD Program in Computer Science and Engineering

XXI Cycle

By

Barbara Furletti

2009

The dissertation of Barbara Furletti is approved.

Program Coordinator: Prof. Ugo Montanari, University of Pisa

Supervisor: Prof. Franco Turini, University of Pisa

The dissertation of Barbara Furletti has been reviewed by:

Dr. Hans-Ulrich Krieger, DFKI - Language Technology Lab. Saarbrucken - Germany

Prof. Bettina Berendt, Department of Computer Science - Catholic University of Leuven - Belgium

IMT Institute for Advanced Studies, Lucca

2009

To the important people of my life.

Contents

List of Figures

List of Tables

List of Abbreviations

AI	Artificial Intelligence
AR	Association Rule
BCM	Bayesian Causal Map
BN	Bayesian Network
BPA	Business Plan Analyser
DB	Database
DBMS	Database Management System
DL	Description Logic
DM	Data Mining
ER	Expert Rule
FOL	First Order Logic
IDE	Integrated Development Environment
ILP	Inductive Logic Programming
IR	Influence Rule
KD	Knowledge Discovery
KDD	Knowledge Discovery in Databases

KR	Knowledge Representation
LP	Logic Programming
MRDM	Multi-Relational Data Mining
NLR	Non-Linear Rating
ODCA	Ontology-Driven Conceptual Analysis
OWL	Web Ontology Language
RDF	Resource Description Framework
RDFS	Resource Description Framework Schema
SLA	Service Level Agreement
SME	Small Medium Enterprise
SN	Social Network
SNA	Social Network Analysis
SQL	Structured Query Language
SVM	Support Vector Machine
URI	Uniform Resource Identifiers
XSD	XML Schema Document

Acknowledgements

I would like to express my deep and sincere gratitude to my supervisor, Prof. Franco Turini. His teachings, suggestions and his experience have been of great value to me. His guidance was essential for the success of the work.
I would like to thank Prof. Dino Pedreschi and Prof. Alessandro D'Atri for having supervised my progress these past three years and for having contributed to the improvement of my the work.
I am grateful to Dr. Hans-Ulrich Krieger and Prof. Bettina Berendt for having accepted to be the reviewers of this Ph.D. Thesis and for their detailed reviews, constructive criticisms and excellent advice.
I have to thank the MUSING project for having provided a useful and convenient framework, and to all the partners that worked with me and that supplied hints for concluding my work.
Thanks to Prof. Laura Fatuzzo, to all my colleagues for their collaboration, and to my friends for having made my days happy.
A special thanks to my family that is always present.
Finally, my best and warmest thanks are for Francesco who shared with me the whole experience with his mind and his heart. Loving thanks to him for his great enthusiasm and his patience, and for having placed his knowledge and his experience at my disposal.

This work is especially for him, and for all the people that believed in it.

If you are not sure what to do, make something – Paul Graham

Vita

December 3, 1976	Born, Livorno, Italy
1995	High School Certificate Commercial School Final mark: 60/60 Istituto Tecnico Commerciale "C. Cattaneo", Cecina (LI), Italy
February 2003	Degree in Computer Science Final mark: 104/110 University of Pisa Pisa, Italy
March 2003	Master Degree in Computer Science Final mark: 103/110 University of Pisa Pisa, Italy
Nov. 2003 - Apr. 2006	Collaboration at the Department of Computer Science University of Pisa Pisa, Italy
May 2006 - Feb. 2009	Fellowship grant at the Department of Computer Science University of Pisa Pisa, Italy
Mar. 2009 - Feb. 2010	Collaboration at the Department of Computer Science University of Pisa Pisa, Italy

Publications

1. M. Baglioni, B. Furletti, F. Turini. "DrC4.5: Improving C4.5 by means of Prior Knowledge", in *Proceedings of the 2005 ACM Symposium on Applied Computing*, Vol 1, pp. 474 – 481, 2005.
2. M. Baglioni, B. Furletti, F. Turini. "A Tool for Economic Plans analysis based on expert knowledge and data mining techniques", in *Proceedings of The IADIS International Conference - Applied Computing 2006*, pp 421 – 426, 2006.
3. F. Turini, M. Baglioni, B. Furletti, S. Rinzivillo. "Examples of Integration of Induction and Deduction in Knowledge Discovery", in *O. Stock, M. Schaerf (Eds.) , Reasoning, Action and Interaction in AI Theories and Systems*, LNAI 4155, pp. 307 – 326, 2006.
4. A. Bellandi, B. Furletti, V. Grossi, A. Romei. "Pushing Constraints in Association Rule Mining: An Ontology-Based Approach", in the *Proceedings of the IADIS International Conference WWW/INTERNET 2007*, 2007.
5. A. Bellandi, B. Furletti, V. Grossi, A. Romei. "Ontology-driven association rules extraction: a case of study", in the *Proceedings of the Workshop "Context and Ontologies: Representation and Reasoning 2007"*, pp. 5 – 26, 2007.
6. M. Baglioni, A. Bellandi, B. Furletti, C. Pratesi, F. Turini. "Improving the Business Plan Evaluation Process: the Role of Intangibles", in *International Journal Quality Technology and Quantitative Management - Special Issue on "Non-Standard Analysis of Customer Satisfaction Survey Data"*, to be published in March 2010.
7. A. Bellandi, B. Furletti, V. Grossi, A. Romei. "Ontological Support for Association Rule Mining", in *the proceedings of the IASTED International Conference on Artificial Intelligence and Applications (AIA 2008)*, 2008.
8. F. Fornasari, B. Furletti, C. Montanari. "An Extensible and Interactive Software Agent for Mobile Devices based on GPS Data", in the *Proceedings of the IADIS International Conference on Applied Computing 2008*, 2008.
9. D. Bacciu, A. Bellandi, B. Furletti, V. Grossi, A. Romei. "Discovering Strategic Behaviour in Multi-Agent Scenarios by Ontology-Driven Mining", in *Advances in Robotics, Automation and Control*, IN-TECH publisher, pp. 171 – 198, 2008.

Presentations

1. B. Furletti, "A Tool for Economic Plans analysis based on expert knowledge and data mining techniques", at *The IADIS International Conference - Applied Computing 2006*, San Sebastian, Spain. 25-28 February 2006.
2. B. Furletti, "Log analysis and monitoring ontology", at *BRITE Project: Joint SC1/SC2 Meeting*, Brussels, Belgium. 1-2 March 2007.
3. M. Baglioni, B. Furletti, "Mining ontologies and classification of business plans", at *MUSING Project: Foundation Meeting*, Sheffield, UK. 19 March 2007.
4. M. Baglioni, B. Furletti, "Analizzatore di Documenti di Business: Il Sistema", at *TetraModel Project: Final Review*, Pisa, Italy. 20 April 2007.
5. A. Bellandi, B. Furletti, "The Event Ontology - first definition", at *BRITE Project: SC1 and SC2 Workshop*. Stockholm, Sweden. 31 May 2007.
6. B. Furletti, "The Business Plan Analyser ver. 1.0", at *MUSING Project: 6^{th} Plenary Financial Risk Management Meeting*, Brussels, Belgium. 12 September 2007.
7. B. Furletti, F. Turini, "Pilot B2.4 Biz Plan Analyser" at *MUSING Project: 7^{th} Plenary Financial Risk Management Meeting*, Siena, Italy. 13 December 2007.
8. A. Bellandi, B. Furletti, "Ontological Support for Association Rule Mining", at *Artificial Intelligence and Applications (AIA 2008) Conference*. Innsbruck. 12 February 2008.
9. B. Furletti, "Pilot B2.4 - Business Plan Analyser", at *MUSING Project: 8^{th} Plenary Financial Risk Management Meeting*, Milan. 1 April 2008.
10. B. Furletti, "Mining Influence Rules out of Ontologies", at *MUSING Project: Foundation Meeting*, Munich, Germany. 8 May 2008.
11. B. Furletti, "Discovering suspicious behaviour by means of direct queries the the ontology", at *BRITE Project: Technical meeting*, Pisa, Italy. 16 May 2008.
12. B. Furletti, S. Miniati "Pilot B2.4 - Business Plan Analyser: Technical Developments", at *MUSING Project: 10^{th} Plenary Financial Risk Management Meeting*, Pavia, Italy. 19 June 2008.
13. B. Furletti, "Mining Influence Rules out of Ontologies", at *Workshop - University of Pisa, INTEX company, Metaware company*, Pisa, Italy. 9 July 2008.

14. B. Furletti,"Knowledge extraction from ontology - PhD thesis on going work", at *Pisa Lab Workshop*, Pisa, Italy. 12 September 2008.
15. A. Bellandi, B. Furletti, "Ontology and reasoning technologies', at *MUSING Project: 2^{nd} Integration Meeting*, Saarbrucken, Germany. 21 November 2008.
16. B. Furletti, S. Miniati, "Pilot B2.4 - The prototype", at *MUSING Project: 11^{th} Plenary Financial Risk Management Meeting*, Milan, Italy. 11 December 2008.
17. B. Furletti, "Ontology and reasoning technologies", at *MUSING Project: 3^{rd} Integration Meeting*, Milan, Italy. 12 December 2008.
18. B. Furletti, "Ranking Ontologies: problems and solutions", at *University meeting*, Pisa, Italy. 25 February 2009.
19. B. Furletti, "Pilot B3.1 - On line Self-Assessment", at *MUSING Project: 13^{th} Plenary Financial Risk Management Meeting*, Brussels, Belgium. 31 May 2009.
20. B. Furletti, "Pilot 3.1 - Ontology Self Assessment", at *MUSING Project: Integration Meeting*, Saarbrucken, Germany. 7 April 2009.
21. B. Furletti, F. Turini,"Online Self Assessment: theoretical and technical aspects", at *Unilateral Meeting - Monte Paschi Siena, University of Pisa, University of Pavia*, Siena, Italy. 31 July 2009.

Abstract

The problem of Knowledge Discovery has always attracted many researchers and continues to be of great relevance to the computer science community in the branch of learning. This thesis aims to contribute to this topic, getting hints from the *Ontology* and *Data Mining* environments.
We investigate a method for extracting new implicit knowledge directly from an ontology by using an inductive/deductive approach. By giving a sort of *Bayesian* interpretation to relationships that already exist in an ontology, we are able to return the extracted knowledge in form of *Influence Rules*.

The idea is to split the extraction process in two separate phases by exploiting the ontology peculiarity of keeping metadata (the schema) and data (the instances) separate. The deductive process draws inference from the ontology structure, both concepts and properties, by applying link analysis techniques and producing a sort of implications (rules schemas) in which only the most important concepts are involved. Then an inductive process, realized by a data mining algorithm, explores the ontology instances for enriching the implications and building the final rules.

A final rule has a form like $<premise \xrightarrow{w} consequence>$ where premise and consequence refer to the class names, and values to their datatype properties, while w, the weight, measures the strength of the influence.
An example of a final rule is:
$Manager.hasAge < 45 \xrightarrow{0.80} Project.hasDegreeOfSuccess = good$.
This can be read as, in 80% of the cases, whenever a manager of a company is less then 45 years old, then the project he manages has a *good* degree of success.

What we want to prove, besides the correctness and feasibility[1] of the project, is that the approach allows us to extract "higher level" rules w.r.t. classical knowledge discovery techniques. In fact, ontology metadata gives a general view of the domain of interest and supplies information about all the elements apart from the fact that they are included as instances in the collected data. The technique is completely general and applicable to each domain. Since the output is a set of "standard" Influence Rules, it can be used to integrate existing knowledge or for supporting any other data mining process.

The thesis includes the following chapters:
Chapter 1 contains a brief introduction of the work, focusing on the main questions that have to be addressed.
Chapter 2 offers an overview of the context of research in which the thesis is part of: data mining and ontologies.
Chapter 3 explores the literature dealing with the open questions raised in chapter 1.
Chapter 4 is the core section; it discusses the proposed solutions and presents all the phases of the extraction process as well as the algorithms and the proofs.
Chapter 5 describes an application of the methodology in the context of MUSING, a European project in the field of business intelligence.
Chapter 6 ends this thesis with final considerations and future possible works.

[1] The term feasibility, here and in the rest of the thesis, has to be intended as the "capability of being done".

Chapter 1

Introduction

We begin to speak about databases (DBs) as repositories of data beginning in the late 1960s when E.F. Codd (Cod70) and his research group at IBM labs applied some mathematical principles and predicate logic to the field of data modelling. Since then, DBs and their evolutions have been used as a source of information to query and manipulate data. But DBs, seen as single static tables, have evolved very quickly becoming actual systems for data management (the Database Management Systems and Relational Database Management Systems) (AGO97; ACPT99), including a collection of programs and tools for storing, modifying, and extracting information.

Knowledge extraction from DBs can be made directly by using a query language or, indirectly by means of inductive methods. In the former case we essentially carry out a data retrieval process, while in the latter higher-level knowledge is mined i.e., association rules, clusters, trees.
In 1974, still at IBM labs, the first language for DB was developed. SEQUEL (Structured English Query Language) (CB74), later called SQL for copyright issues, was the forerunner of all the query languages becoming the standard[1] for relational DB. Despite many efforts in this direction, now only the core of the SQL, the so-called Entry Level has been main-

[1]The IBM dialect of SQL become an ANSI standard in 1986 and one year later also standard ISO.

tained. Each company selling DBMS extended the standard SQL, creating a proprietary query language i.e., Oracle uses PLSQL, SQLServer uses T-SQL and the IBM System now uses DB2.
In any case, the querying process returns a set of data possibly aggregated or projected in several ways.

Using Knowledge Discovery techniques we can, instead, analyse the rough data and extract new unknown implicit knowledge i.e., associations among data, similarities of data (Qui86; HK00; HMS01). Knowledge discovery represents a real non-trivial process for identifying patterns that are new, valid, useful and understandable, starting from a set of data. Depending on the objectives, the final knowledge can be a new result, a prediction or a confirmation about consolidated facts and theories.

In recent years, with the advent of Web 2.0 and the Semantic Web era, ontologies have became important, replacing the traditional storing systems in many applications (SHB06). We can say that they now represent the new technology for knowledge representation, data storage and information sharing (Bie03; Smi; Smi03).
As with the DBs, ontologies are also equipped with query languages that permit one to retrieve information. As expected, the ontology query languages are implemented in ontology query systems. These systems, that correspond to the DBMSs for the DBs, are frameworks that provide several "tools" such as reasoner engines, languages for querying and languages for defining the rules. Some of these query languages, for example SPARQL (PS08), resemble the SQL syntax. Even in this field, the most interesting knowledge is extracted by using a reasoner (Jena2[2], RACER[3], Pellet[4] are examples of most popular reasoners), that is, an engine able to infer logical consequences from a set of asserted facts or axioms.
A reasoner makes implicit knowledge explicit by implementing a sort of decision procedure which starts from a set of logical axioms and finds the relations between them and whether (or not) they are satisfiable. From

[2]Jena2: `http://jena.sourceforge.net/inference/`

[3]RACER - Renamed Abox and Concept Expression Reasoner: `http://www.sts.tu-harburg.de/~r.f.moeller/racer/`

[4]Pellet: `http://clarkparsia.com/pellet/`

this point of view, the ontology is the collection of axioms (relations, definitions and constraints).

The question that we ask ourselves and try to answer with this thesis, is about the possibility of extracting further knowledge from ontologies, besides the one obtained by using the traditional reasoning systems.
The answer is yes, and with respect to traditional methods, our method gives a Bayesian interpretation to the relationships that already exist in an ontology. Traditional reasoners, in fact, perform deductive reasoning for extracting "hidden" facts that are true under every interpretation. Instead, the approach we propose, adopts a probabilistic interpretation and returns a set of weighted Influence Rules (IRs).

The thesis deals with the idea of combining ontologies and Data Mining techniques in a novel way. We investigate a method for extracting new implicit knowledge starting from the ontology schema rather than from the instances. To realize it, we adopt essentially a deductive/inductive approach, and we take hints and inspiration from graph theory (Die00; CMH03; WM03), link analysis (AP05; FLM+06; Kle98) and traditional data mining techniques (BGL+06; BL04; BL05). The idea is to explore the T-Box first, and then to integrate the extracted information by mining the A-Box. The T-Box represents the ontology structure and is composed by concepts (classes or types of instances), roles (built-in predicates), and features (attributes/properties) of the instances, while the A-Box represents the instances, that is, the assertions about individuals (relation instances). The output of this analysis is a set of IRs, e.g. a set of weighted implications between ontology concepts.

What we would like to prove is the feasibility of the technique and also show that the extracted knowledge is at a "higher level" since it abstracts from the instances but is domain-dependent at the same time. In this way, the hidden information can be considered in a general way, independent of the representation/description given by the data. In our case, the ontology schema is the "formalism" for representing the knowledge: it is our Knowledge Representation (KR) formalism. It aims at representing the world, objects and relations, and permits us to have a point of view at a higher level. A further objective is to see how this high level

knowledge, extracted from the ontology, can improve the inductive construction of classification models.
The general idea is summed up by figure 1, pointing out the merging of the deductive and inductive approach.

The deductive step operates on the ontology schema (T-Box) by ex-

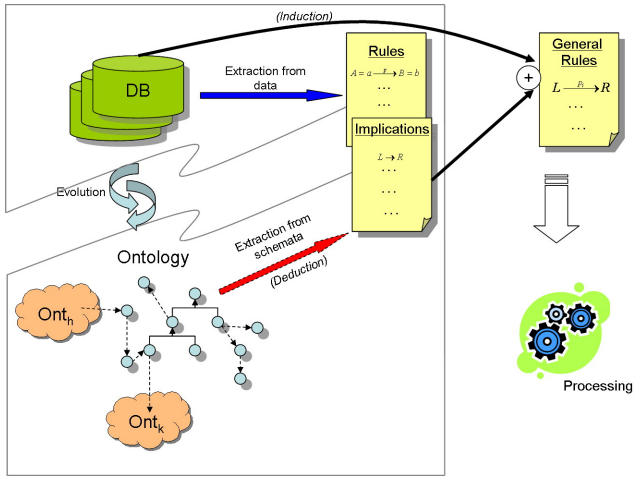


Figure 1: Knowledge extraction from the ontology schema.

ploring concepts and object properties. For this purpose, graph theory and link analysis techniques have been used. The result is a set of rule schemas that have the form of classical implications:

$$L \rightarrow R,$$

which means that when the left hand side L holds, then the right hand side R holds too.
Nevertheless, the instances are a fundamental component of our approach: they are used in the second phase not for extracting additional rules but for characterizing the deducted implications providing them with values for the variables, and a measure for the importance of the rules. This action that realizes the inductive step has been put in place by applying a data mining technique. We can re-write the previous implication as a rule

in a stricter sense:

$$L \xrightarrow{p_i} R,$$

where p_i is the strength by which L implies R.
The IRs are general rules among concepts of the domain, thus can be used for supporting any other (data mining) process or in other applications for enriching the existing knowledge (processing task).

The methodology, described and developed in the thesis, has been customized and tested inside the European project MUSING (Mus06). MUSING (MUlti-industry, Semantic-based next generation business INtelliGence) aims at developing a new generation of Business Intelligence tools and modules based on semantic-based information and content systems. It integrates Semantic Web and Human Language technologies and combines declarative rule based methods and statistical approaches for enhancing the technological foundations of knowledge acquisition and reasoning in BI applications. In MUSING, the ontologies have been adopted as a means for representing the knowledge and as a source of information. Furthermore, several analysis tools of inductive and deductive type have been developed. MUSING has been a good testing environment and a valuable source of data for our system. In particular, the knowledge extraction process has been applied to a subset of the MUSING ontology, and the IRs extracted have been used for enriching and integrating the expert knowledge.

The proposal is part of an interesting research sector that tries to answer many relevant questions. In this thesis, we intend to answer some of these questions and to present the results of our research work.
We organized the thesis in a modular and hopefully original way, exploiting the multi-disciplinary nature of the work, and sometimes digressing into some historical and interesting facts. The state of the art has been dived in two parts. In chapter 2 we describes essentially the two main research areas, giving background information on data mining and ontologies, while in chapter 3 we analyse recent results and articles whose

topics are closely related to those of our thesis and from which we take inspiration. Due to this particular organization, part of our work is still shown in these first chapters, but an exhaustive and detailed discussion is given in chapter 4, the core chapter, where we describe in depth algorithms and strategies. The last two chapters contain a case of study and the conclusions. Below is a detailed summary of each chapter.

Chapter 2 presents an overview of the two main research areas in which the proposed approach finds its basis: data mining and ontologies. In the first part, basic notions of the knowledge discovery field are provided. In the second part, we introduce the ontologies starting from its philosophical origins up to the "modern", or computer science oriented, conceptualization.

Chapter 3 analyses thoroughly the various aspects of the proposal. It is presented in the form of open questions which we try to answer taking inspiration from the literature and the state-of-the-art main techniques for knowledge discovery and ontology analysis. We tried to create and maintain a certain correspondence among raised problems, questions and phases of the extraction process. Furthermore, we make comparisons with other, both recent and consolidated, representation formalisms or analysis methods.

Chapter 4 is the core of the thesis. After having provided the reader with sufficient background, we explain, step-by-step, our methodology showing both the theoretical and technical details including specifications, theorems, formulas and pseudo codes, all supported by a simple but useful example.

Chapter 5 describes the case study which is an actual application of our methodology in the context of MUSING (Mus06). We present the domain of application and a general overview of the project, the requirements, the available tools, its objectives and a short description of the developed components. The description is then focused on the component we are directly involved with, and on how the ontology extraction process has been customized and applied. Finally, an exhaustive discussion of the results is provided.

Chapter 6 concludes this work summing up the objectives, the main

steps and discussing the obtained results. Some proposals for future extensions and improvements are then suggested.

Appendix A provides some technical details about the implementation of the system while Appendix B provides additional information.

Chapter 2

Contextualizing the Work

Data mining and reasoning are the two main research areas in which our work bases its theoretical foundations. In this chapter we dedicate two separate sections for introducing Knowledge Discovery (KD) and Ontology.

2.1 Knowledge Discovery: a Brief Introduction

Knowledge Discovery in Databases (KDD) is focused on the development of methodologies and techniques that "make sense" out of data, i.e. for extracting relevant and non-trivial information from rough data. The phrase "knowledge discovery in databases" was coined at the first KDD workshop in 1989 to emphasize that knowledge is the end-product of a data-driven discovery. Formally, KDD is defined as:

> *the non trivial process for identifying patterns that are new, valid, useful and understandable, starting from a set of data.*

In this context, data are a set of facts i.e. cases in a database, while a pattern is an expression in some language describing a subset of the data or a model applicable to the subset. KDD is thus a sequence of steps that, starting from rough data, leads to the discovery of knowledge. Depending on the objectives, the final knowledge can be a new result or the

discovery of unknown information, a prediction or a confirmation about consolidated facts and theories. One of the most complete and exhaustive

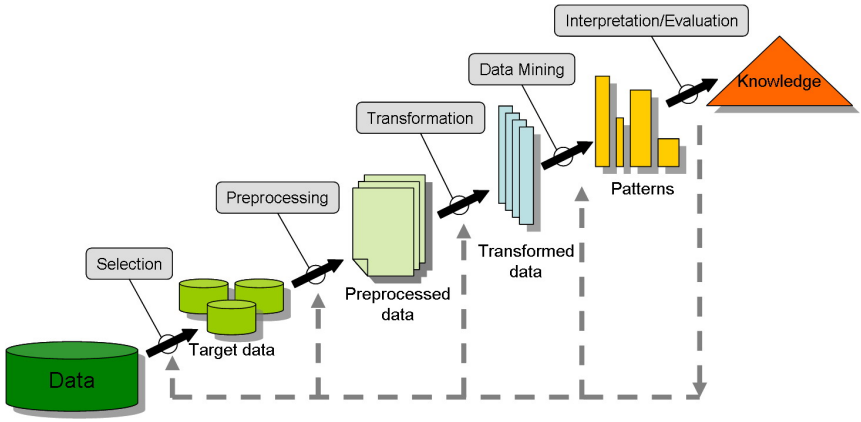


Figure 2: KDD Process.

definitions of the KDD process presented in the literature was provided by Shapiro in 1996 (PSFS96). Shapiro's definition (depicted in figure 2) highlights the following steps:

1. **Developing an understanding of the application domain**. This step aims at identifying the goal of the KDD process from the customer's point of view.

2. **Creating a target data set**. It is for selecting a data set, or focusing on a subset of variables or data samples, on which a discovery must be performed.

3. **Data cleaning and preprocessing**. It is a basic operation that filters the data from incorrect, unnecessary information or noise. It includes removing noise, collecting the necessary information to model or account for noise, deciding on strategies for handling missing data fields, and accounting for time-sequence information and known changes.

4. **Data reduction and projection**. It is for finding useful features to represent the data depending on the goal of the task. With dimensionality reduction or transformation methods, the effective number of variables under consideration can be reduced, or invariant representations for the data can be found.

5. **Matching**. It matches the goals of the KDD process (step 1) with a particular data mining method i.e. summarization, classification, regression, clustering, ...

6. **Exploratory analysis and model and hypothesis selection**. It chooses the data mining algorithms and selects methods to be used for searching data patterns. This process includes the task of deciding which models and parameters might be appropriate, and matching a particular data mining method with the overall criteria of the KDD process. For example, the end user might be more interested in understanding the model than its predictive capabilities.

7. **Data mining**. It searches for patterns of interest in a particular representational form or a set of such representations including classification rules or trees, regression, and clustering. The user can significantly aid the data mining method by correctly performing the preceding steps.

8. **Interpretation of the mined patterns**. It aims at interpreting the mined patterns, possibly returning to any of steps 1 through 7 for further iterations. This step can also involve visualization of the extracted patterns and models or visualization of the data given the extracted models.

9. **Activities over the discovered knowledge**. It uses the knowledge directly, incorporating the knowledge into another system for further action, or simply documenting it and reporting it to interested parties. This process also includes checking for and resolving potential conflicts with previously believed (or extracted) knowledge.

The KDD process can involve significant iterations and can contain loops between any two steps. Most of the previous work on KDD was focused on step 7, the data mining (HK00), but the other steps are indeed important, and probably more so, for the success of the KDD application. KDD is considered an interdisciplinary process that binds in a strong way various research fields such as machine learning, pattern recognition, database, statistics, artificial intelligence, knowledge acquisition, expert systems and high-performance computing. KDD processes and data mining techniques were first used in the scientific field. At the beginning of the 1990s, data mining was used in astronomy for analyzing large quantities of data. In this context, scientists realized images analysis systems based on data mining techniques (i.e. classification and clustering), for classifying images and objects present in space.
Some primary research and application challenges for KDD include:

- **Massive datasets and high dimensionality.** These datasets create combinatorial explosive search spaces for model induction. Possible solutions include efficient algorithms, sampling, approximation methods, massive parallel processing, dimensionality reduction techniques, and incorporation of prior knowledge.

- **User interaction and prior knowledge.** Since the KDD process is, by definition, interactive and iterative, it is a challenge to provide a high-performance, rapid-response environment that also assists users in the proper selection and matching of appropriate tools and techniques to achieve their goals. There needs to be more emphasis on human-computer interaction and less emphasis on total automation, with the aim of supporting both expert and "novice" users. Many current KDD methods and tools are not truly interactive and do not easily incorporate prior knowledge, about a problem except in simple ways. For example, Bayesian approaches use prior probabilities over data and distributions as one way of encoding prior knowledge while others employ deductive database capabilities to discover knowledge that is then used to guide the data mining search.

- **Overfitting and assessing statistical significance.** Overfitting is caused when an algorithm searches for the best parameters for one particular model using a limited set of data. Possible solutions include cross-validation, regularization, and other sophisticated statistical strategies. Simple methods to handle this problem include adjusting the test statistic as a function of the search and randomization testing.

- **Missing data.** This problem is massively present in business databases where important attributes can be missing if the database is not well designed. Missing data can result from operator error, actual system and measurement failures, or from a revision of the data collection process over time e.g., new variables are measured, but they were considered unimportant a few months before. Possible solutions include greater sophisticated statistical strategies in order to identify hidden variables and dependencies.

- **Understandability of patterns.** In many applications, it is important to make the discoveries more understandable to humans. Possible solutions include graphical representations, rule structuring, natural language generation, and techniques for visualization of data and knowledge. Rule refinement strategies can also help to address a related problem: discovered knowledge may be implicitly or explicitly redundant.

- **Managing changing data and knowledge.** Rapidly changing (non-stationary) data may make previously discovered patterns invalid. In addition, the variables measured in a given application database may be modified, deleted, or augmented with new measurements over time. Possible solutions include incremental methods for updating the patterns and treating change as an opportunity for discovery by using it to drive the search of the patterns.

- **Integration.** Integration issues include integration with a Database Management System (DBMS) (e.g., via a query interface), integration with visualization tools, and accommodation of real-time sen-

sor readings. Highly interactive human-computer environments as outlined by the KDD process permit both human-assisted computer discovery and computer-assisted human discovery. Development of tools for visualization, interpretation, and analysis of discovered patterns is of fundamental importance. Such interactive environments can enable practical solutions to many real-world problems far more rapidly than humans or computers operating independently.

A very attractive application field is Business Analysis (marketing, finance, fraud detection, ...) where the KDD process is used as "decision support" for making previsions and for wide range analysis. In the marketing field during the 90s, Agrawal opened a very important research stream, the so called "market basket analysis" (AIS93; AS95). In the finance and investment fields, the KDD gives a strong support to decision makers in the hard task of granting credits, and to managers in the task of self-assessment. In this context, many applications move within the sphere of action of the *Basel II International Agreement*[1], that enforce the use of analysis tools for identifying and evaluating the credit risk and for validating the evaluations.

By getting ideas from our experience in KDD, we have also contributed to this area in the past. We have, in fact, investigated a system for improving the discovery process by means of domain rules. Our focus was to classify business documents that described innovative projects in order to make previsions of success/failure (feasibility) of new submitted plans, and to realize a sort of (self) assessment (BFT06; KMS02). In our case, rules provided by experts in the economic and business fields have

[1]The Basel II Framework describes a more comprehensive measure and minimum standard for capital adequacy that national supervisory authorities are now working to implement through domestic rule-making and adoption procedures. It seeks to improve on the existing rules by aligning regulatory capital requirements more closely to the underlying risks that banks face. In addition, the Basel II Framework is intended to promote a forward-looking approach to capital supervision, one that encourages banks to identify the risks they may face today and in the future, and to develop or improve their ability to manage those risks. As a result, it is intended to be more flexible and better able to evolve with advances in markets and risk management practices [`http://www.bis.org/publ/bcbsca.htm`].

been formalized by means of Bayesian Causal Maps, a formalism that combines Causal Maps and Bayesian Networks (Hec96). Historically, Causal Maps have been introduced for the formalization and the interpretation of human intents and human thinking through relations among concepts. By adding the concepts of Bayesian Networks (BNs) to Causal Maps we get Bayesian Causal Maps, which enrich the former by introducing a probability by which a concept may imply a related one. The classification strategy, instead, was based on the well-known Quinlan's C4.5 classification trees algorithm (Qui93). For this purpose, we developed a new algorithm, DrC4.5[2], in which we drive the construction of classification trees by using the domain rules (BFT05). The carried out system is an example of how classical techniques can be improved by means of a-priori or domain knowledge. In particular, we obtained satisfactory results concerning the correctness of classification (higher number of new instances correctly classified) and the size of the models (trees with less leaves), compared with the C4.5 algorithm. So, in the cases where the classification accuracy was the same as for C4.5, we generated trees that reduced the over-specialization of the models.

DrC4.5 is cited here because it is the first prototype algorithm to inspire the implementation of YaDT-DRb (Bel07), the algorithm, based on YaDT (Rug04), that we use in the MUSING project for developing an analytical tool. It will become more clear later, especially during the description of the case study in chapter 5, why, in this section, we make reference to these particular works on Business Analysis, and what are the relationships with the work developed in this thesis. It is nevertheless useful, to note that this section focuses on a particular application of the thesis in order to extend and upgrade the system analysis on business documents.

[2]DrC4.5 stands for "Domain Rules C4.5".

2.2 Ontologies: State of the Art

In the last 10 years, within the computer science community, the word "ontology" has immediately recalled "technical" concepts such as graphs, semantic reasoning and knowledge representation. For example, a developer, can rely on the wide range of specification languages and tools for representing, editing and analysing such knowledge. Maybe theoretical researcher knows, instead, that the notion of ontology originates from a more remote past far from the modern and technological world of computers and internet.

By looking at various literatures, we find many correct definitions of ontology that highlight its theoretical and technical aspects. It will be very useful to look at these ontologies though their formal definitional language, as a representational model (conceptual diagram) or through the software programs that implement them.

In this section, we present a general view of the genesis and the state of the art of the ontologies as well as some significant examples of applications.

2.2.1 From Philosophy ...

In short, "ontology" may be defined as *the study of being as such*. The introduction of the Latin word "ontologia" was an attempt to modernize the classical debate about metaphysics. The term metaphysics was used initially by Aristotle's students to refer to what Aristotle himself called "first philosophy" (ØAS05). Aristotle's view of ontology is often referred to as the realist view. In his work *"Categories"* (AriCE) he provides a list of categories, which can be seen as an inventory of what there is in terms of the most general kinds of entities. These categories can be used to differentiate things as well as to refine specific aspects of things. Since these categories are one of the first recognized ontology, it could be interesting to show them.

Substance (e.g., man, horse)
Quantity (e.g., two cubits long, three cubits long)

Quality (e.g., white, grammatical)
Relation (e.g., double, half, greater)
Place (e.g., in the Lyceum, in the market-place)
Time (e.g., yesterday, last year)
Position (e.g., is lying, is sitting)
State (e.g., shod, armed)
Action (e.g., to lance, to cauterize)
Affection (e.g., to be lanced, to be cauterized)

The important property of the categories is that they are not composite. That is, they need to be composed in order to make statements about the nature that can yield affirmation. One should not focus too much on the used language but rather on the things that the categories try to define. The exhaustiveness of the list is also debatable but as a first guide for structuring an ontology, its precision is striking. As we will show later, these categories have a strong relation with computer science (see section 2.2.2) even if they come from a remote past.

The term *ontologia* instead, was created in the circles of German protestants around 1600. The first appearance of the Latin word *ontologia* can be found in two books published in 1613 by Rudolf Göckelin (Professor of Logic in the University of Marburg) (Goc80) and Jacob Lorhard (Professor at the University of St. Gallen - Switzerland) (Lor). However, as Dennis Bielfeldt (Bie03) pointed out, "ontology is as old as philosophy itself". Sometimes "ontology" is used in a broader sense to refer to the study of what might exist, while "metaphysics" is used for the study of which of the possible kind of ontologies is in fact true. The first occurrence in English appeared instead, in Bailey's 1721 dictionary which defines ontology as "an Account of being in the Abstract" (Smi03). Philosophical ontology is called descriptive or realist ontology. It does not seek to explain but rather to describe reality in terms of a classification of entities. This description can be considered exhaustive in the sense that it can serve as an answer to such questions as:

What classes of entities are needed for a complete description

and explanation of all the goings-on in the universe?
Or, what classes of entities are needed to give an account of what makes true all truths?
Or, what classes of entities are needed to facilitate the making of predictions about the future?

Sometimes a division is made between formal and material (or regional) ontology.
Formal ontology is domain-neutral and it deals with those aspects of reality (for example the identity) which are shared by all material regions. Material ontology instead deals with those features (for example mind or causality) which are specific to given domains. The history of philosophical ontology is indeed marked by a certain tradeoff between generativity on the one hand and descriptiveness on the other. "Generativity" means the power of an ontology to yield new categories and thus to exhaust the domain that is to be covered by ontological investigation in some recursive fashion. Thus, generativity gives ontology its power while descriptiveness ties an ontology to the world beyond.

It is interesting to point out how ontology and science are related to each other. Philosophical ontology is a descriptive enterprise. It is distinguished from the specific sciences not only in its radical generality but also in its primary goal or focus. Ontology especially seeks taxonomy rather than predication or explanation. Therefore, we can assert that Ontology is (very largely) qualitative while Science is (very largely) quantitative. Science starts, very roughly, with measurement and prediction. Even if there exists an ontology of measure (BP90), ontologists do not measure reality.
Philosophical ontology tells us what categories exist within a given domain of reality and thus what categories are available for the measurement process. Science tells us for example how the measurable behaviour of entities of a certain class is correlated with the behaviour of entities of a second class. The sciences, by definition, can deal only with the objects that fall within their respective domains. Ontology deals with transcategorial relations including the relations that hold between entities belong-

ing to distinct domains of science, and also between those entities and the entities recognized by common sense.

Indeed, in the course of the Twentieth Century, a range of formal tools became available to ontologists for the development and testing of their theories. Ontologists nowadays have a choice of formal frameworks (deriving from formal logic, as well as from algebra, category theory, mereology, set theory, topology) in terms of which their theories can be formulated. These new formal tools allow philosophical ontologists to express intuitive principles and definitions in a clear and rigorous fashion, and they can also allow for the testing of theories for consistency and completeness through the application of the methods of formal semantics (Smi; Smi03).

2.2.2 ...to the Information Science

As stated by B. Smith, the first use of the term "ontology" in the computer and information science literature occurred in 1967, in a work on the foundations of data modelling by S. H. Mealy (Mea67). From the scientific point of view, the ontology is, in its first approximation, a table of categories in which every type of entity is captured by some node within a hierarchical tree. This view comes directly from Aristotle's thinking of categories, and it has been adopted also by contemporary ontologists. With a reference to Aristotle's categories (AriCE), for example, things like substance, quantity, quality, location and time are all vital components of a Service Level Agreement[3] (SLA). A typical SLA would have, for example, statements concerning maximum (quantity) bandwidth (substance) in a subnet (place) during peak hours (time) with a certain reliability (quality).

In a related development, the term "ontology" has become popular in the field of computer and information science and especially in domains such as knowledge engineering, natural language processing, cooperative information systems, intelligent information integration, and knowl-

[3]Service Level Agreement describes the obligations and guarantees of service providers and consumers.

edge management. The philosopher-ontologist, in principle at least, has only the goal of establishing the truth about the domain in question. In contrast, in the world of information systems, an ontology is a software or formal language designed with a specific set of uses and computational environments, and often ordered by a specific client, customer or application program in a specific context. In this field, ontologies are rapidly developing thanks to their focus on classification and on constraints on allowable taxonomies and definitions, a focus not foreseen by its initial progenitors. One of the first important work is (GW00), in which the authors propose a general methodology for ontology-driven conceptual analysis (ODCA), which combines the established tradition of formal ontology and philosophy with the needs of information systems design. In particular, they outline a common problem of ontologies, i.e. that their taxonomic structure is often poor and confusing. Their methodology, based on four fundamental ontological notions (identity, unity, rigidity, and dependence), permits them to represent the behaviour of a property with respect to these notions by means of a set of meta-properties. Their goal is to show how these meta-properties impose some constraints on the way subsumption is used to model a domain.

So to the question *"What are the advantages of developing an ontology?"*, we can answer with more and interesting reasons:

- To share common understanding of the structure of information among people or software agents.
- To enable reuse of domain knowledge.
- To make domain assumptions explicit.
- To separate domain knowledge from operational knowledge.
- To analyze domain knowledge.

Since the beginning of 1990, one of the most common goals in developing ontologies has been for technical people and software agents to share a common understanding of the structure of information. For example, different Websites share and publish the same ontology of the terms they all use, allowing computer agents to extract and aggregate information from them. The agents can use this aggregated information to answer

user queries or as input data to other applications.
Reuse is one of the principles of programming methods, and it is also valid for knowledge (ontology). Therefore, if one group of researchers develops an ontology in detail (for example for describing "Time", "Banking system", ...), others can simply reuse it for their domains an purposes. It is common practice to start from general upper ontologies, such as *SUMO*[4] or *PROTON*[5] ontologies, and extend them to describe their own domain of interest. Making explicit domain assumptions permits one to make changes easily when the knowledge about the domain changes. In addition, explicit specifications of domain knowledge are useful for new users who must learn the used terms.

Separating the domain knowledge from the operational knowledge is another common use of ontologies, similar to the method adopted in object-oriented programming. For example, we can describe a task of configuring a product from its components according to a required specification, and implement a program that does this configuration independent of the products and components themselves.
Analysing domain knowledge is possible once a declarative specification of the terms is available. Formal analysis of terms is extremely valuable when attempting both to reuse existing ontologies and extending them.

2.2.3 Logics and Ontology: Languages and Reasoning

Ontologies are made up of formal theories about a specific domain, and thus have a formal logical language associated with them.
Logic has been proposed since the early days of the Artificial Intelligence (AI) as a framework for KR and reasoning (Nil02; MW85). In this context it provides the three main components necessary for structuring and handling the objects of the actual world, that is:

[Syntax] - An alphabet;
[Semantic] - An interpretation of the alphabet symbols;

[4]SUMO: Suggested Upper Merged Ontology - `http://www.ontologyportal.org/`
[5]PROTON: PROTo ONtology - `http://proton.semanticweb.org/`

[Proof System] - A method for making proofs.

Propositions are the basic items that logic deals with. The *Propositional logic* or *0-order logic* is used essentially for expressing beliefs, and in computer science, for hardware specifications. Its syntax is based on *Atoms* (P, Q, ...) which represent propositions, and *Logic Connectives* such as conjunction, disjunction, negation, implication ($\wedge$, $\vee$, $\neg$ and $\supset$). Its semantics is given by an *Interpretation Function* that assigns a truth-value to each set of propositional symbols (e.g., ν: $P \longrightarrow Bool$).

One of the most straightforward way to determine whether a sentence is valid is by a complete case analysis of the possible truth values assigned to its propositional symbols. Such a process is facilitated by means of *Truth Tables* or *Semantic-based trees*. The main limitation in the expressivity of the propositional language is its lack of generality. For example, in order to express a general fact such as, "every block is on the table", one has to state this property for each single block in the world.

Predicate Logics or First Order Logic (FOL) overcomes the limitations of propositional languages by allowing a better granularity in the construction of atomic statements. The language is extended to include *terms* (a term denotes an object in the world), and *relational symbols* of ariety n ($n \geq 0$), representing a relation between n objects. Existential and Universal ($\exists$ and $\forall$) quantification over a variable are introduced. An interpretation function I gives semantics to predicates. In general, an interpretation for an expression ϵ may assign a value to some symbols that are not free in ϵ (constant, variable, function or predicate) or variables that occur only bound in ϵ. Many proof systems have been developed from which Tableau, Axiomatic approaches, ...
While FOL is used for software/hardware specification and for building models, higher order, modal, temporal, probabilistic and fuzzy logics have been introduced for formalizing actions and their effects and for modelling more complex concepts (MW85; NS93; IA06).

Researches first adopted the existing logics (developed by mathematicians, logicians and philosophers), such as propositional, FOL, and modal logic to represent and reason about knowledge. However, the need to represent and efficiently reason about the many aspects of the world have lead to the design of "new logics". The main examples are description logics and Horn clause logics which gave rise to Prolog and Logic Programming[6]. They are at the basis of current systems for representing ontologies and taxonomic knowledge.
For the logic-based KR, it is necessary to pay attention to the fundamental trade off between expressive power, influenced by ontological requirements, and computational complexity, determined both by the intrinsic properties of the modelling constructs and by the computational properties of the reasoning algorithms.

The main problem in automated reasoning is to determine whether a "conjecture" φ, that represents a property to be verified, is a logical consequence of a set S of assumptions which express properties of the reference object (e.g., a system, a circuit, a program, a data type, a communication protocol, a mathematical structure). Theorem proving tries to solve the problem of finding a proof of φ from S. It comprises both deductive theorem proving, which concerns the problem stated above, (in symbols: $S \models \varphi$), and inductive theorem proving where the problem is to determine whether S entails all ground instances of φ; in symbols: $S \models \varphi\sigma$, for all ground substitutions σ).

In general the more expressive the logic, the more difficult it is to reason on it. For instance, checking if $S \models \varphi$ in propositional logic is decidable, while in FOL it is not. For subsets of FOL it is decidable, but with varying complexity depending on the set of operators. Decidability may stem from imposing restrictions on the logic, the form of admissible formulas for S and φ, or the theory presented by the assumption in S.
For this proposal, we will focus on DL approaches.

[6]Logic programming was proposed with the goal of combining the use of logic as a representation language with efficient deduction techniques, based on a backward inference process (goal-directed) which allows one to consider a set of formulas as a program. Prolog is the most widely used logic programming language.

2.2.4 Description Logic Based Approaches

Description Logics (DLs) are a family of knowledge representation languages that can be used to represent the knowledge of an application domain in a structured and formal way. The name "description logics" is motivated by the fact that, on the one hand, the important notions of the domain are described by concept descriptions, i.e., expressions that are built from atomic concepts (unary predicates) and atomic roles (binary predicates) using the concept and role constructors provided by the particular DL. On the other hand, DLs differs from their predecessors, such as semantic networks and frames, in that they are equipped with a formal, logic-based semantics.
As we can see in the following of this paragraph, DL is the theoretical basis of the recommended standard language for ontology representation (Dlg; Wik).

The evolution (and especially the hierarchy) of the various XML-based languages is shown in figure 3 which explains the semantic web architecture in agreement with the Tim Barners-Lee[7] point of view (BLF97).

The XML layer is used as a syntax layer while the RDF layer represents the data layer and permits to assign types to resources and links. The ontology layer specifies meaning and structure of the data: it is the container that defines in a formal way the relations between terms. The logic layer provides rules that enable further intelligent reasoning while, the proof layer supports the exchange of proofs in inter-agent communication.

XML documents are supported by XML Schemas (generally referred as XSD). An XML Schema provides a means for defining the structure and the content of an XML document and the corresponding constraints. These constraints are generally expressed by using combinations of grammatical rules, Boolean predicates on the content, data types that regulate the content of elements and attributes, and more specialized rules such as uniqueness and referential integrity constraints.

[7]Tim Barners-Lee is currently the Director of the World Wide Web Consortium and considered the founder of the web.

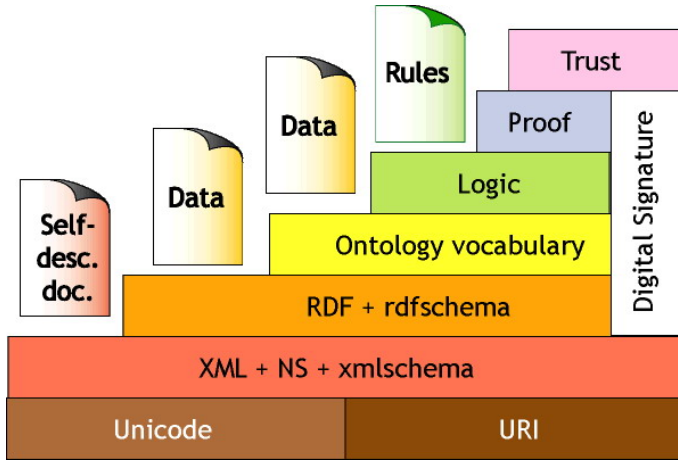


Figure 3: Semantic Web layers.

RDF - Resource Description Framework (LS99) is most commonly mentioned as a language, but it is rather a data model, independent of any domain or implementation. It has been developed to provide meaning to the Web documents, adding metadata in order to achieve terminological consensus on the Web. It provides interoperability between the applications that exchange machine-understandable information on the Web, and it emphasizes facilities to enable automated processing of Web resources. RDF can be used in various application areas such as in *resource discovery* to provide better search engine capabilities and in *cataloguing* for describing the content and content relationships available at a particular Web site, page, or digital library. It can also be used by *intelligent software agents* to facilitate knowledge-sharing and exchange, in *content rating*, in *describing collections of pages* that represent a single logical "document", for *describing intellectual property rights* of Web pages, and for *expressing the privacy preferences* of a user as well as the privacy policies of a Web site. RDF with digital signatures will be the key for building the "Web of Trust" for electronic commerce, collaboration, and other applications.

The RDF model is based on three types of objects: Resources, Proper-

Subjects (Resources)	http://www.MarioRossi.it/doc1.html www.comuni.it/servizi/codfisc/RSSMRA70A01E715H
Predicates (Properties)	Author, Name, Affiliation
Objects (Values)	Mario Rossi mario.rossi@imtlucca.it IMT

Table 1: Identification of the sentence elements.

ties and Statements.

[Class] *Resources* are all things being described by RDF expressions. Thus a resource may be an HTML document, part of a Web page (e.g. a specific HTML or XML element within the document source), or it can be a collection of pages (e.g. an entire Web site).

[Property] A *Property* is a specific feature, attribute or a relation that describes resources and that has a defined meaning. A property together with its value for a specific resource makes a statement about that resource.

[Statement] An RDF statement is a resource with its own property so a *Statement* is a tuple composed of subject (resource), predicate (property) and object (value).

RDF is a graph based data model, and it consists of nodes and edges. Nodes correspond to objects or resources and the edges correspond to properties. The labels on the nodes and on the edges are Uniform Resource Identifiers (URIs). As an example, consider the following sentence:

> *The person identified by the fiscal code RSSMRA70A01E715H with name Mario Rossi, Email address mario.rossi@imtlucca.it, affiliation IMT, is author of the resource http://www.MarioRossi.it/doc1.html.*

The sentence has the parts shown in table 1 and graphically corresponds to the diagram in figure 4.

Graphically, relations between resources, properties and values are depicted by using labelled and oriented graphs; resources are nodes (el-

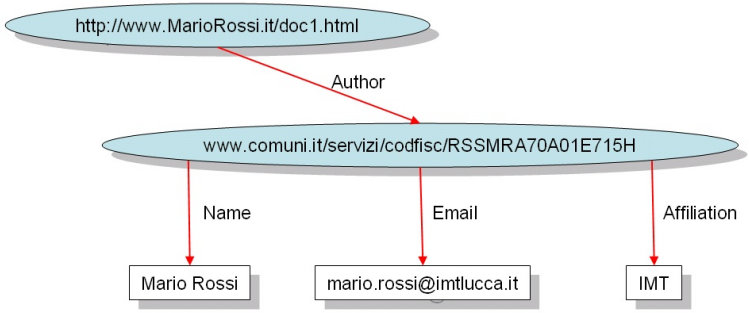


Figure 4: Graphic representation of two linked RDF statements.

lipses), properties are oriented edges while values are rectangles.

RDF itself does not define any primitives for creating ontologies, but it provides the basis for several other ontology definition languages such as RDFS. The RDF Schema or RDFS (Com04b) has been developed, such as the XML Schema for an XML document, in order to define the vocabulary used in RDF data models by specifying which kinds of properties apply to which kinds of objects, what values the objects can take and what kinds of relationships between those objects exist. Therefore, RDFS is considered as a first move toward an ontology language for the Web. It offers a fixed set of modelling primitives such as *rdfs:Class*, *rdf:Property* or the *rdfs:subClassOf* relationship to define RDF vocabularies for some specific application. In RDFS it is possible to define classes of classes, classes of properties, classes of literals that are strings, integers, Booleans and so forth, and classes of statements. Using RDFS properties, which are *rdf:type*, *rdfs:subClassOf* and *rdfs:subPropertyOf*, it is possible to define instanceOf relationship between resources and classes, subsumption relationship between classes and subsumption relationship between properties, respectively. Using *rdfs:domain* and *rdfs:range* properties it is possible to restrict the resources that can be subjects or objects of the property. Nevertheless, RDFS is not expressive enough for defining useful ontologies. For example, disjoint, union, intersection and complement classes

cannot be defined, cardinality restrictions are not present, and properties cannot be declared as transitive, symmetric or inverse of each other.

Recently, a work-group at W3C has continued the work for producing a recommendation for an ontology language. The Web Ontology Language (OWL) (Com04a) is designed for use by applications that need to process the content of information instead of just presenting information to humans. It facilitates the machine interpretability of Web contents better than that supported by XML, RDF, and RDFS. OWL is syntactically layered on RDF, but it adds more vocabulary for describing properties and classes: among others, relations between classes (e.g. disjointness), cardinality (e.g. "exactly one"), equality, richer typing of properties, characteristics of properties (e.g. symmetry), and enumerated classes. The basic elements of the OWL ontology are classes, properties, instances of classes, and relationships between these instances.

[Class] A class is a collection of individuals (object, things, ...) and it is the most basic concept for describing part of the world. Every individual in the OWL world is a member of the class `owl:Thing`. Domain specific root classes are defined by simply declaring a named class.

[Individual] An individual is an object of the world, and in particular a member of a class. Individuals are related to other objects and to data values via properties.

[Property] A property is a binary relation that lets us assert general facts about the members of classes and specific facts about individuals (e.g. `hasFather`, `hasPet`, `serviceNumber`). There are two types of properties: *datatype property* and *object property*. While the former expresses relations between instances of classes and RDF literals and XML Schema datatypes, the latter expresses relations between instances of two classes.

The following example 2.1, taken from (Cap05), can exemplify the concepts just described.

Example 2.1 *A simple ontology.*
Suppose we have seven individuals (`Andrew`, `Milan`, `Naples`, `Nic`, `Rome`, `Pisa` and `Mary`) grouped in two classes (`Towns` and `Persons`) and related by means of three types of properties (`hasBrother`, `hasWife`, `liveInTown`). A simple ontology, built by using those elements, is graphically depicted in figure 5.
`Nic`, `Mary` and `Andrew` are `Persons` while `Naples`, `Milan`, `Rome` and `Pisa` are `Towns`. `Mary`, who lives in `Pisa`, is the wife of `Nic`. `Mary` also has a brother named `Andrew` who lives in `Naples`. The OWL formal-

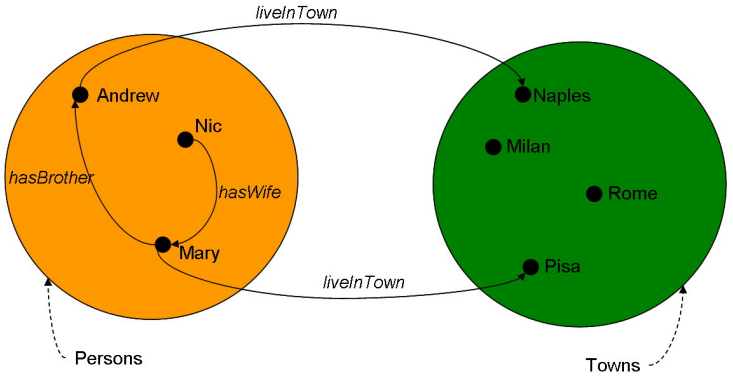


Figure 5: A simple ontology.

ization of the classes is shown in table 2. The meaning is that `Towns` and `Persons` are resources of RDF[8] and classes in OWL. The individuals

<owl:Class rdf:ID="Towns"/>
<owl:Class rdf:ID="Persons"/>

Table 2: OWL Class definition.

are introduced by declaring them as being members of a class (table 3). Each object property is defined and linked to the classes by expressing the domain (`rdfs:domain`) and the co-domain (`rdfs:range`) as shown in table 4.

Since `hasWife` and `hasBrother` properties tie individuals of the same class, it is possible to express the individuals involved in the rela-

[8]The `rdf:ID` attribute on a node element gives a relative RDF URI reference.

<Towns rdf:ID=”Milan” /> <Towns rdf:ID=”Naples” /> <Towns rdf:ID=”Pisa” /> <Towns rdf:ID=”Rome” />
<Persons rdf:ID=”Nic” /> <Persons rdf:ID=”Mary” /> <Persons rdf:ID=”Andrew” />

Table 3: OWL Individuals definition.

<owl:ObjectProperty rdf:ID=”hasBrother”> <rdfs:domain rdf:resource=”&owl;Thing” /> <rdfs:range rdf:resource=”#Persons” /> </owl:ObjectProperty>
<owl:ObjectProperty rdf:ID=”hasWife”> <rdfs:domain rdf:resource=”&owl;Thing” /> <rdfs:range rdf:resource=”#Persons” /> </owl:ObjectProperty>
<owl:ObjectProperty rdf:ID=”liveInTown”> <rdfs:domain rdf:resource=”Persons” /> <rdfs:range rdf:resource=”#Town” /> </owl:ObjectProperty>

Table 4: OWL Properties definition.

tions. Therefore, we can extend the definition of table 3 with that one of table 5.

```
<Persons rdf:ID="Nic" />
    <hasWife rdf:resource="#Mary" />
</Persons >
<Persons rdf:ID="Mary" />
    <hasBrother rdf:resource="#Andrew" />
</Persons >
```

Table 5: OWL definition of Properties on Individuals.

◁

The standard (Com04a) foresees three increasingly-expressive sublanguages:

[OWL Lite] It supports the users' primarily needs with a classification hierarchy and simple constraints. For example, while it supports cardinality constraints, it only permits cardinality values of 0 or 1. It has a lower formal complexity than the other versions, so it should be simpler to provide tool support for OWL Lite. Furthermore, it provides a quick migration path for thesauri and other taxonomies.

[OWL DL] It owes its name to the correspondence with description logics, a field of research that has studied the logics that form the formal foundation of OWL. OWL DL supports those users who want the maximum expressiveness while maintaining computational completeness and decidability. OWL DL includes all OWL language constructs, but they can be used only under certain restrictions. For example, while a class may be a subclass of many classes, a class cannot be an instance of another class.

[OWL Full] It is for users who want maximum expressiveness and the syntactic freedom of RDF with no computational guarantees. For example, in OWL Full a class can be treated simultaneously as a collection of individuals and as an individual in its own right. OWL

Full allows an ontology to augment the meaning of the predefined (RDF or OWL) vocabulary.

It is important to point out that OWL Full can be considered as an extension of RDF, but OWL Lite and OWL DL can be considered only as extensions of a limited version of RDF. Thus, each OWL (Lite, DL, Full) document is an RDF document, and each RDF document is an OWL Full document, but only some RDF documents are an OWL Lite or OWL DL documents.

2.2.5 Ontology Based Reasoning: an Example

As we have seen in the previous section, languages like OWL specify a vocabulary and constrain the use of that vocabulary by restrictions. However, they also provide axioms which allow one to deduce new information from explicit information. Ontologies are set to play a key role in the "Semantic Web" by providing a source of shared and precisely defined terms that can be used for describing the resources. Reasoning over such descriptions is essential for accessibility purposes, automating processes and discover new knowledge. From a formal point of view, OWL can be seen as to be equivalent to a specific member in the DLs family, which allows OWL to exploit the considerable existing body of DL reasoning including class consistency and subsumption, and other ontological reasoning.

A very useful example of reasoning by using OWL is taken from (WGZP04). In this paper, the authors propose an OWL encoded context ontology (named CONON) for modelling context in pervasive computing environments, and for supporting logic based context reasoning. CONON provides an upper context ontology that captures general concepts about basic context, and provides extensibility for adding domain-specific ontology in a hierarchical manner. Based on this context ontology, they studied the use of logic reasoning to check the consistency of context information, and to reason over low-level, explicit context to derive high-level, implicit context. In the specific, the example shows how

Transitive Property	(?P rdf:type owl:TransitiveProperty) ∧ (?A ?P ?B) ∧ (?B ?P ?C) ⇒ (?A ?P ?C)
subClassOf	(?a rdfs:subClassOf ?b) ∧ (?b rdfs:subClassOf ?c) ⇒ (?a rdfs:subClassOf ?c)
subPropertyOf	(?a rdfs:subPropertyOf ?b) ∧ (?b rdfs:subPropertyOf ?c) ⇒ (?a rdfs:subPropertyOf ?c)
disjointWith	(?C owl:disjointWith ?D) ∧ (?X rdf:type ?C) ∧ (?Y rdf:type ?D) ⇒ (?X owl:differentFrom ?Y)
inverseOf	(?P owl:inverseOf ?Q) ∧ (?X ?P ?Y) ⇒ (?Y ?Q ?X)

Table 6: Parts of OWL ontology reasoning rules.

to deduce information starting from a specific context, and the formalization of rules. In particular it explains how to extract implicit knowledge from an explicit one. The OWL rules, that involve OWL properties, are shown in table 6. The application context can be informally described as follows:

> *User Wang is currently located in the bedroom, which is in turn, a part of the home building.*

By means of the rules that involve the `owl:TransitiveProperty` and `owl:inverseOf`, we can conclude that:

> *Wang is located in the home building.*

The formalization of the example is shown in table 7.

While the ontology layer already provides means for deducing new knowledge (information) and provides restricted reasoning support, many applications require further means to combine and deduce information. OWL adds considerable expressive power to the Semantic Web, however, for a variety of reasons it has expressive limitations. Many of the limitations of OWL stem from the fact that, while the language includes a relatively rich set of class constructors, the language provided for talking about properties is much weaker. In particular, there is no composition constructor, so it is impossible to capture relationships between a composite property and another possibly composite property. To address this

INPUT	DL Reasoning Rules	(?P rdf:type owl:TransitiveProperty) ∧ (?A ?P ?B) ∧ (?B ?P ?C) ⇒ (?A ?P ?C)
		(?P owl:inverseOf ?Q) ∧ (?X ?P ?Y) ⇒ (?Y ?Q ?X)
	Explicit Context	<owl:ObjectProperty rdf:ID="locatedIn"> <rdf:type="owl:TransitiveProperty" /> <rdfinverseOf rdf:resource="#contains" /> </owl:ObjectProperty> <Person rdf:ID="Wang"> <locatedIn rdf:resource="#Bedroom" /> </Person> <Room rdf:ID="Bedroom"> <locatedIn rdf:resource="#Home" /> </Room>
OUTPUT	Implicit Context	<Person rdf:ID="Wang"> <locatedIn rdf:resource="#Home" /> </Person> <Building rdf:ID="Home"> <contains rdf:resource="#Bedroom" /> <contains rdf:resource="#Wang" /> </Building> <Room rdf:ID="Bedroom"> <contains rdf:resource="#Wang" /> </Room>

Table 7: Reasoning about location by using OWL ontology.

problem, a possible solution is to extend OWL with a more powerful language for describing properties. For example, a decidable extension of the description logics underlying OWL DL to include the use of composition in subproperty axioms has already been investigated. However, in order to maintain decidability, the usage of the constructor is limited (Hor05).

Logic Programming systems (LP), such as Prolog, offer efficient environments to do so. Large communities from the LP environment are working on new solutions for improving the exchange of rules and the reasoning ability. Some of them are working on a standard for exchanging rules in the semantic web called RuleML[9]. Although it is a valid candidate, RuleML is not layered on top on ontology but operates on the data layer only, so the two "environments" are split. Some other researchers are studying instead on systems for integrating ontologies and rules.
SWRL[10] (Semantic Web Rule Language) is a proposal for a Semantic Web rules-language, combining sublanguages of the OWL with those of the Rule Markup Language (Unary/Binary Datalog).

It is important to mention that there are important connections and implications between the knowledge modelling components (concepts, roles, etc.) used to build an ontology, the knowledge representation paradigms (frames, description logics, logic) used to represent formally such components, and the languages used to implement the ontologies under a given knowledge representation paradigm.
Most languages have been developed following two approaches: First-Order predicate Logic (FOL) and the XML-RDF (Description Logic based). Whereas the languages of the first type are more generic, the XML-RDF based languages are specific for the development of Web ontologies. The *Semantic Web* in fact, is built around a semi-structured data model (RDF) and an explicit conceptualization for such data (the ontologies).

[9]RuleML: `http://www.ruleml.org/`
[10]SWRL: `http://www.w3.org/Submission/SWRL/`

Chapter 3

Answering the Open Questions

In this chapter we look at the main issues related to the proposal, and the solutions we judge suitable for approaching the problems. Therefore, each paragraph is titled with a question that refers to a particular step of analysis for the system we propose. In each subsection, we inquire several methodologies and theories with the attempt to answer the corresponding initial question. Furthermore, section 3.3 gives a short overview of Social Networks and Social Network Analysis, and discusses their relationships with ontologies.

3.1 How to Mine the Ontology Schema?

The main issue deals with the analysis of the ontology schema and the extraction of the interesting information based on the ontology structure (concepts, object and datatype properties). The ontology schema essentially refers to the structural aspect of the ontology, and the attempt to infer "semantics" starting from this schema can seem too ambitious. Nevertheless, if one considers the objective of the work, the idea makes sense. The schema describes objects and relations, and the analysis we want to perform concerns the discovery of the most important relationships

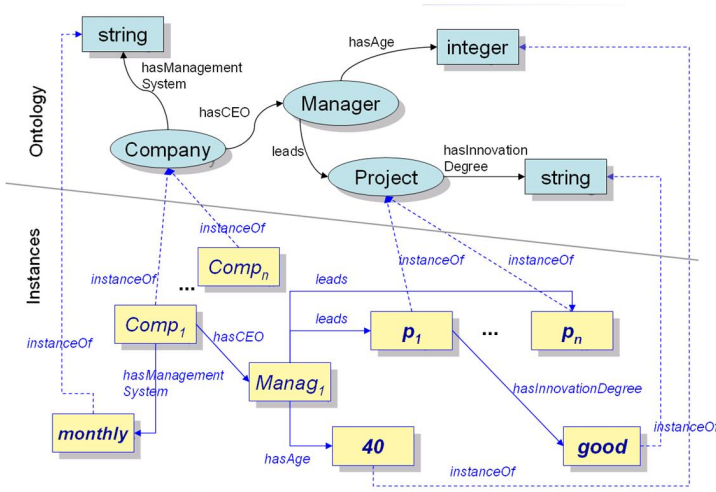


Figure 6: Example of rules extraction from ontology schema and instances.

among concepts (for the first phase). In figure 1, this phase is represented by a deductive step. By navigating the ontology throughout its connections, we can identify the more relevant concepts and the corresponding relationships. This analysis leads to the definition of the $Implications$ as shown in figure 1.

The example that has driven us during this study is reported in figure 6. Let us suppose we have a fragment of ontology that describes companies and the business environment where *Company*, *Manager* and *Project* are concepts. Furthermore, continuous arrows represent properties of the ontologies while the dotted ones are used for connecting instances to the classes they belong to. From the ontology schema, we would like to extract the causal relation that links up the four concepts and that express a relationship like:

> *"If a manager has a certain age, and he leads a certain project, then the project he manages has a certain degree of innovation."*.

More formally:

Manager(X) $\wedge$ *has_age(X, N)* $\wedge$ *leads(X, P)* $\longrightarrow$ *has_innovation_degree(P, V)*.

At this stage, this relation is not properly a (influence) rule, but rather a schema of the possible rules because it lacks both a weight and a quantification for the variables N and V. To solve this problem, many technologies have been investigated, taking inspiration from the graph theory and the Semantic Web environment.
What follows is a short survey.

3.1.1 Graph Theory: a Structural Analysis

The simplest way of analysing the ontology structure is to consider it as a Direct Graph. If we want to "capture" the structural information, the correspondence between ontology and graph has to be formalized and implemented. For our purpose, it is sufficient to put in place the following simple actions:

A1: Identify each ontology concept as a labelled node in the graph.
This correspondence is quite obvious and it is essential to keep track of the concept name.

A2: Codify each object property between concepts as a direct edge between the corresponding nodes.
Again, matching object properties with edges is quite natural, but in this case for the kind of analysis we want to perform, we can leave out the semantics of the properties; we are not interested in the information a property carries, but rather its existence.

A3: Keep track of the *is-a* relationships by using "special" arrows.
As we will explain in section 4.2.1, these kinds of relationships are important even if they are not considered to be the object properties, and for this reason they are coded as "special arrows".

In addition to this formalization, it has been proven that the representation of the ontology graph by means of its Adjacency Matrix is practically used. This is, in fact, a means of representing which vertices of a

graph are adjacent to which other vertices. Given a graph G with n vertices, it is an $n \times n$ matrix where the nondiagonal entry a_{ij} is the number of edges from vertex i to vertex j, and the diagonal entry a_{ii}, depending on a convention, is either once or twice the number of edges (loops) from vertex i to itself. Undirected graphs often use the former convention of counting loops twice, whereas directed graphs typically use the latter one. In the special case of a finite simple graph, it is a $(0, 1)$-matrix with zeros on its diagonal. The useful feature is that the adjacency matrix is unique for each graph up to permuting rows and columns.

Section 4.2.1 supplies formalizations and examples of the expounded concept as well as all the technical details.
Nevertheless, we think it is important to mention that the graph theory supplies a wide range of tools for the structural analysis of the graphs. Of particular interest is the research line that merges the graph analysis to the DM or, rather, that tends to apply DM techniques to graphs. As we will see in section 3.3, this kind of research has become attractive thanks to the increasing interest on Social Networks and Social Networks Analysis. For the sake of completeness, we report a brief review of graph theory related to the DM (getting hints from (WM03)), and we point to section 3.3 for an overview of the Social Networks.

For our purposes, it is interesting and useful to explicitly cite the five theoretical bases of graph-based data mining: subgraph categories, subgraph isomorphism, graph invariants, mining measures and solution methods.

Subgraph categories. Let us consider a graph G represented as $G(V, E, f)$ where: V is a set of vertices, E a set of edges connecting some vertex pairs in V and f a mapping $f : E \to V \times V$ (figure 7(a)).
The most generic class of the substructure of G is a *general subgraph* where $V_s \subset V$, $E_s \subset E$ and $v_i, v_j \in V_s$ for all edges $f(e_h) = (v_i, v_j) \in E_s$. Figure 7(b) is an example in which a vertex v_5 and edges e_4, e_6, e_7, e_8, e_9 are missing.

Another important and generic class of the sub-structure is an *in-*

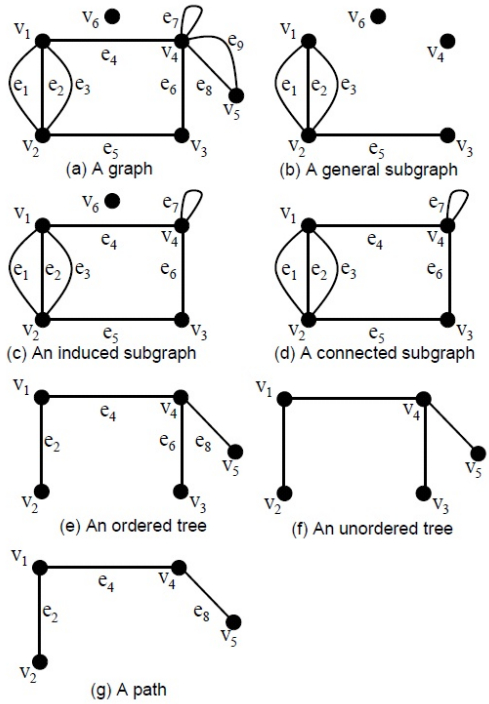


Figure 7: Representative subgraphs.

duced subgraph where: $V_s \subset V$, $E_s \subset E$ and $\forall v_i, v_j \in V_s$, $e_h = (v_i, v_j) \in E_s \Leftrightarrow f(e_h) = (v_i, v_j) \in E$. An induced subgraph G_i^s of a graph G has a subset of the vertices of G and the same edges between pairs of vertices as in G. In the case of figure 7(c) the vertex v_5 is missing and only the edges e_8 and e_9 are also missing. e_4, e_6, e_7 are retained since they exist among v_1, v_3 and v_4 in the original G.

The third class of the substructure is a *connected subgraph* where $V_s \subset V$, $E_s \subset E$ and all vertices in V_s are mutually reachable through some edges in E_s. Figure 7(d) shows the case where v_6 is further missing from the graph of figure 7(c).

Considering the labels of edges in the tree, and whether they are ordered in a way that the label of an edge is always younger than the labels of its lower (upper) and right (left) edges, the tree is defined as an *ordered tree* (figure 7(e)). If the edge is not ordered or does not have labels, the tree is called an *unordered tree* (figure 7(f)).

Finally, if the substructure does not include any branches, it is called a *path* of the original graph G (figure 7(g)).

Subgraph isomorphism. Given two graphs $G_x(V_x, E_x, f_x)$ and $G_y(V_y, E_y, f_y)$, the *subgraph isomorphism* problem consists of finding the subgraphs $G_{sx}(V_{sx}, E_{sx}, f_x)$, $G_{sy}(V_{sy}, E_{sy}, f_y)$ and a bijection mapping g_{xy} between the vertices in V_{sx} and the vertices in V_{sy} such that G_{sx} and G_{sx} are identical, i.e., $f_x(e_{xh}) = (v_{xi}, v_{xj}) \in E_{sx}$ iff $f_y(e_{yh}) = (v_{yi}, v_{yj}) \in E_{sy}$, where $v_{yi} = g_{xy}(v_{xi})$ and $v_{yj} = g_{xy}(v_{xj})$.

For example, the graphs (b) and (d) in figure 7 share the subgraph composed of the vertices $\{v_1, v_2, v_3\}$ and the edges $\{e_1, e_2, e_3, e_5\}$ under the bijection mapping of $v_i = g_{bd}(v_i)$, $i = 1, 2, 3$. This mapping is a subgraph isomorphism between the graphs (b) and (d) of figure 7.

Graph invariants. Graph invariants are the quantities that characterize the topological structure of a graph. If two graphs are topologically identical, i.e. isomorphic, they also have identical graph invariants, though the reverse property does not hold. Examples of graph invariants are the number of vertices, the degree of each vertex, i.e.,

the number of edges connected to the vertex, and the number of cyclic loops. This method can be used to reduce the search space to solve the subgraph isomorphism problem. If any of the graph invariants show different values between two subgraphs, the subgraphs are not isomorphic.

One of the most generic and important graph invariants is the *canonical label* and the *canonical form*. The former is defined as the lexicographic minimum (or maximum) code, and the canonical form of the adjacency matrix is the matrix corresponding to the canonical label. The use of both significantly reduces the graph representation ambiguity and the search space. Often, when we want to apply machine learning, DM or statistical approaches, a graph has to be transformed into a feature vector. A feature vector is an n-dimensional vector of numerical features that represent some objects.

Mining measures. The measures assigned with substructures of graphs are very similar to those applied to the DM analysis. The most popular is the *Support*. Given a graph data set D, the support of the subgraph G_s, is defined as:

$$sup(G_s) = \frac{\text{number of graphs including } G_s \text{ in } D}{\text{total number of graphs in } D}.$$

The anti-monotonicity of the support is insufficient for some mining objectives, for example, for finding subgraphs which appear more than a minimum support (minsup) but also less than a maximum support (maxsup).
Many other mining measures which are very commonly used in the machine learning field are also used in some graph-based data mining approaches, especially, information entropy, information gain, gini-index and minimum description length.

Solution methods. It concerns the research methods. The first type of search method is the conventional greedy search that belongs to heuristic search and direct matching (depth-first search and breadth-first search). The second type of search method applies the *Inductive*

Logic Programming (ILP). The induction is the combination of the abduction to select some hypotheses and the justification to seek the hypotheses to justify the observed facts.

The third type uses an *inductive database*. Given a data set, a mining approach such as inductive decision tree learning, basket analysis and ILP is applied to the data to pregenerate inductive rules, relations or patterns. The induced results are stored in a database. The database is queried by using a query language designed to express query conditions on the forms of the pregenerated results in the database. This framework is applicable to graph-based mining. Subgraphs and/or relations among subgraphs are pregenerated by using a graph-based mining approach, and stored in an inductive database. A query on the subgraphs and/or the relations is made by using a query language dedicated to the database.

The fourth type is to apply a *complete level-wise search* which is popularly used in the basket analysis. In case of graph-based data mining, the data are not the transactions, i.e., sets of items, but graphs, i.e., combinations of a vertex set $V(G)$ and an edge set $E(G)$ which include topological information. Accordingly, the above level-wise search is extended to handle the connections of vertices and edges. The search in a given graph data starts from the frequent graphs of size 1 where each one consists of only a single vertex; this is similar to the A-priori algorithm. Subsequently, the candidate frequent graphs of size 2 are enumerated by combining two frequent vertices. Then the support of each candidate is counted in the graph data and only the graphs having higher support than the minsup are retained. In this counting stage edge information is used. If the existence and the label of the edge between the two vertices do not match, the graph of size 2 is not counted as an identical graph. This process is further repeated to incrementally extend the size of the frequent graphs in a level-wise manner, and finishes when the frequent graphs are exhaustively searched.

The fifth type is *Support Vector Machine* (SVM). This is a heuristic

search and an indirect method in terms of the subgraph isomorphism problem and used in the graph classification problem. It is not dedicated to graph data but to feature vector data. Given feature and class vectors $(x_1, y_1), \ldots, (x_L, y_L)$ $x_i \in Z$, $y_i \in \{+1, -1\}$, where L is the total number of data, $i = 1, \ldots L$, Z a set of vectors and y_i a binary class labels, each sample feature vector x_1 in the data is classified by:

$$y = sgn(\sum_{j=1}^{n} j_i \alpha_j \phi(x_i) \bullet \phi(x) + b).$$

Here $\phi : Z \rightarrow H$ where H is the Hilbert space[1], $\alpha_i, b \in R$ and α_i positive finite. By extending the feature space to a far higher dimension space via ϕ, SVM can properly classify the samples by a linear hyper plane even under complex nonlinear distributions of the samples in terms of the class in Z. The product $\phi(x_i) \bullet \phi(x)$ can be represented by the kernel function $K(X_{G_x}, X_{G_y})$ for graphs where $X_{G_x} = x_i$ and $X_{G_y} = x$. This function K represents a similarity between two graphs G_x and G_y. Accordingly, SVM can provide an efficient classifier based on the set of graph invariants (WM03).

3.1.2 Ontology Evaluation Metrics: Pointing Out the Relevant Concepts

When we talk about evaluating ontologies, what we usually have in mind is a sort of "objective" evaluation of how "good" an ontology is. Methodologies such as OntoClean (GW02) help to validate taxonomic relationships with respect to general ontological notions such as essence, identity and union. Others suggest assessing ontology completeness, consistency and correctness in terms of consistency of inference, lack of redundancy, lack of errors, and so on. Furthermore, many have argued that the only true way to evaluate an ontology is to use it in applications and to assess

[1]The mathematical concept of a Hilbert space generalizes the notion of Euclidean space. It extends the methods of vector algebra from the two-dimensional plane and three-dimensional space to infinite-dimensional spaces. Explicitly, a Hilbert space is an inner product space, an abstract vector space in which distances and angles can be measured, which is "complete" [Wikipedia].

the applications performances. In other words, like every software product, ontologies need to be controlled before being deployed in practical applications. Because of their nature, the evaluation metrics and quality enforcement procedures developed for software engineering do not work. For example, we can not evaluate ontologies in terms of their correctness with respect to a given process specification, described, for instance, using an I/O function. For this reason, many *ad hoc* techniques have been developed (SGPD+04).

There exist two types of evaluation:

i) content evaluation, and

ii) ontology technology evaluation.

Evaluating contents is for preventing applications from using inconsistent, incorrect, or redundant ontologies. A well-evaluated ontology does not guarantee the absence of problems, but it will make its use safer. Similarly, evaluating ontology technology will ease its integration with other software environments, ensuring a correct technology transfer from the academic to the industrial world.

For our purposes, we found some ideas in (HD06), where the authors propose some evaluation techniques based only on the ontology structure. Their aim is to supply parameters and methods for increasing the quality of an ontology. The evaluation method includes six parts: *Concept Quantity*, *Property Expectation*, *Property Standard Deviation*, *Tree Balance*, *Concept Connectivity*, and *Key Concept Quantity*.
From this set, the more attractive and useful technique is the last one. The goal of Key Concept Quantity evaluation is to find out the key concepts in the ontology. This evaluation helps the ontology developers to focus their attention on the main concepts (the key ones), and for us it can represent a starting point for the analysis of a large ontology.
The authors' idea is to extract the most important concepts from the ontology and to start, from this set, the search of the more interesting relationships.

Since it is difficult to judge what the most important concepts are without considering and knowing their semantics, the authors propose to consider the following heuristic:

> *"if a concept is more important than the others, it has more contacts with other concepts than the other concepts".*

As described in 3.1.1, also in this case the suitable formalism for the ontology is a (direct) graph $G =< V, E >$, where each concept is a vertex V in the graph. The strategy for highlighting the key concepts is the following: if a concept has an object property, whose value is an instance of another concept, an edge E will be drawn between these two concepts. At first, each edge in graph G is given a weight with the default value of 1. Consequently, because the subclasses inherit the property of their parents, a contact between parents implies the relations of their subclasses. Therefore, the edge between two higher concepts also implies the edges between every two subclasses even if we do not draw them. Consequently, the weight between two higher concepts depends on the number of invisible edges[2], and we update the weight of an edge with the product obtained by multiplying the number of concepts and subclasses of each side.
Secondly, we calculate the weight of each concept by adding the weight of each edge that connects to the concept and sort the concepts with their weight.
Finally, we consider some of the concepts as key concepts by selecting those who obtain a score above the threshold. For different applications, developers can decide the proportion themselves.

This approach was certainly a good starting point for tackling the problem and for directing the research to more suitable techniques. Since the beginning, it has been clear that the approach gave simplistic solutions and presented some weaknesses. The similarity of an ontology with a graph is not completely exploited because the computation of the im-

[2]For invisible edges we mean the "hidden" connections that are induced by the taxonomic relations of an ontology.

portant concepts is based only on the in-links. As we will prove, it is possible, even necessary, to consider also the out-links. In this way we do not ignore the importance of the concepts that originate the relationships. The example below shows the application of the Key Concept Quality technique to the *resume* ontology (Yan) an ontology that models relevant information about a person's career. We will now point out the weakness just described in the example below.

Example 3.1 *Key Concepts of the resume ontology.*
This example aims at showing the application of the Key Concept evaluation to an actual ontology: the *resume* ontology (Yan). This ontology models essentially the concepts for describing the career of a person. For the case study we use the subset of the resume ontology shown in figure 8. As described earlier, we consider the hierarchy of concepts and their object properties, and we apply the sequence of steps for weighting properties and consequently for ranking the concepts. First, we point out the triples *Concept*$_i$ - *Relation*$_h$ - *Concept*$_j$. Then we weight all the relations keeping into consideration the fact that the edge between two higher concepts also implies the existence of the same edges between every two subclasses of them. In other words, if $Concept_i$ has 2 subclasses and $Concept_j$ has 3 subclasses, then $Relation_h$ has a weight of 12. 12 results from the multiplication of 3 ($Concept_i$ and its 2 subclasses) and 4 ($Concept_j$ and its 3 subclasses).

For evaluating the concepts, we calculate the weight of each concept by adding the weight of each edge that connects to the concept. Each step of analysis on the *resume* ontology is shown and summarized in the following pictures. Figure 9 shows the main classes related with an object property; for lack of space, for each parent class we omitted the list of all the subclasses reporting only their number. For example, the concept *Award* has 14 subclasses and it is related to *Organization* by the object property *awardedBy*. *Organization* has 8 subclasses. In agreement with the evaluation strategy, the weight 135 of the edge (relation) *awardedBy* (connecting *Award* and *Organization*) results from the product of 15 (*Award* and its 14 subclasses) and 9 (*Organization* and its 8 subclasses).

In the next step of evaluation we give a weight to the concepts summing up the weights of the incoming edges. Figure 10 shows the results in the last column named Concept Weight. For clarity, we order the re-

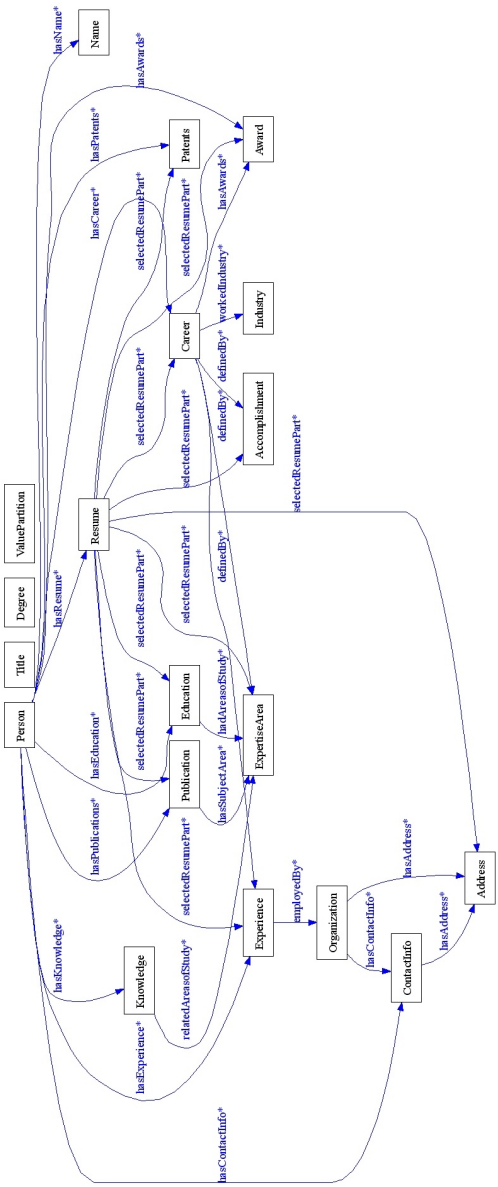


Figure 8: Graph of the (subset) *resume* ontology.

sulting table on the basis of range concepts so we can see immediately what are the incoming edges considered for each range class. Referring again to *Organization*, its final weight is 162 resulting from the sum of the weights of *awaredBy* (135 + 9 + 9) and *employedBy* (9).
At the end, the Key Concepts of resume ontology are: *Award* (180), *Organizaton* (162), *Activity* (20), *Career* (20), *CareerAccomplishment* (16), *ExpertiseArea* (14), and so on. Notice that, the hidden[3] subclasses have the same value of the corresponding parent classes, while those presented in the result table maintain the value collected.

An important observation is what we expected after taking a look at the ontology, and what we actually obtained. Since the ontology describes the careers of persons, we would expect, without knowing the computation strategy, that *Person* appeared as Key concept. Obviously it is not so, because *Person* has many outgoing links, and no incoming ones. From this point of view this methodology underestimates the actual "expressive power" of this ontology, in the sense that *Person* is, objectively, a significant concept but the method "badly" rates it.
We will see that the link analysis principles could go beyond this limitation.

◁

[3]The subclasses are not explicitly reported in the picture.

Domain Class	Nr. of SubCls of the domain class	Relations	Rel. Weight	Range Class	N. of SubCls of the range class
Experience	0	achieved	1	Accomplishment	0
Education	0	attendedInstitution	1	EducationalInstitution	0
Award	14	awardedBy	135	Organization	8
Degree	0	awardedBy	9	Organization	8
Patents	0	awardedBy	9	Organization	8
Career	9	definedBy	10	Accomplishment	0
Career	9	definedBy	10	Experience	0
Career	9	definedBy	10	ExpertiseArea	0
Experience	0	employedBy	9	Organization	8
Accomplishment	0	had_activity	20	Activity	19
Experience	0	had_role	1	Role	0
Education	0	hadAreasofStudy	1	ExpertiseArea	0
Accomplishment	0	has_CareerAccomplishments	16	CareerAccomplishments	15
ContactInfo	0	hasAddress	1	Address	0
Organization	8	hasAddress	9	Address	0
Career	9	hasAwards	150	Award	14
Person	0	hasAwards	15	Award	14
Person	0	hasCareer	10	Career	9
Organization	8	hasContactInfo	9	ContactInfo	0
Person	0	hasContactInfo	1	ContactInfo	0
Person	0	hasEducation	1	Education	0
Person	0	hasExperience	1	Experience	0
Person	0	hasKnowledge	1	Knowledge	0
Person	0	hasName	1	Name	0
Person	0	hasPatents	1	Patents	0
Person	0	hasPublications	1	Publication	0
Person	0	hasResume	1	Resume	0
Publication	0	hasSubjectArea	1	ExpertiseArea	0
Knowledge	0	relatedAreasofStudy	1	ExpertiseArea	0
Resume	0	selectedResumePart	1	Accomplishment	0
Resume	0	selectedResumePart	1	Address	0
Resume	0	selectedResumePart	15	Award	14
Resume	0	selectedResumePart	10	Career	9
Resume	0	selectedResumePart	1	Education	0
Resume	0	selectedResumePart	1	Experience	0
Resume	0	selectedResumePart	1	ExpertiseArea	0
Resume	0	selectedResumePart	1	Patents	0
Resume	0	selectedResumePart	1	Publication	0
Career	9	workedIndustry	10	Industry	0

Figure 9: Evaluation of Relations.

Domain Class	Nr. of SubCls of the domain class	Relations	Rel. Weight	Range Class	N. of SubCls of the range class	Concept Weight
Experience	0	achieved	1	Accomplishment	0	12
Career	9	definedBy	10	Accomplishment	0	
Resume	0	selectedResumePart	1	Accomplishment	0	
Accomplishment	0	had_activity	20	Activity	19	20
ContactInfo	0	hasAddress	1	Address	0	11
Organization	8	hasAddress	9	Address	0	
Resume	0	selectedResumePart	1	Address	0	
Career	9	hasAwards	150	Award	14	180
Person	0	hasAwards	15	Award	14	
Resume	0	selectedResumePart	15	Award	14	
Person	0	hasCareer	10	Career	9	20
Resume	0	selectedResumePart	10	Career	9	
Accomplishment	0	has_CareerAccomplishments	16	CareerAccomplishments	15	16
Organization	8	hasContactInfo	9	ContactInfo	0	10
Person	0	hasContactInfo	1	ContactInfo	0	
Person	0	hasEducation	1	Education	0	2
Resume	0	selectedResumePart	1	Education	0	
Education	0	attendedInstitution	1	EducationalInstitution	0	1
Career	9	definedBy	10	Experience	0	12
Person	0	hasExperience	1	Experience	0	
Resume	0	selectedResumePart	1	Experience	0	
Career	9	definedBy	10	ExpertiseArea	0	14
Education	0	hadAreasofStudy	1	ExpertiseArea	0	
Publication	0	hasSubjectArea	1	ExpertiseArea	0	
Knowledge	0	relatedAreasofStudy	1	ExpertiseArea	0	
Resume	0	selectedResumePart	1	ExpertiseArea	0	
Career	9	workedIndustry	10	Industry	0	10
Person	0	hasKnowledge	1	Knowledge	0	1
Person	0	hasName	1	Name	0	1
Award	14	awardedBy	135	Organization	8	162
Degree	0	awardedBy	9	Organization	8	
Patents	0	awardedBy	9	Organization	8	
Experience	0	employedBy	9	Organization	8	
Person	0	hasPatents	1	Patents	0	2
Resume	0	selectedResumePart	1	Patents	0	
Person	0	hasPublications	1	Publication	0	2
Resume	0	selectedResumePart	1	Publication	0	
Person	0	hasResume	1	Resume	0	1
Experience	0	had_role	1	Role	0	1

Figure 10: Evaluation of the concepts.

3.1.3 Link Analysis: Another Way for Measuring Relevant Concepts

The Link Analysis "was born" almost simultaneously at the IMB Research Centre and at Stanford University in 1998 thanks to Kleinberg (Kle98), and Brin and Page (BP98). While Brin and Page's PageRank algorithm became the core component of the Google search engine, bringing them fame and an office in the Google's headquarters), Kleinberg's work (the HITS algorithm) was not immediately successfully commercially developed. Nevertheless, it has earned him the recognition in the mathematical community "for outstanding contributions in mathematical aspects of Information Science". Finally, in 2001, the search engine Teoma (now purchased by Ask Jeeves) adopted it as the heart of its technology. In any case, both of them completely revolutionized the way we use the web, and much more.

However, the real precursors of the link analysis *á la* PageRank and HITS, have been part of a long tradition of analytics and measures from Bibliometrics. This tradition also includes what was essentially PageRank, 22 years earlier (refer to the contribution of Pinski and Narin (PN76)), and Social Network Analysis (see for example Wassermann and Faust's book (WF94)). In particular, Pinski and Narin described a methodology to determine "influence measures" for the scientific press, authors and scientific subfields, based on citations. These methods, as we will see with PageRank and HITS, tried to assign a weight to each publishing entity so that the higher the weight, the more influence the particular entity had within the collection.

The Link Analysis is a set of methods for determining, in the web world, the relative authority of a web page on the basis of the hyperlink structure. A hyperlink is a technological capability that enables, in principle, one specific Web site to connect seamlessly with another. The shared (bilateral or unilateral) hyperlinks among Web sites allow documents and pictures to be referred to through the Web. A hyperlink between two Web sites functionally brings them closer together. These methods also aim at producing improved algorithms for the ranking of Web search results.

In our context, the idea that drives these methods can be used to provide a sort of "authority measure" to the ontology concepts as done for Web pages.

As stated in (The06), in the information science the potential for link analysis was recognized when the commercial search engine $AltaVista$[4] released an interface that allowed users to conduct various types of searches for pages containing links. Subsequently, the term *webometrics* was coined for the quantitative analysis of web-related phenomena, including links from an information science perspective (AI97). Later on, webometrics analysed search engine results (BI99) and web page changes over time (BIP04; Koe04). Although much has been written about analysis methodologies, there is no unanimity concerning the question of how to interpret link analysis research results. The question of interpretation has been addressed in published studies but typically from the perspective of individual research questions rather than from a generalized framework. Summing up, there is no clearly stated theory or methodology for link count interpretation. In (The06) an exhaustive overview is reported.
Now, we will present an overview of the two first methods previously introduced.

PageRank

PageRank, developed by Brin and Page (BP98), is a topological measure of the "prestige" of Web pages independent from the query or other information (semantics). It is based on the assumption that good quality pages are more likely to be linked than poor quality ones, and therefore that mining information about the link structure of the Web could be more effective at identifying the best pages matching search engine queries than a simple text-matching algorithm. The PageRank method can be seen as a "random walk with random reset" on graph G that represents the pages and their connections. With this view in mind, it is easy to interpret the formal representation.

Given graph G and the associated adjacency matrix $A_{n\times n}$ (a definition

[4]AltaVista: `http://www.altavista.com/`

is given in section 4.2.1), an estimation of the out-degree of a node is given by the matrix $W_{n\times n}$, where each entry is calculated as:

$$w_{ij} = \frac{a_{ij}}{\sum_{k=1}^{n} a_{ik}} \tag{3.1}$$

The random reset part of PageRank is described by a factor composed by a probability $0 < \alpha < 1$ (which determines whether we restart) and a positive $n \times n$ uniform transition probability matrix U:

$$U = \begin{bmatrix} 1/n & \dots & 1/n \\ \vdots & \ddots & \vdots \\ 1/n & \dots & 1/n \end{bmatrix} \tag{3.2}$$

Composing 3.1 and 3.2, the PageRank algorithm can be algebraically described by the matrix P:

$$P = \alpha U + (1 - \alpha)W \tag{3.3}$$

α, also called *damping factor* is, by default, set to 0.15.
Mathematically speaking, this process describes a Markov chain, and thanks to this fact it is easy to prove the convergence of the method. In particular, it converges to the dominant eigenvector of the matrix P^T (see (FLM+06) for all the details).

We can informally say that the method describes "a user surfing the web": he starts from a random page i and, with probability $(1 - \alpha)$ he follows one of the links of the current page and, with probability α, he moves to a random page. At the end, the $p_i \in P$ measures the time a probabilistic surfer lasts on page i.
PageRank is currently famous because it is adopted by the Google[5] search engine to sort all documents matching a given query, with particularly valuable results (BGS02).
Recently PageRank has been used for solving problems not related to the web environment. We cite, for example (BRdS06), where the authors describe the use of a weighted version of the algorithm to obtain a metric

[5]Google: www.google.com

that reflects the prestige of a journal. Instead of merely counting the total citations of a journal, the "importance" of each citation is determined in a PageRank fashion.

In a completely different context, there is Jiang's work (Jia06) where PageRank has been used to rank spaces or streets in order to predict how many people (pedestrians or vehicles) come to the individual spaces or streets. In their paper, they apply an extended PageRank algorithm (weighted PageRank) to the space-space topology for ranking the individual spaces, and find surprisingly that the resulting scores are significantly correlated to human movement, both pedestrian and vehicle, in four observed areas of London.

HITS: Hypertext Induced Topic Selection

HITS is a link analysis algorithm that rates Web pages based on two evaluation concepts: authority and hub. The authority estimates the content value of the page, while the hub the value of its links to other pages. In other words, as shown in figure 11, hub is a page with outgoing links and authority is a page with incoming links.

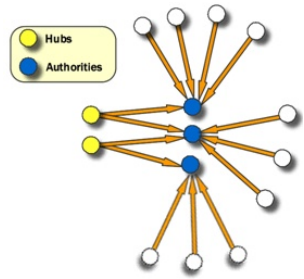


Figure 11: Hub and Autority pages.

Kleinberg observed that there exists a certain natural type of balance between hubs and authorities in the WWW graph defined by the hyper-

links, and that this fact could be exploited for discovering both types of pages simultaneously.

HITS (like PageRank) is put into an iterative algorithm applied to the subgraph G_σ of the web graph, derived from a sort of text matching procedure (for further details see the procedure `Subgraph` in (Kle98)) of the query terms σ in the search topic. For this reason it is query-dependent. The core of the algorithm starts from G_σ and computes hub ($y^{<p>}$) and authority ($x^{<p>}$) weights by using an iterative procedure qualified to mutually reinforce the values. It becomes natural to express the mutually reinforcing relationship between hubs and authorities, as: "If p points to many pages with high x-values, then it should receive a large y-value, and if p is pointed to by many pages with large y-values, then it should receive a large x-value". $\mathcal{I}$ and $\mathcal{O}$ operations have been defined for updating the weights.
$\mathcal{I}$ updates the authority x-weights as:

$$\mathcal{I}: \; x^{<p>} \leftarrow \sum_{q:(q,p)\in E} y^{<q>}.$$

$\mathcal{O}$ updates the hub y-weights as:

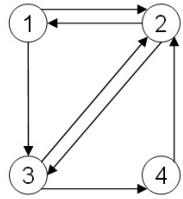
$$\mathcal{O}: \; y^{<p>} \leftarrow \sum_{q:(p,q)\in E} x^{<q>}.$$

Since the two operations are mutually recursive, a fixed point is needed for guaranteeing the termination of the computation. Even if the number k of iterations is a parameter of the algorithm, it is proven that, with arbitrarily large values of k, the sequences of vectors $x_1, x_2, \ldots, x_k$ and $y_1, y_2, \ldots, y_k$ converge to the fixed points x^* and y^* (Theorem 3.1 in (Kle98)). As one can guess, and as it happens for the main information retrieval methods, linear algebra (Aba00) supplies "tools" of support for formalizations and proofs.

First, it is possible to represent the graph G_σ in matrix form with the help of the adjacency matrix A:

$$A_{ij} = \begin{cases} 1 & \text{if an edge from } i \text{ to } j \text{ exists} \\ 0 & \text{otherwise} \end{cases}$$

For example, an adjacency matrix A associated to the small web graph G is shown in figure 12.



$$A = \begin{array}{c} \\ P_1 \\ P_2 \\ P_3 \\ P_4 \end{array} \begin{array}{c} \begin{array}{cccc} P_1 & P_2 & P_3 & P_4 \end{array} \\ \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{bmatrix} \end{array}$$

Figure 12: A small web graph with its adjacency matrix associated.

Then, one can easily observe that, the iterative and mutual call of $\mathcal{I}$ and $\mathcal{O}$, can be (re)written as:

$$\begin{aligned} x_i &= A^T y_{i-1} \\ y_i &= A x_i \end{aligned} \tag{3.4}$$

Stated that, it is easy to trace the computation of x^* and y^* back to the mathematical computation of the principal eigenvector [6] of a matrix $A^T A$ and AA^T, respectively. From 3.4, after k iterations, we obtain

$$\begin{aligned} x^{(k)} &= (A^T A)^{(k-1)} A^T \mathbf{u} \\ y^{(k)} &= (AA^T)^{(k)} \mathbf{u} \end{aligned} \tag{3.5}$$

where $\mathbf{u}$ is the initial seed vector for x and y.

Example 3.2 *HITS Example.*

Let us suppose that the subset of nodes that contains the query terms is $\{1, 6\}$. Let us suppose again that the subgraph G of the whole web graph is the one depicted in figure 13.

[6] It should be useful to recall that, if M is a symmetric nxn matrix, an *eigenvalue* of M is a number λ with the property that, for some vector ω, we have $M\omega = \lambda\omega$. The set of all such ω is a subspace of $\mathbb{R}^n$, which we refer to as the *eigenspace* associated with λ; the dimension of this space will be referred to as the multiplicity of λ. It is a standard fact that M has at most n distinct eigenvalues, each of them a real number, and the sum of their multiplicities is exactly n.

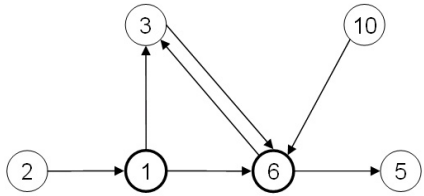


Figure 13: Subgraph

The adjacency matrix A associated with G is the following:

$$A = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 5 \\ 6 \\ 10 \end{array} \begin{array}{c} \begin{array}{cccccc} 1 & 2 & 3 & 5 & 6 & 10 \end{array} \\ \begin{bmatrix} 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \end{array}$$

The corresponding authority and hub matrices are:

$$A^T A = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 5 \\ 6 \\ 10 \end{array} \begin{array}{c} \begin{array}{cccccc} 1 & 2 & 3 & 5 & 6 & 10 \end{array} \\ \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

and

$$AA^T = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 5 \\ 6 \\ 10 \end{array} \begin{array}{c} \begin{array}{cccccc} 1 & 2 & 3 & 5 & 6 & 10 \end{array} \\ \begin{bmatrix} 2 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 2 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} \end{array}$$

The characteristic polynomial of $A^T A$ is $\lambda^2(\lambda-1)(\lambda^3-6\lambda^2+9\lambda-2)$ and the correspondent eigenvalues are $\lambda_1 = 0$, $\lambda_2 = 1$, $\lambda_3 = 2-\sqrt{3}$, $\lambda_4 = 2$

and $\lambda_5 = 2 + \sqrt{3}$.
The characteristic polynomial of AA^T is $\lambda^2(\lambda-1)(\lambda-2)(\lambda^2-4\lambda+1)$ and the correspondent eigenvalues are $\lambda_1 = 0$, $\lambda_2 = 1$, $\lambda_3 = 2 - \sqrt{3}$, $\lambda_4 = 2$ and $\lambda_5 = 2 + \sqrt{3}$.
The normalized[7] eigenvectors related to authority and hub and associated to the dominant eigenvalue ($\lambda_5 = 2 + \sqrt{3}$), are:

$$x^T = [0, 0, 0.3666, 0.134, 0.5, 0]$$
$$y^T = [0.3660, 0, 0.2113, 0, 0.2113, 0.2113]$$

This means that node (or page) 6 is the most authoritative for the query, while node (or page) 1 is the best hub for this query.

◁

Kleinberg made the assumption that a unique eigenvalue exists, i.e. that $|\lambda_1(M)| > |\lambda_j(M)$ for each $1 < j \leq n$.
The key point is that the dominant eigenvalue of a matrix is not always unique. When it is so, the (HITS) convergence is valid but, when the dominant eigenvalue is repeated, the ranking vector could be any non-negative vector in the multidimensional dominant eigenspace. In particular, the ranking vector may not be unique and may depend on the initial seed vector. The following example 3.3, taken from (FLM+06), should clarify this.

Example 3.3 *Repeated eigenvalues.*

Given the web graph of figure 14 and the correspondent associated adjacency matrix A, we apply the algorithm as in equation 3.5. By using the uniform vector $y_0 = [1/\sqrt{6}, \ldots, 1/\sqrt{6}]$ as seed, we obtain the following authority and hub vectors:

$$x = [2/\sqrt{5}, 1/2\sqrt{5}, 1/2\sqrt{5}, 1/2\sqrt{5}, 1/2\sqrt{5}, 0]$$
$$y = [0, 1/\sqrt{5}, 1/\sqrt{5}, 1/\sqrt{5}, 1/\sqrt{5}, 1/\sqrt{5}].$$

This would mean that vertex 6 is not a better hub than vertices 2, 3, 4 and 5, and that we would obviously expect a different result.
Let us try, instead, to use the seed vector x_0 rather than y_0, and let us see how hub and authority vectors change.

[7]The normalization is done by using the *norm 1*, i.e. $\|x\|_1 = \sum_{i=1}^{n} |x_i|$.

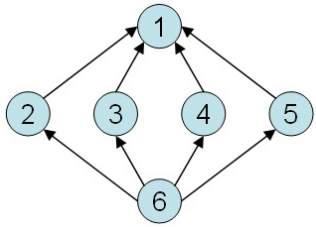


Figure 14: A web graph with non-unique largest eigenvalue for $A^T A$.

$x = [1/\sqrt{5}, 1/\sqrt{5}, 1/\sqrt{5}, 1/\sqrt{5}, 1/\sqrt{5}, 0]$
$y = [0, 1/2\sqrt{5}, 1/2\sqrt{5}, 1/2\sqrt{5}, 1/2\sqrt{5}, 2/\sqrt{5}]$.

This would mean that vertex 1 is now not a better authority than vertices 2, 3, 4 and 5.
The output is sensitive to the initial seed vector because the dominant eigenvalue is not unique. In fact:

$DET(A^T A) = \lambda^4(\lambda - 4)^2 \Rightarrow$
$\lambda_1 = \cdots = \lambda_4 = 0$
$\lambda_5 = \lambda_6 = 4.$

◁

Unfortunately, HITS suffers another weakness in the presence of badly-connected graphs: it can return ranking vectors that inappropriately assign 0-weights to part of the network that should gain a higher rank. From a mathematical point of view, this happens when the dominant eigenvalue is unique but there are "lower" eigenvalues repeated. Example 3.4 shows such a case.

Example 3.4 *Repeated lower eigenvalues.*
Suppose we now have the web graph of figure 15 and the corresponding associated adjacency matrix A. In this case the characteristic polynomial for $(A^T A)$ is:

$DET(A^T A) = \lambda^5(\lambda - 3)(\lambda - 2)^2 \Rightarrow$
$\lambda_1 = \cdots = \lambda_5 = 0$
$\lambda_6 = \lambda_7 = 2$
$\lambda_6 = 3.$

The dominant eigenvalue is simple ($\lambda_6 = 3$) but eigenvalues 0 and 2 are repeated. In this case, as long as the initial seed vector is positive, the output of HITS is:

$x = [0, 1, 0, 0, 0, 0, 0, 0]$ and
$y = [0, 0, 0, 1/\sqrt{3}, 1/\sqrt{3}, 1/\sqrt{3}, 0, 0]$.

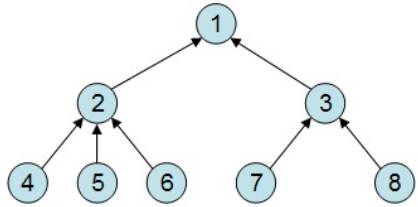


Figure 15: A web graph with simple largest eigenvalue but repeated lower ones, for $A^T A$.

Only node 2 gets a positive authority. We would expect that node 1 should be a better authority than any other node because it is accessible from every node. Meanwhile, node 3 should be only slightly less important than node 2 since there is only one more node pointing to node 2.

◁

The two limitations just discussed can be overcome by applying the strategy of Farhat et all. In (FLM$^+$06) they propose to replace the simple adjacency matrix A associated to the web graph with the *Exponentiated Adjacency Matrix* able to give information about paths longer than 1. In this way, during the computation of the hub score of a page, each iteration HITS does not only look at the authority scores of adjacent pages, but also considers "indirect connections". The idea starts from the observation that, if A identifies the paths of length 1 from each couple of nodes, then A^2 identifies the path of length 2 up to A^m that identifies the path of length m. Considering paths of length 1 more important than paths of greater length, the proposed exponentiated matrix has the following format:

$$A + A^2/2! + A^3/3! + \cdots + A^m/m! + \cdots = e^A - I, \tag{3.6}$$

where I is the identity matrix.
This series converges because each entry in the m^{th} term is bounded by $n^m/m!$. Also, the matrices $A + A^2/2$ or $I + A$ would overcome the weaknesses.
We omit the details because our aim is to exploit this important result for our problem rather than to go back again over the proofs.
In section 4.2 we will show how this work helps us in the task of extracting the relevant concepts from an ontology.

Another interesting study that "justifies" the generalisation of Kleinberg's ideas to other domains besides the web has further driven us to improve our work. Agosti and Pretto in (AP05) present a theoretical study of a generalized version of HITS. They perform a mathematical analysis by focusing on the convergence of the algorithm, examining it in a general "abstract" context, independently of the web application.

The result that interests us is the possibility of giving positive weight to the edges of the graph, maintaining the correctness and termination requirements related to the HITS computation. This fact enables us to semantically enrich the graph, and to model different problems. Essentially it maintains the procedures but it changes the input representation replacing the simple adjacency matrix A with a "weighted adjacency matrix" W.
As one can imagine, $W = [w_{ij}]$ is an $n \times n$ matrix whose the generic entry (i, j) is:

$$w_{ij} = \begin{cases} e_{ij} & \text{if an edge from } i \text{ to } j \text{ has the positive weight } e_{ij} \\ 0 & \text{if no edge exists from } i \text{ to } j \end{cases} \tag{3.7}$$

In this revised formulation, the mathematical representation of the iterative procedure of the algorithm becomes:

$$\begin{aligned} x^{(k)} &= (W^T W)^{(k-1)} W^T \mathbf{u} \\ y^{(k)} &= (W W^T)^{(k)} \mathbf{u} \end{aligned} \tag{3.8}$$

Theorem 4.1 in (AP05) guarantees that matrix $B = W^T W$ has a strictly dominant eigenvalue if and only if one of the matrices $B_1, B_2, \ldots, B_m$ has

a strictly dominant eigenvalue that is greater than the strictly dominant eigenvalue of every other matrix of this multiset of matrices. B is a matrix with diagonal blocks with this structure:

$$B = \begin{bmatrix} B_1 & 0 & \dots & 0 \\ 0 & B_2 & \dots & 0 \\ \vdots & & \ddots & \\ 0 & 0 & \dots & B_m \end{bmatrix} \tag{3.9}$$

where B_i is the matrix having as an underlying graph the weighted component i.
Omitting the proofs, we can point out that the convergence of the revised HITS guarantees that $x^{(k)}$ and $y^{(k)}$ converge to the unit authority and hub vectors which are linear combinations of all the dominant eigenvectors of $W^T W$ and WW^T, respectively.

3.1.4 Semantic Association Discovery: a Promising Approach

One of the approaches for retrieving information from ontologies and in general from the Semantic Web is to search for relations between concepts. The simple (direct) relations such as the *is-a* or *is-part-of* relations can be found easily also by using a query language (KAC+02). This means that all descending classes of one class can be retrieved even on a different level. However, in the Semantic Web there are more complex relationships among entities; such complex relationships can be represented by a path between two entities consisting of other entities and their properties. An interesting contribution can be found in (AS02; Bar04) where the authors propose and implement a class of operators (called $\rho{-}operators$) for discovering semantic associations over RDF. This class contains $\rho\ path$, $\rho\ connect$ and $\rho\ iso$ operators that have the following meaning:

ρ **path** returns all paths between two entities in the graph (ontology schema).

ρ **connect** returns all intersecting paths on which two entities lie.

ρ **iso** implies a similarity of nodes and edges along the path, and returns such similar paths between entities.

This work seems to come very close to our approach and it proposes interesting solutions to common issues such as how to extract information from an ontology and how to find hidden relationships among concepts. For this reason we report the example 3.5 (taken from (AS02)) with the aim of clarifying the power of these operators.

Example 3.5 *ρ-operators on RDF graph*

Suppose we have the ontology (schema and instances) of figure 16. The ontology describes the association between Artists and Museums. Thanks to ρ operators we can discover the following information.
(i) Relation between resources &r6 and &r8. Such an association represents a piece of information that a painter called Pablo Picasso had painted a painting that is exhibited in the Reina Sofia Museum.
(ii) Intersection between resources &r6 and &r8 and between resources &r9 and &r8. This association represents the fact that two artists had their artefacts (a painting and a sculpture respectively) exhibited in the same museum.
(iii) Relation between resources &r1 and &r9. This represents a fact that both subjects are classified as painters.

$\triangleleft$

Beyond this simple example, an important usage of searching such complex associations can be found in the field of national security for identifying suspicious passengers at airports. This is done by looking for available connections between them or in the field of business intelligent applications. Note that it is not possible to always encode this kind of association as inference rules because they cannot be known a priori, but instead discovered with experience. Then, it is important to develop methods for identifying or discovering associations by using general and domain-independent characteristics, and then by applying domain knowledge to guide the search process, allowing the search to focus on only associations that are important in that context.
As one can notice, the main strategy makes use of ontology instances besides the only ontology schema, but for the first step of our research it does not seem to be useful. Furthermore, we think that the general idea

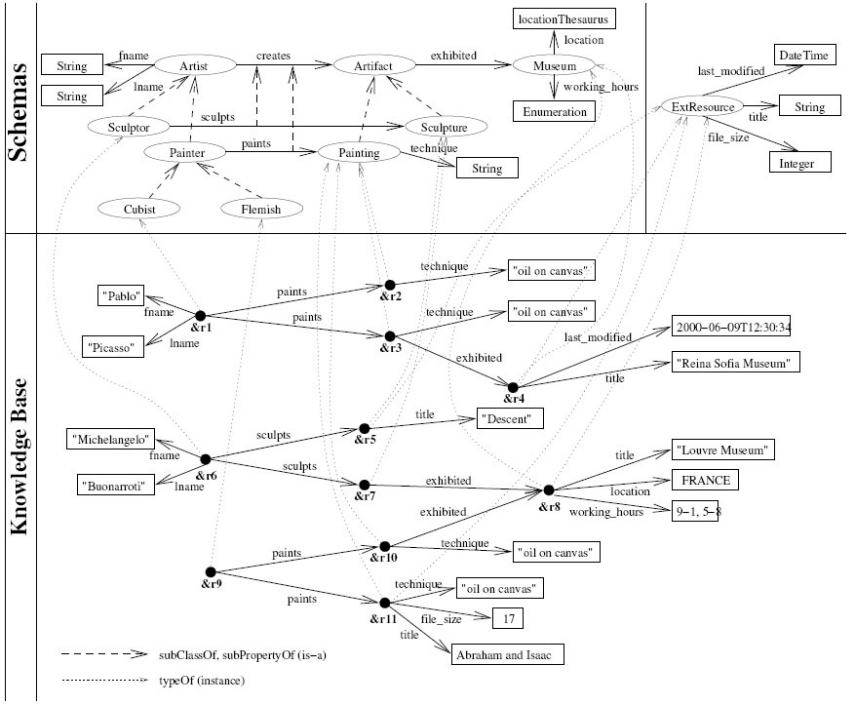


Figure 16: Example of Rho Operators on RDF Ontology

could be exploited in intermediate phases and we would like to study this approach in depth.

3.1.5 Multi-Relational Data Mining

As we have already described in section 2.1, traditional DM looks for patterns in data and a lot of existing approaches known as Propositional DM look for patterns in a single data table. Most real-world databases, however, store information in multiple tables. Records in each table represent parts, and individuals can be reconstructed by joining over the foreign key relations between the tables. Multi-Relational Data Mining (MRDM) approaches act on this structure and look for patterns that involve multiple tables (relations) from a relational database. When we are looking for patterns in multi-relational data, it is natural that the patterns involve multiple relations.

It is known that there is the growing interest in the development of the DM algorithms for various types of structured data, for example, graph-based data mining, and mining of tree-structured and XML documents. Some interesting improvements are in the web search (CMH03) and in Molecular Biology (FM04).
We already discussed in section 3.1.1 the theoretical bases of graph-based data mining that is actually a case of Structural Data Mining like MRDM. We will discuss now some fundamentals of the MRDM so that it will be easier to catch the similarity with the ontology mining strategy that we are proposing.

MRDM can analyse data from a multi-relation database directly, without the need to transfer the data into a single table first. Thus the mined relations can reside in a relational or deductive database. The relational data can be represented by means of tables (extensional view) or explicit logical rules (intentional view that consists of relationships that can be inferred from other relationships). The corresponding relational patterns extend the types of propositional patterns considered into a single table. They are stated in a more expressive language than patterns defined into a single table, so we have relational classification rules, relational asso-

ciation rules, and so on. They are typically expressed by using a subset of first-order logic. The following example 3.6, taken from (Dze03), will clarify the concepts.

Example 3.6 *Multi relational data mining.*

Let us consider the relational database in figure 17 composed of two tables: `Customer` contains the general information of a customer, and `MarriedTo` contains the matrimonial relationships. The relational clas-

Customer table

ID	Gender	Age	Income	TotalSpent	BigS
c1	Male	30	214000	18800	Yes
c2	Female	19	139000	15100	Yes
c3	Male	55	50000	12400	No
c4	Female	48	26000	8600	No
c5	Male	63	191000	28100	Yes
c6	Male	63	114000	20400	Yes
c7	Male	58	38000	11800	No
c8	Male	22	39000	5700	No
...	...	...	...	...	...

MarriedTo table

Spouse1	Spouse2
c1	c2
c2	c1
c3	c4
c4	c3
c5	c12
c6	c14
...	...

Figure 17: Relational Database.

sification rule, predicting the feature of a person to be a big spender and that involves both tables, is the following:

$$\texttt{big\_spender}(C_a, \text{Age}_a, \text{Income}_a, \text{TotalSpent}_a) \leftarrow$$
$$\texttt{married\_to}(C_a, C_b) \wedge$$
$$\texttt{customer}(C_b, \text{Age}_b, \text{Income}_b, \text{TotalSpent}_b, \text{BS}_b) \wedge$$
$$\text{Income}_b \geq 108000$$

In agreement with the rule, a big spender is one who is married with a person with a high income. As we can see the rule uses predicates; `big_spender` defines the "goal feature" while `married_to` and `customer` belongs to the tables Customer and MarriedTo, respectively.

Instead, a propositional rule is defined on a single table and it does not involve predicates. The following is an example:

IF Income > 108000 THEN BigSpender = Yes.

This example clearly shows how relational patterns are more expressive than propositional ones.

◁

An alternative approach to MRDM (called *propositionalization*) consists in the creation of a single table from a multi-relational database in a systematic fashion by joining the tables. This system permits one to apply to the resulting unique table, the traditional data mining algorithms. What is the actual benefit in making efforts to update the existing algorithms in order to handle relational databases and in order to increase the computational complexity?
The answer is the simplest and the most reasonable one: for one-to-one and many-to-one relations, we can join the extra fields to the original relation without problems, but for one-to-many relations, problems occur either in loss of meaning or in loss of information through aggregation. Consider for example the relations:

`customer`(CustID, Name, Age, SpendsALot)
`purchase`(CustID, ProductID, Date, Value, PaymentMode)

where each customer can make multiple purchases, and suppose we are interested in characterizing customers that spend a lot. Merging the two tables via natural join, builds a relation `purchase1` where each row corresponds to a purchase and not to a customer. One possible aggregation would give rise, for example, to the relation:

`customer1`(CustID, Age, NofPurchases, TotalValue,
SpendsALot)

with loss of information. In fact, it is not possible to induce the pattern:

`customer`(CID, Name, Age, yes) ←
Age > 30 ∧
`purchase`(CID, PID, D, Value, PM) ∧
PM = credit_card ∧ Value > 100

from either of the relations `purchase1` and `customer1` considered on their own.
Clearly, these approaches benefit of the efficiency of the traditional DM algorithms but have limited expressiveness.

If we represent a relational database by means of a graphical formalism, for example the Entity-Relation diagram, a certain similarity with graphs (and why not, with an ontology graph) immediately rises. Each database table (relation) can be seen as a node and the corresponding items as the associated properties or as datatype properties if speaking in terms of ontologies. Primary and foreign keys can be seen as edges (object properties on the other hand) between the nodes.
Without going into detail, in general, MRDM algorithms traverse the graph of tables and associations, that makes the data model a strong way of guiding the search process. One can simply see here the similarity between the exploration of a graph that maybe represents an ontology and the search of patterns in a data base with multiple tables.

As is the case of many DM algorithms originating from the field of machine learning, many MRDM algorithms come from the field of Inductive Logic Programming (ILP) that is situated at the intersection of machine learning and logic programming. Initially, ILP focussed on automated program synthesis from examples, formulated as a binary classification task, but in recent years its scope has broadened to cover the whole spectrum of DM tasks such as classification, regression, clustering and association analysis.

The ILP paradigm employs (small) logic programs to describe patterns. The logic programs need to be induced from a database of logical facts. The facts in the database represent parts of the structured individuals, while the class of each part can be identified by the predicate of the fact. The individuals are often identified by means of group or facts. ILP typically uses a logic language as the pattern language, for example Prolog. ILP algorithms often allow as input not only the database of ground facts, but also intentional predicate definitions that may help the search process for interesting patterns.

3.2 How to Compute Weights and How to Associate Them to Implications?

This question corresponds to the characterization of the implications discovered at schema level during the deductive step (see section 3.1.3 for theoretical aspects and section 4.2 for technical details). Since we have not exploited the data until now, the use of the instances is now essential. The idea is to perform an inductive step of analysis over the knowledge base where the ontology instances are stored. In general, we can presume that the instances are stored in a repository separate from the ontology; if they are directly part of the ontology, instead, an intermediate step is needed for retrieving them.

The approach that seems to be more suitable for characterizing the implications consists in the use of one of the consolidated algorithms for knowledge discovery, and in particular, methods for Association Rule Discovery (AIS93; AS94; AMS+96). Actually, as we will see later, for our purposes it will be sufficient to stop the process after the discovery of the frequent patterns.

The goal of these techniques is to detect relationships or associations between specific values of categorical variables in large data sets, that is, to show attribute value conditions that occur frequently together in a given dataset. A typical well known and widely-used application example of AR mining is the Market Basket Analysis (AIS93). In short, given a large number of items (articles from a supermarket like *bread*, *milk*, *napkins*, etc...), the market basket problem aims at discovering what items the customers buy together, even if we do not know who the customers are.

The formal definition follows.

Let $I = I_1, I_2, \ldots, I_m$ be a set of binary attributes, called items. Let T be a database of transactions. Each transaction t is represented as a binary vector, with $t[k] = 1$ if t bought the item I_k, and $t[k] = 0$ otherwise. There is one tuple in the database for each transaction. Let X be a set of some items in I. We say that a transaction t satisfies X if for all items I_k in X, $t[k] = 1$.

An AR is an implication of the form $X \Rightarrow I_j$, where X is a set of some

items in I, and I_j is a single item in I that is not present in X. The rule $X \Rightarrow I_j$ is satisfied in the set of transactions T with the *confidence* factor $0 \leq c \leq 1$ if at least $c\%$ of transactions in T that satisfy X also satisfy I_j.
The intuitive meaning is that, if the items X are found together in the itemset, then there is a good chance (the confidence factor) of finding also the item I_j.
Normally we search only for rules that had confidence above a certain threshold.

Beside the confidence, another important measure of a rule is the *support*. The support is the fraction of transactions in T that satisfy the union of items in the consequent and antecedent of the rule. Notice that the support should not be confused with the confidence. While the latter is a measure of the rule's strength, the former corresponds to statistical significance (please refer to (AIS93; AS94; AMS+96) for an exhaustive treatment).

The ARs mining problem can be decomposed into two subproblems:

P1 Generate all combinations of items that have fractional transaction support above a certain threshold, called *minimum support*.

P2 For a given large itemset $Y = I_1, I_2, \ldots I_k$, with $k \geq 2$, generate all rules (at the most k rules) that use items from the set $I_1, I_2, \ldots I_k$.

This problem has been extensively studied in the last years. Several variations to the original A-priori algorithm (AS94), as well as completely different approaches, have been proposed. All the proposed algorithms browse bottom-up the huge solution search space, i.e. the lattice graph of $2^{|I|}$ itemsets, by exploiting various strategies for pruning it. Among these strategies, the most effective regards the exploitation of the downward closure property: if a k-itemset is frequent, then all of its $k-1$-subsets have to be frequent as well. On the other hand, if a k-itemset is discovered to be infrequent, then all its supersets will not be frequent as well.

For completeness, we report the main steps of the AR mining process, in its standard formulation associated to the A-priori algorithm, just cited.

Step 1 Scan the transaction database to get the support S of each 1-itemset, compare S with the minimum support (*min_sup*) threshold, and get a set of frequent 1-itemsets, $L1$.

Step 2 Join L_{k-1} with itself for generating a set of candidate k-itemsets. Use A-priori property to prune the infrequent k-itemsets from this set.

A-priori Property

- Reduce the search space to avoid finding of each L_k requires one full scan of the database.
- If an itemset I does not satisfy min_sup, then the I is not frequent.
- If an item A is added to the itemset I, then the resulting itemset (i.e., $I \cup A$) cannot occur more frequently than I. Therefore, $I \cup A$ is not frequent either.

Step 3 Scan the transaction database to get the support S of each candidate k-itemset in the final set, compare S with min_sup, and get a set of frequent k-itemsets, L_k.

Step 4 If the candidate set is empty, then go ahead to Step 5, else go back to Step 2.

Step 5 For each frequent itemset l, generate all non-empty subsets of l.

Step 6 For every non-empty subset s of l, output the rule: "$s \Rightarrow (l - s)$".

The following example 3.7, shows an application of the algorithm.

Example 3.7 *Running the A-Priori algorithm.*

Suppose we have table (a) of figure 18 that describes transactions. Each transaction is identified by a code with which is associated the list of items. Let us see how the process generates the frequent itemsets from which the AR set can be derived. Applying Step 1, the algorithm generates table (b) in which the association between (single) items and the number of occurrences (sup_count) is shown. By joining table (b) with itself (in agreement with Step 2), both the 2-Itemset candidates and the

support count are computed (table (c)). Having fixed the minimum support min_supp = 2, the pruning procedure eliminates the itemset with sup_count < 2 (table (d) is the result). Since from table (d) a new list of candidates can be computed, the process re-starts from Step 2 until the frequent 3-Itemsets are obtained (table (e)).

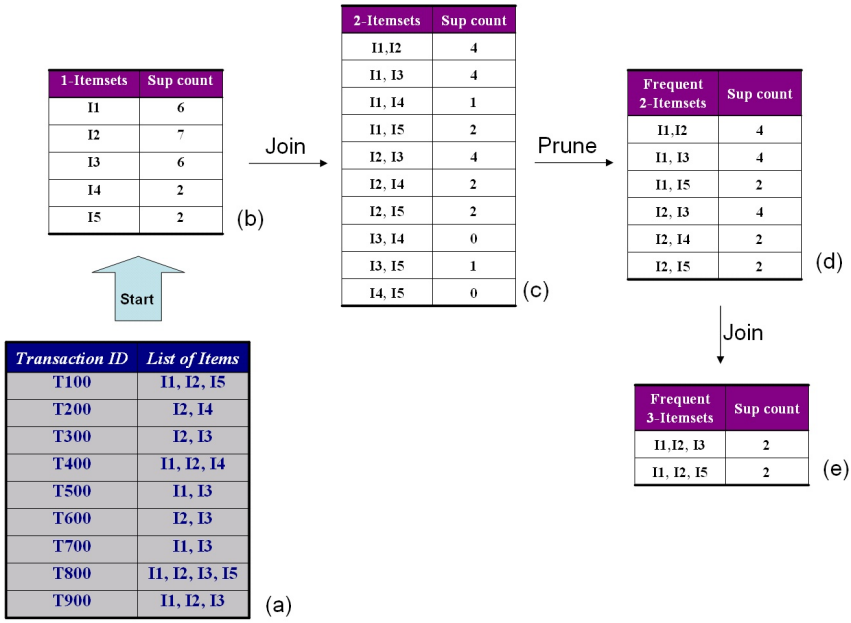


Figure 18: Steps of the Frequent Itemsets generation.

◁

As already mentioned, the result we are interested in corresponds to the output of sub-problem $P1$ because at this stage we have all the elements for characterizing the rule schemas (values for the attributes and a probability estimation). This statement, that might appear obscure here, will be clarified during the explanation of the methodology in section 4.4.

For our purposes we use PATTERNIST, a pattern discovery algorithm

developed at the CNR[8] in Pisa by colleagues of the ISTI[9] department, within the project P^{3}D[10]. PATTERNIST is the result of a research activity that has now come to the implementation of a more sophisticated (and documented) system: ConQueSt (BGL$^+$06; BL05; BL04).

ConQueSt is a constraint-based querying system devised with the aim of supporting the intrinsically exploratory nature of pattern discovery. It provides users with an expressive constraint-based query language which allows the discovery process to be effectively driven toward potentially interesting patterns. Constraints are also exploited to reduce the cost of pattern mining. The system is built around an efficient constraint-based mining engine which entails several data and search space reduction techniques, and allows new user-defined constraints to be easily added.
As stated, PATTERNIST represents the "forefather" of the current core component of ConQueSt dedicated to the computation of the frequent itemsets. For our purposes, the whole ConQueSt system is more complex than required, so we actually use the basic implementation of PATTERNIST now available only for the didactics and that circulates among colleagues.

In order to answer the original question put by the title of this paragraph, let us consider again the example depicted in figure 6, section 3.1. Let us again suppose that we have extracted the following rule schema by means of the analysis of the ontology structure:

$$Manager(X) \wedge has\_age(X, N) \wedge leads(X, P) \longrightarrow has\_innovation\_degree(P, V)$$

This rule sounds like:

> *"If a manager has a certain age, then the project he manages has a certain degree of innovation."*.

[8] Consiglio Nazionale delle Ricerche: `http://www.cnr.it/`

[9] Institute of information science and technology "Alessandro Faedo": `http://www.isti.cnr.it/`

[10] P^{3}D: *Privacy Preserving Pattern Discovery* is a project developed by the Knowledge Discovery and Delivery Laboratory together with the High Performance Computing Laboratory with the aim of designing a frequent pattern query engine within a privacy preserving environment.

Now, what we would like to obtain from the inductive step (i.e., by applying PATTERNIST to the set of ontology instances), is the assignment of values for the attributes V and N on the basis of the data frequencies, and a weight for the implication on the basis of the support measure. A characterization for the implication shown in the example could be the following:

$$Manager(X) \wedge has\_age(X, N) \wedge (N < 40) \wedge leads(X, P) \xrightarrow{0.6} has\_innovation\_degree(P, good)$$

The intuitive meaning is:

> *"If the manager's age is lower than* 40*, than the innovation degree of the project he manages is good with a probability of 60%".*

The details are provided in the core chapter 4, section 4.4.

3.3 Semantic Web: Social Networks, Ontologies or Both?

Nowadays, when we say Social Networks, we immediately think of one of the platforms accessed by people of every age, whether for meeting other people or for sharing information: Social Video Sharing (YouTube), Social Photo Sharing (Flickr), Social Community (MySpace, Facebook, LinkedIn), Social Bookmarking (Delicious) Social Encyclopedia (Wikipedia), Social Music Community (LastFM), etc ...
We can assert that the way of consuming information has changed because of or thanks to, the birth of the Web 2.0 and, consequently, to the creation of these networks. Especially in the younger generations, people do not necessarily go into bookshops or libraries to increase their general knowledge, but instead they prefer to spend time on Google, MySpace, Facebook, Yahoo, MSN or authoring Web sites. Surfers have converted themselves from spectators to actors.

Statistics, taken from (BH06), report that the market share of Internet visits to the top 20 social networking websites grew by 11.5% from January to February 2007, to account for 6.5% of all Internet visits in February

2007.

In the third decade, just started, the information available on-line is becoming understandable also to computers; in that way it can be linked, merged, processed and re-used under other "forms" (e.g. originating other information) by means of automatic tools.
Technically speaking, a Social Network (SN) is a model where social entities such as people and organisations, happenings such as events and finally locations, are connected to each other by certain relationships at a certain time. More precisely, it can be defined as a social structure, community, or society made of nodes that are generally accounts, individuals or organizations. It shows the ways in which they are connected through various social familiarities, affiliations and/or relationships ranging from casual acquaintance to close familial bonds (figure 19).
The Social Network Analysis (SNA) is thus the mapping and measuring

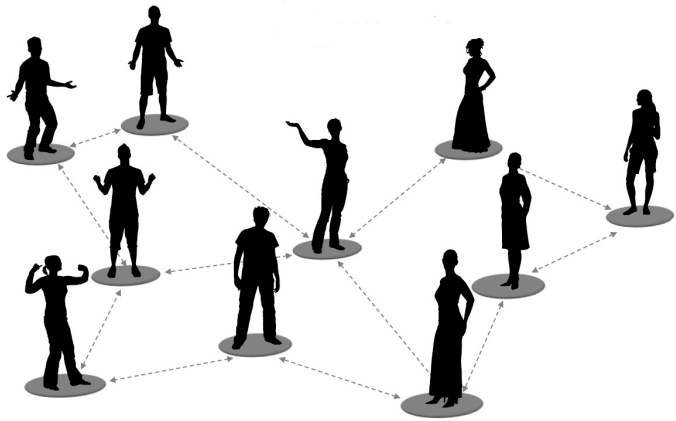


Figure 19: Social Network: visualization.

of relationships and flows between people, groups, organizations, animals, computers or other information/knowledge processing entities.
As for the ontology, also the SNA originates from a more "ancient" science: Sociology (1700). We passed though Modern Sociology and So-

cioeconomics (1800), Formal Sociology (1900), Sociometry (1950) before speaking explicitly of SNA (1990).

One of the first examples of SNA was the study conducted by Dr. Stanley Milgram, an American social psychologist at Yale University. In 1967, he conduced the small-world experiment that is the basis of the *six degrees of separation* concept. Milgram sent several packages to random people in the United States, asking them to forward the package, by hand, to someone specific or to someone who is more likely to know the target. The average path length for the received packages was around 5.5 or 6, resulting in widespread acceptance for the term "six degrees of separation".
After that, but before the advent of the massive use of the internet, the British anthropologist and evolutionary biologist Robin Dunbar, who specialized in primate behaviour, theorized a limit to the number of people with whom one can maintain stable social relationships, the so called "Dunbar's number" (Dun92). According to Dunbar, as brains evolve, they become larger in order to handle the unique complexities of larger social groups. Humans have the largest social groups because they have the largest cortex. For this reason, he developed an equation, which works for most primates, in which he plugged in what he calls the *neocortex ratio* of a particular species (the size of the neo-cortex relative to the size of the brain), and the equation gives the maximum expected group size for each species. For humans, the maximum group size is 147.8, or about 150.

One can ask whether the internet and the (Virtual) Social Networks can change this theory. The answer, I think, is an open question.

Going back to the technical aspect, the nodes in the network are the people and groups, while the links show relationships or flows between the nodes. SNA provides both a visual and a mathematical analysis of human relationships. The relationships among people are distinguished in kinship (i.e. *father of*, *wife of*), other role-based (i.e. *boss of*, *teacher of*, *friend of*), cognitive (i.e. *knows*, *aware of*), affective (*likes*, *trusts*) and interactions (*give advice*, *talks to*, *fights with*). While the relationship examples between people and organisations could be: *buy from / sell to*, *leases*, *owns*

shares of, *subsidiary of*, *is leader of* or *is founder of*.

The notion of time is another important aspect to take into account during the design of SNs. Events happen at a given time; states of affairs evolve over time such that places that are present now may not exist later; people who are alive today may be dead tomorrow, or organizations that did not exist before may come into being today.
Already in this first definition, the similarity between SNs and ontologies clearly rises: the ontology is commonly made for the specification and explication of concepts and relationships related to a given domain, while a SN has the same purpose but with the focus on social relations and entities. SNs are often modelled via ontologies because an ontology through reasoning and inference mechanisms do not allow for the definition of contradictory or inconsistent information. Thanks to the inference and to the definition of rules, the discovery of new hidden information is possible.

As an example, we cite the recent work of Jung and Euzenat (JE07) in which it is stated that, since the goal of SNA is to help users to take advantages from SNs, it would be convenient to take more information into account. With that aim, the authors introduce a three-layered model which involves the network between people (social network, $\mathcal{S}$), the network between the ontologies they use (ontology network, $\mathcal{O}$) and a network between concepts occurring in these ontologies (concept network, $\mathcal{C}$). The $\mathcal{O}$ layer shows what the ontology under analysis are, and how they are related. The typical relations are the inclusion and the import. This layer is "pointed" by the upper $\mathcal{S}$ layer. In this way for each individual (a physical person), we know the ontology it belongs to (i.e. where it is defined). The $\mathcal{C}$ layer describes the concepts, the hierarchies of concepts and shows the properties that binds them.
Figure 20 shows an instance of a three-layered social semantic network.
It is not surprising that, even in this context, the metrics used for measuring the individual tendency to be collaborative with each other is based on the Hub and Authority concepts introduced by Kleinberg.

The objective relationship from the $\mathcal{S}$ to $\mathcal{O}$ is through the explicit usage of an ontology by a user, while the one from $\mathcal{O}$ to $\mathcal{C}$ is through the

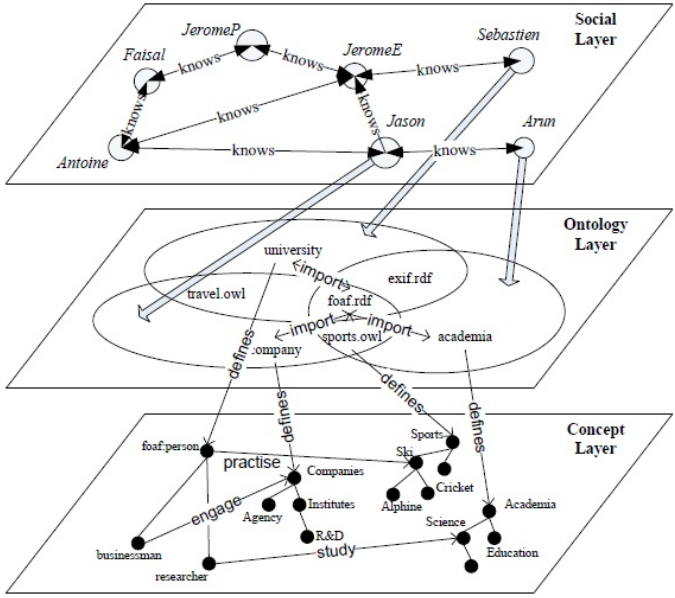


Figure 20: An instance of a three-layered social semantic network.

definition of a concept in an ontology. In order to exploit the actual potential of this model, it is necessary to analyse these nets and to propagate information between the layers. Without going much into detail, in the $\mathcal{C}$ layer the interesting relationship to infer is the similarity because, in order to find relationships between concepts from different ontologies, it is necessary to identify the entities denoting the same concepts. Also, at the $\mathcal{O}$ layer level a distance metrics is defined: it provides a good idea of the distance between two ontologies. Once these measures on ontologies are obtained, this distance is further used on the $\mathcal{S}$ layer, assuming that people using the same ontologies should be close to each other. Please refer to (JE07) for all the details about the strategy and the exact metrics.

This is only an example of how ontologies and SNs are related, and how they can be used for extracting new implicit knowledge; the research is very active in this context. On the other hand, we are quite confident that the approach we are presenting could be applied for discovering social relationships in a social context.

Chapter 4

Extracting New Knowledge from the Ontology

This chapter represents the core of the thesis. Referring to the four-steps analysis sketched in chapter 1, and also to the available alternatives proposed in chapter 3, we discuss here the solutions that we have adopted and what we think are the more suitable ways of solving the problem of extracting new knowledge from an ontology. We then present the result in IRs form. After having summarized some technical aspects of the strategy (section 4.1), we dedicate the remaining sections to an accurate description of choices we did, methodologies, algorithms and technical details. The algorithms, described here by showing the pseudo-code, have been implemented, and the software of the whole system has been realized and found to work.
The whole presentation is supported by a "running example" that is updated little by little starting from example 4.1 until example 4.5. The aim of these examples is to clarify each step of the analysis and to show the intermediate results. We are aware that the example could seem too much simplistically structured to an expert reader, but the objective is to exemplify the contents rather than to show the advantages of our method. The actual power of the system will be discussed in detail in chapter 5 during the discussion of the case study (the MUSING project).

4.1 The Strategy

While in the previous sections (see chapter 1 and 3) we gave a rough outline of the approach, in this section we will look at all the main steps in detail (figure 21).
We split the whole procedure into four main tasks each one dedicated to a particular phase of the extraction.

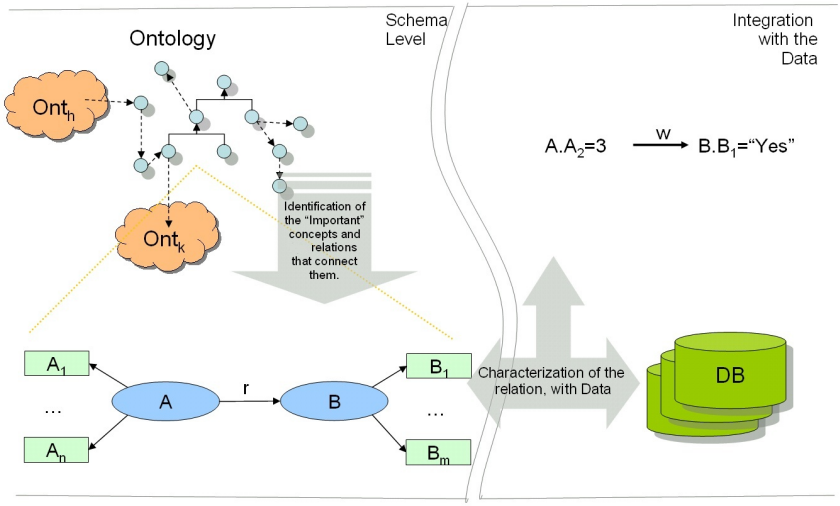


Figure 21: Steps of analysis.

[Step 1] Identification of the concepts.

Analysis of the ontology schema and extraction of the most relevant concepts.

For the extraction, we exploit the possibility of representing the ontology as a graph with its associated Adjacency Matrix (AM). The AM points out the existence of a link between two concepts. In order to extract the relevant concepts, we analyse only the schema of the ontology; the idea is to use a link analysis method as the one

used in the semantic web environment. We customized the HITS algorithm by Kleinberg, implementing a new algorithm able to handle ontologies: HITSxONTO. While HITS works with web pages and hyperlinks, HITSxONTO works on concepts and object properties.

[Step 2] Influence Rule Schema building.
After having pointed out the relevant concepts, we have to identify the "original"[1] concept connections (direct and indirect) by using their object properties. Note that for this purpose, we are not interested in the semantics of the relations but in their existence. Furthermore, we have to associate to all concepts their own data properties, that will be used in the following step.

[Step 3] Characterization of the Influence Rules Schemas.
We would like to give values to the IRs items (in particular to the datatype properties associated with the involved concepts) and a weight for the implications previously discovered. To do so, we analyse the instances that correspond to the metadata in the ontology, and we extract the frequent items with an associated value. The algorithm we use is PATTERNIST, a tool developed at the KDD Lab in Pisa. We then collect, from the frequent itemsets, the values for the Influence Rules schemas and the weights for the implications.

[Step 4] Validation.
The Validation is needed to guarantee that the IRs are consistent and do not conflict with each other. The best way for validating the rules is to ask a domain expert; nevertheless some *ad-hoc* procedures can be implemented with reference to the domain under analysis and foreseeable use.

The first two steps are essentially deductive; they are a sort of "top-down" approach that starts from the theory and tries to find something. The third one is an inductive step, a sort of "bottom-up" approach; we

[1]The term "original" refers to the concepts existing in the actual definition of the ontology.

move from the observations (the instances) to the results (the IRs).
The methodology we propose can be employed in all applications (data mining or non-data mining applications) that make use of additional information in the form of rules or for enriching pre-existing knowledge repository / structures.

Before getting to the significant part, let us introduce the "running example" we already mentioned.

Example 4.1 *Running Example - Part 1: The ontology.*

Let us consider the fragment of ontology shown in picture 22[2]. This ontology describes questions and answer options of a qualitative questionnaire that a financial institution submits to its customers (in this case, managers of companies) when they ask for credit. The data (qualitative information) collected by the questionnaire are used for computing a qualitative score as part of a measure used for deciding the company "worthiness". The list of concepts is the following:

```
A - CapitalizationStrategy
B - Company
C - CustomerBase
D - DiversificationOfProduction
E - FinancialDebt
F - LevelOfCompetition
G - MPSQuestionnaire
H - ManagementTeam
I - MarketState
J - OrganizationalStructure
K - PreviousAchivements
L - QualitativeScore
M - QualityCertificate
N - RelationshipWithTheBankingSystem
O - ResearchAndDevelopment
P - StrategicVisionAndQualityManagement
```

The object properties, with the corresponding domain and range concepts are:

[2]The picture has been realized by using the Jambalaya plug-in from Protégé. Jambalaya is a tool developed by the University of Victoria's CHISEL software engineering group that merges the Shrimp information browser into Protégé as a tab widget.

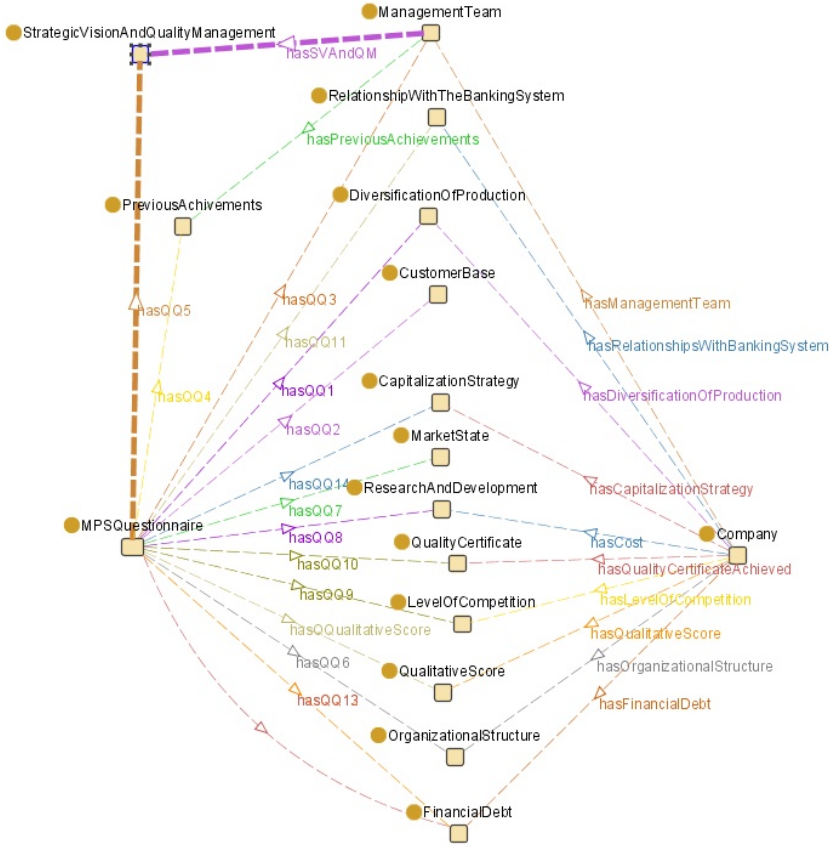


Figure 22: The fragment of the ontology used in the "running example".

Concept = Company

Obj Prop. = hasQualitativeScore
⇒ QualitativeScore
Obj Prop. = hasFinancialDebt
⇒ FinancialDebt
Obj Prop. = hasCapitalizationStrategy
⇒ CapitalizationStrategy
Obj Prop. = hasDiversificationOfProduction
⇒ DiversificationOfProduction
Obj Prop. = hasCost
⇒ ResearchAndDevelopment
Obj Prop. = hasOrganizationalStructure
⇒ OrganizationalStructure
Obj Prop. = hasManagementTeam
⇒ ManagementTeam
Obj Prop. = hasLevelOfCompetition
⇒ LevelOfCompetition
Obj Prop. = hasRelationshipsWithBankingSystem
⇒ RelationshipWithTheBankingSystem
Obj Prop. = hasQualityCertificateAchieved
⇒ QualityCertificate

Concept = MPSQuestionnaire

Obj Prop. = hasQQ8
⇒ ResearchAndDevelopment
Obj Prop. = hasQQ9
⇒ LevelOfCompetition
Obj Prop. = hasQQ10
⇒ QualityCertificate
Obj Prop. = hasQQ1
⇒ DiversificationOfProduction
Obj Prop. = hasQQ3
⇒ ManagementTeam
Obj Prop. = hasQQ2
⇒ CustomerBase
Obj Prop. = hasQQualitativeScore
⇒ QualitativeScore
Obj Prop. = hasQQ14
⇒ CapitalizationStrategy
Obj Prop. = hasQQ5
⇒ StrategicVisionAndQualityManagement

```
   Obj Prop.  = hasQQ4
     ⇒ PreviousAchivements
   Obj Prop.  = hasQQ13
     ⇒ FinancialDebt
   Obj Prop.  = hasQQ7
     ⇒ MarketState
   Obj Prop.  = hasQQ12
     ⇒ FinancialDebt
   Obj Prop.  = hasQQ6
     ⇒ OrganizationalStructure
   Obj Prop.  = hasQQ11
     ⇒ RelationshipWithTheBankingSystem
Concept = ManagementTeam
   Obj Prop.  = hasPreviousAchievements
     ⇒ PreviousAchivements
   Obj Prop.  = hasSVAndQM
     ⇒ StrategicVisionAndQualityManagement
```

◁

4.2 Ontology Schema Mining

This section aims at describing the first step of the extraction process. Both theoretical and technical aspects will be discussed, and a short example will be provided.

4.2.1 From the Ontology to the Weighted Adjacency Matrix

As previously discussed, one of the main changes to HITSxONTO w.r.t. HITS is the adjacency matrix to use in input. The adjacency matrix is the algebraic and computable representation of the ontology that permits one to handle this structure in a simpler way. Furthermore, it permits one to prove important and desirable properties of the whole algorithm (termination/convergence and stability).

An adjacency matrix A is, by definition, an nxn matrix associated with a graph G having n nodes, so that each entry a_{ij} of A is defined as follows:

$$a_{ij} = \begin{cases} 1 & \text{if an edge from } i \text{ to } j \text{ exists} \\ 0 & \text{otherwise} \end{cases}$$

This matrix is symmetric if G is not directed.

The ontology cannot be represented as a simple directed graph because "multiple relationships" can occur between two nodes, i.e. two nodes can be connected by more than one object property with different label (semantics). The solution is to use a weighted adjacency matrix where each entry represents the number of edges connecting two nodes. If we call W the nxn matrix associated with the ontology O, the correspondent definition is:

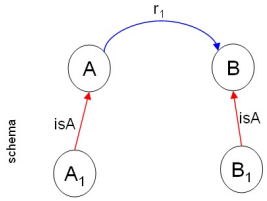
$$w_{ij} = \begin{cases} k & \text{if } k \text{ edges from } i \text{ to } j \text{ exist} \\ 0 & \text{otherwise} \end{cases}$$

It is interesting to consider all the possible cases rising from the ontology relationships (object properties and *is-a* relations) and the way in which they are solved in the matrix. Critical aspects are related to the existence of more than one object property between the same two concepts, and to the relationships in presence of hierarchies of concepts. In this latter case, the inheritance property has to be handled. The general rule that drives the inheritance is the fact that only the super-concept determines the hierarchy, i.e. specifies whether each of its sub-concept is the domain or the range. This is because in an ontology the sub-concepts inherit the object properties of the parents; the opposite case does not hold.
It is important to observe that in OWL it is possible to specify restrictions on properties in a way that the general notion of inheritance among classes (subconcepts and superconcepts) does not always hold. By means of restriction, one can define exceptions, and it is further possible to constrain the range of a property in specific contexts in a variety of ways (Com04a). For the moment, we can ignore this fact because in the first step of our analytical process, we consider only the structure of the ontology (e.g. its graph representation) and the information about the restrictions is lost during the codification from ontology to a graph. Neverthe-

less, this non-trivial aspect is very interesting and could be part of future extensions in which we refine the structural analysis by using additional information coded in the ontology.

Each of the following cases presents a fragment of both the ontology and the associated weighted adjacency matrix.

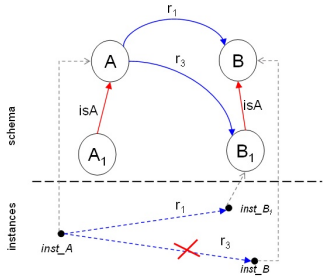
Case 1: Simple inheritance [Figure 23]. The sub-concepts inherit the object properties of the super-concept to which they are connected by the *isA* property. A_1 inherits from A the object property r_1, becoming in its turn, the domain of r_1. B, being the range of r_1, hands on the "range state" to B_1. Even if the edges from A_1 to both B and B_1 do not physically exist, the corresponding relationships hold and can be coded in W as shown in figure 23.



$$W = \begin{array}{c} \\ \mathrm{A} \\ \mathrm{A_1} \\ \mathrm{B} \\ \mathrm{B_1} \end{array} \begin{array}{c} \begin{array}{cccc} \mathrm{A} & \mathrm{A_1} & \mathrm{B} & \mathrm{B_1} \end{array} \\ \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

Figure 23: Case 1 - Simple Inheritance.

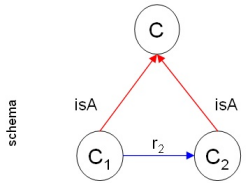
Case 2: Complex inheritance [Figure 24]. With respect to Case 1, here we also have one object property (r_3) connecting a super-concept (A) with a not relative (i.e. a concept that is not sibling or son) sub-concept (B_1). It is the first case of indirect multiple relationships. A_1 inherits from A the object properties r_1 and r_3, establishing relations with B and B_1. A has a single relation with B but a double relation with B_1 (one indirectly induced by r_1, and the other directly induced by r_3 as the entry w_{AB_1} shows). Notice that w_{AB} = 1 (and not = 2) because r_3 is defined on B_1 and the inheritance is not applicable toward B, i.e. B does not inherit the "range state" of B_1 induced by r_3. In fact, given instances $inst\_A \in A$ and $inst\_B \in B$, they cannot be connected by means of r_3.



$$W = \begin{array}{c} \\ A \\ A_1 \\ B \\ B_1 \end{array} \begin{array}{c} \begin{array}{cccc} A & A_1 & B & B_1 \end{array} \\ \begin{bmatrix} 0 & 0 & 1_{r1} & 1_{r1}+1_{r3} \\ 0 & 0 & 1_{r1} & 1_{r1}+1_{r3} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

Figure 24: Case 2 - Complex Inheritance.

Case 3: No inheritance (1/3) [Figure 25]. The inheritance is not applicable from the sub-concepts to the parent, so only the relationship between C_1 and C_2 exists and it is pointed out in the entry $w_{C_1C_2} = 1$.

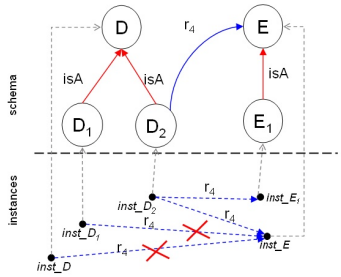


$$W = \begin{array}{c} \\ C \\ C_1 \\ C_2 \end{array} \begin{array}{c} \begin{array}{ccc} C & C_1 & C_2 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \end{array}$$

Figure 25: Case 3 - No Inheritance (1/3).

Case 4: No inheritance (2/3) [Figure 26]. This is the case in which neither the parent concept, nor the sibling concept inherit the "domain state" induced by an object property defined on a concept which is son and brother, respectively. Among the set of relative concepts $\{D$, D_1, $D_2\}$, only D_2 has a relation directly with E and indirectly with E_1.

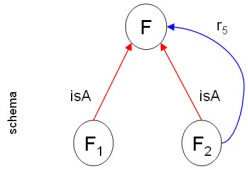
Case 5: No inheritance (3/3) [Figure 27]. As already stated, brother concepts do not inherit the "mutual" object properties, and parents concepts do not inherit from their (son) sub-concepts. In this case, only F_2 is domain of the object property r_5, and it can establish a direct



$$W = \begin{array}{c} \\ \mathrm{D} \\ \mathrm{D_1} \\ \mathrm{D_2} \\ \mathrm{E} \\ \mathrm{E_1} \end{array} \begin{array}{c} \begin{array}{ccccc} \mathrm{D} & \mathrm{D_1} & \mathrm{D_2} & \mathrm{E} & \mathrm{E_1} \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

Figure 26: Case 4 - No Inheritance (2/3).

relation with F and an indirect one with F_1 (because this latter inherits the "range state" from F).

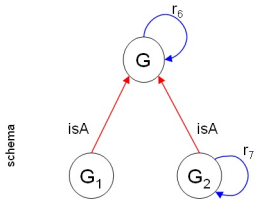


$$W = \begin{array}{c} \\ \mathrm{F} \\ \mathrm{F_1} \\ \mathrm{F_2} \end{array} \begin{array}{c} \begin{array}{ccc} \mathrm{F} & \mathrm{F_1} & \mathrm{F_2} \end{array} \\ \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} \end{array}$$

Figure 27: Case 5 - No Inheritance (3/3).

Case 6: Circular properties [Figure 28]. A "circular property" exists when its domain and its range coincide: this is the case of G w.r.t. the object property r_6 and G_2 w.r.t. r_7. G is thus in relation with itself, and both its sons G_1 and G_2 thanks to r_6. On the other hand, both G_1 and G_2 inherit from G the object property r_6 so that they establish a connection with all the other concepts. Moreover, G_2 has one more connection with itself, thanks to r_7 (note in fact that $w_{G_2G_2} = 2$).

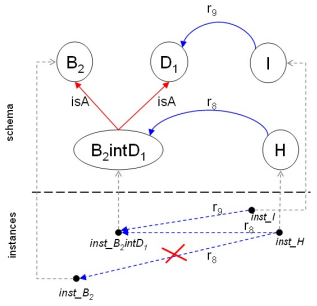
Case 7: Class Intersection [Figure 29]. We introduce here the intersection class. This class contains instances that are common to its parent classes. B_2intD_2 is the intersection of B_2 and D_1 because it is simultaneously sub-concept of both B_2 and D_1.
B_2intD_2 is range of the object property r_8 but does not hand on this



$$W = \begin{array}{c} \\ \mathrm{G} \\ \mathrm{G_1} \\ \mathrm{G_2} \end{array} \begin{array}{c} \begin{array}{ccc} \mathrm{G} & \mathrm{G_1} & \mathrm{G_2} \end{array} \\ \begin{bmatrix} 1_{r_6} & 1_{r_6} & 1_{r_6} \\ 1_{r_6} & 1_{r_6} & 1_{r_6} \\ 1_{r_6} & 1_{r_6} & 1_{r_6} + 1_{r_7} \end{bmatrix} \end{array}$$

Figure 28: Case 6 - Circular Properties.

"range state" to B_2 or D_1. Moreover, $B_2 int D_2$ inherits the "range state", induced by r_9, from D_1 so that the concept I can establish a connection with D_1 (directly), and with $B_2 int D_1$ (indirectly). The concept H is connected only with $B_2 int D_1$.



$$W = \begin{array}{c} \\ \mathrm{B_2} \\ \mathrm{D_1} \\ \mathrm{B_2intD_1} \\ \mathrm{H} \\ \mathrm{I} \end{array} \begin{array}{c} \begin{array}{ccccc} \mathrm{B_2} & \mathrm{D_1} & \mathrm{B_2intD_1} & \mathrm{H} & \mathrm{I} \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \end{bmatrix} \end{array}$$

$$\mathrm{B_2intD_1} \equiv (\mathrm{B_2} \cap \mathrm{D_1})$$

Figure 29: Case 7 - Class Intersection.

Case 8: Multiple Properties [Figure 30]. In an ontology two concepts can be related by more than one property; for example, they can be respectively, domain and range for two object properties with different semantics. For our analysis it is important to point out this fact in the matrix by writing the number of relationships. A and B are related by the object properties r_1 and r_3, so $w_{AB} = 2$.

As you can see, two important cases are missing in this description: the *Union Classes* and the *Property hierarchies*. As stated in the W3C

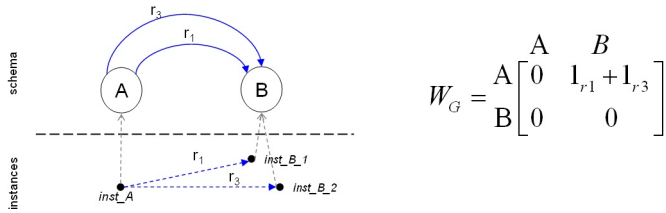


Figure 30: Case 8 - Multiple properties.

specifications ((Com04a)), the union class contains elements that are included in some of its subclasses, while property hierarchies may be created by making one or more statements indicating that a property is a sub-property of one or more other properties. These cases are ignored in this release, but we have already planned their implementation for the future prototype.

Example 4.2 *Running Example - Part 2: The adjacency matrix.*
With reference to the ontology in figure 22 and the list of associated concepts, the corresponding adjacency matrix that respects the constraints just reported, is in figure 31. As one can expect, the matrix contains for the most part the value 0; this is because in the ontology there are not many connections.

AdjMatrix=

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
A	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
B	1	0	0	1	1	1	0	1	0	1	0	1	1	1	1	0
C	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
D	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
E	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
F	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
G	1	0	1	1	2	1	0	1	1	1	1	1	1	1	1	1
H	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1
I	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
J	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
K	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
L	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
M	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
N	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
O	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 31: The adjacency matrix associated to the ontology in figure 22.

4.2.2 HITSxONTO Algorithm

HITSxONTO is the customized version of the HITS algorithm for handling ontologies that we presented in section 3.1.3. Like HITS, HITSxONTO is based on the concept of authority and hubness, and its purpose is to measure the importance of the ontology concepts, basing only on the ontology topology (the T-Box). In other words, it tries to deduce which concepts can be considered particularly "important" (authorities) and which ones give a particular importance to other concepts (hubs).
The general idea of the original version, as well as the main strategy, has been preserved thanks also to the natural comparison that can be made between web and ontology "elements" (figure 32). In the ontology the elements we have to consider are the concepts and the object properties. In addition, the *is-a* relation is taken into consideration, but only for constructing the matrix as shown in the previous section. The datatype properties, instead, are not relevant to this process but will be indispensable in the next steps. Therefore, a web page can be seen as an ontology concept, and a hyperlink resembles an object property.

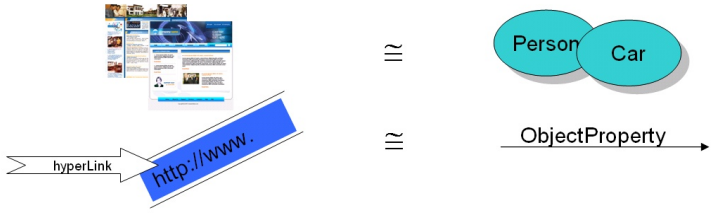


Figure 32: Web and Ontology comparison.

The main algorithm variant concerns the pre-processing phase, such as the preparation of the input and the general adaptation to the ontology. As already stated, the object properties no longer become important but maintain their importance as connective elements. We are, in fact, interested in the existence of a property rather then its meaning. Furthermore,

as discussed in section 4.2.1, we must also consider the "hidden" relationships induced by the *is-a* properties.

HITSxONTO is iterative as well, and follows the same core steps as HITS. Algorithm 1 shows the pseudo code.

```
 INPUT: An ontology O
 OUTPUT: A set of ranked concepts
1: Read O;
2: Construct the Graph G = (V, E) from O;
3: Draw M;
4: Compute M^T;
5: Reset hScore and aScore;
6: Initialize h_prev and a_prev;
7: repeat
8:    hScore ← M x a_prev;
9:    aScore ← M^T x hScore;
10:   Normalize hScore and aScore;
11:   h_prev ← hScore;
12:   a_prev ← aScore;
13: until {(hScore ≠ h_prev) || (aScore ≠ a_prev)};
14: Return hScore and aScore;
```

Algorithm 1: Pseudo code of the *HITSxONTO* algorithm.

Algorithm 1 Description. After having loaded the ontology O, the correspondent graph representation (G) is derived (steps 1 and 2). G is a graph where each node corresponds to an ontology concept, and each edge to an object property or to an *is-a* relation.
By using the translation rules discussed in section 4.2.1, the input matrix M is filled in (step 3), and from M, the transposed matrix M^T is computed (step 4). M (and obviously M^T) is an nxn matrix, where n is the number of concepts of the ontology/nodes of the graph.
$hScore$ and $aScore$ are two n-dimensional vectors where the ranking values (hub and authority respectively) associated to the concepts, are stored at each iterative run (the cycle is performed by steps 7, ..., 13). In order to support the iterative computation, two temporary vectors are needed:

h_{prev} and a_{prev} store the ranking values, computed at the iteration $i-1$, that are needed in the next iteration i.
In steps 5 and 6 we initialize the vectors. It is usual (see (FLM+06; Kle98; AP05)) to initialize these authority and hub vectors with a uniform distribution, such as $1/n$, 1 or $1/\sqrt{n}$, for each $i \in [1, ..., n]$ entry. The following are examples of hub seed vectors.

i) $hScore_0 = [1/n, \ldots, 1/n]$
ii) $hScore_0 = [1/\sqrt{n}, \ldots, 1/\sqrt{n}]$

We already discussed in section 3.1.3 the problem of stability of the HITS algorithm related to the dependency of the starting seed vectors: whenever the dominant eigenvalue is not unique, the HITS output is sensitive to the initial seed vector. It is important to point out that HITSxONTO does not suffer from this weakness because, in order to solve this issue, we applied the strategy proposed by Farahat et all. in (FLM+06) (and discussed in section 3.1.3). We remind the reader that, reasoning in algebraic terms, the computation of the authority and hub vectors (of both HITS and HITSxONTO) can be reduced to the computation of the dominant eigenvectors of the matrix A^TA and AA^T respectively (where A is the original adjacency matrix). From this statement, there obviously follows the strong relation with the initial adjacency matrix (A in this case). Farahat et all. proved this dependency and found a way for solving the HITS weakness by acting on (modifying) the adjacency matrix and building from that an exponentiated adjacency matrix:

$$EXPO(A) = (e^A - I) = \sum_{k=0}^{\infty} \frac{1}{k!} A^k$$

They also assert that matrices such as $A + A^2/2!$ or $I + A$, rather than $e^A - I$, would exclude these possibilities so long as the ontology graph is weakly connected [3]. For our purposes, we adopt the same strategy. The adjacency matrix derived directly from the ontology graph is a weighted matrix in

[3] By "weakly connected" we mean a graph ontology in which many of nodes/concepts that are connected to few other nodes, are isolated, or an ontology graph that is composed of unconnected graphs.

its own right, because we apply the translation rules (described in section 4.2) and the matrix entries can have positive integer values different from 0 and 1. Then we compute the exponentiated adjacency matrix.
Here, for obvious computational complexity issues related to the matrix size (in an ontology it is possible to have to handle hundreds of concepts) and also related to the computation of the matrix $e^A - I$, we approximate the exponentiated matrix to:

$$EXPO(A) = \sum_{k=0}^{m} \frac{1}{k!} A^k$$

where m is a finite positive integer.

This choice does not limit us and does not disprove the theory just discussed. Let us observe that if a matrix is *nilpotent*[4], then $A^n = 0$ and the exponentiated matrix can be computed directly from the series expansion, and the series terminates after a finite number of terms:

$$e^A = I + \frac{A^2}{2} + \frac{A^3}{3!} + \cdots + \frac{A^{(n-1)}}{(n-1)!}.$$

In the context of the ontologies, the associated adjacency matrices have mostly zero entries, because in general, big ontologies are badly connected so that they benefit from the nilpotent propriety.
The choice of m in the computation of $EXPO(A)$ depends on both the structure and the size of the matrix and can be empirically fixed. In our tests, in most of the cases, we fixed $m = 3$.

After this important digression, let us continue the description of the algorithm.
The iterative process for computing the scores starts at step 7 and continues until step 13. As for HITS, a kind of mutual reinforcing approach is exploited. This process can be described algebraically[5] by using the notation adopted in section 3.1.3 as the following products:

$$\begin{aligned} h^{(k)} &= Ma^{(k-1)} \\ a^{(k)} &= M^T h^{(k)} \end{aligned} \tag{4.1}$$

[4] A matrix A is nilpotent if $A^n = 0$ for some positive integer n (here 0 denotes the matrix where every entry is 0).

[5] This algebraic formulation will be useful for proving important properties of the algorithm: the termination and the correctness.

$h^{(k)}$ and $a^{(k)}$, that correspond to $hScore$ and $aScore$ at the k-th iteration, are computed on the basis of the results obtained at the $k-1$-th iteration. The iterative process ends when the fixed point is reached (step 13); in real terms it ends when the vectors do not change from an iteration to the next one.

Each iteration requires the normalization of the vectors to control the value ranges (step 10). The normalization measure used in this case is a very common one:

$$\sum_{i=0}^{n}(V[i])^2 = 1, \tag{4.2}$$

where V is a vector with $n+1$ elements.
In steps 11 and 12 temporary vectors are re-assigned for starting a new cycle.
At the end the vectors containing the final rankings for hub and authority, are returned (step14).
From equation 4.1, replacing the i-th term with its definition, after k iterations (a part from normalizations), we obtain the following equivalences:

$$\begin{aligned} h^{(k)} &= (MM^T)h^{(k-1)} = (MM^T)Ma^{(k-2)} = \cdots = (MM^T)^{(k-1)}Ma^{(0)} \\ a^{(k)} &= (M^TM)a^{(k-1)} = (M^TM)M^Th^{(k-1)} = \cdots = (M^TM)^{(k)}a^{(0)} \end{aligned} \tag{4.3}$$

Both the matrices (MM^T) and (M^TM) play a fundamental role in the study of the HITSxONTO convergence. The convergence is based on the properties of these matrices and consequently of M. As for HITS, the termination is guaranteed and authority and hub vectors tend to the dominant eigenvector of (M^TM) and (MM^T), respectively. Furthermore, thanks to the definition of the input matrix (as described in section 4.2.1) and to the results obtained in (FLM+06), uniqueness is also guaranteed.

It is useful and interesting to show the proof of the HITSxONTO convergence that follows directly from the results obtained by Agosti et all. in (AP05) where they conducted a study of a generalized version of HITS. Before the proof, we cite some theorems originated both from the work presented in (AP05), and about known properties of the matrices, essential for justifying some crucial steps.

Theorem 4.2.1
The eigenvalues of a real symmetric matrix A are real numbers.

Theorem 4.2.2
Let A be a real symmetric $n \times n$ matrix. Then $\mathbb{R}^n$ has an orthonormal basis consisting of eigenvectors of A.

Corollary 4.2.1
If A is an $n \times n$ real symmetric matrix, then there are real matrices L and D such that $L^T L = LL^T = I$ and $LAL^T = D$, where I is the identity matrix and D is the diagonal matrix of eigenvalues of A.

Theorem 4.2.3
A real symmetric matrix is positive semi-definite if and only if its eigenvalues are nonnegative. A real symmetric matrix is positive definite if and only if its eigenvalues are positive.

Theorem 4.2.4
A real symmetric matrix A is positive semi-definite if and only if there is a real matrix B such that $A = B^T B$.

Theorem 4.2.5 ***Frobenius theorem***
Let A be a real nonnegative and irreducible square matrix and $\rho(A)$ its spectral radius [6]*. Then:*

1. $\rho(A) > 0$;
2. $\rho(A)$ *is an eigenvalue of A belonging to a real positive eigenvector;*
3. $\rho(A)$ *has* 1 *as algebraic multiplicity.*

Theorem 4.2.6
Matrix $M^T M$ has a strictly dominant eigenvalue if and only if one of the matrices $B_1, B_2, \ldots, B_m$ has a strictly dominant eigenvalue that is greater than the strictly dominant eigenvalue of every other matrix of this multi-set of matrices. This theorem follows from some results shown in (AP05).

Corollary 4.2.2
If the matrix $M^T M$ represents a connected graph G_w, then it has a strictly dominant eigenvalue. This corollary follows from some results shown in (AP05).

[6]Spectral Radius Definition: Let $\lambda_1, \ldots, \lambda_s$ be the (real or complex) eigenvalues of a matrix A_{nxn}. Then its spectral radius $\rho(A)$ is defined as: $\rho(A) = max_i(|\lambda_i|)$.

Proof or the HITSxONTO convergence.

Let us consider equations 4.3, and let us focus on the authority vector, re-writing it in the equivalent following way:

$$a^{(k)} = (M^T M)^{(k-1)} M^T \mathbf{u} \tag{4.4}$$

where u is the initial seed vector and $k = 1, 2, \ldots$.
Moreover, let us define $M^T M \equiv B$, where $B_{ij} \equiv bij = \sum_{k=1}^{n} m_{ki} m_{kj}$. Each entry $bij \neq 0$ iff there is at least one node k of the associated graph with outgoing arcs towards nodes i and j at the same time.
From theorems 4.2.1, 4.2.4 and 4.2.3 we get that $M^T M$ is an $n \times n$ real and nonnegative symmetric matrix with a particular structure, so that all its n eigenvalues are real and nonnegative. Grouping the connected components of the original graph and ordering them, B can be seen as a matrix with diagonal blocks as follow:

$$B = \begin{bmatrix} B_1 & 0 & \ldots & 0 \\ 0 & B_2 & \ldots & 0 \\ \vdots & & \ddots & \\ 0 & 0 & \ldots & B_m \end{bmatrix} \tag{4.5}$$

The eigenvalues of B are calculated by adding the eigenvalues of B_1 to the eigenvalues of B_2, and so forth, up to the eigenvalues of B_m. Each of these matrices is a real, irreducible and nonnegative symmetric matrix with real nonnegative eigenvalues since the eigenvalues of $M^T M$ are real and nonnegative. From corollary 4.2.2 and theorem 4.2.5, we find that each of the matrices B_i, which is different from 0, has a strictly dominant eigenvalue, i.e. an eigenvalue which is strictly greater than all the other eigenvalues of the matrix. Moreover, all the entries of its associated eigenvector are greater than 0. We can see that the eigenvectors of matrix B can be obtained from the eigenvectors of the matrices $B_1, B_2, \ldots, B_m$ by just considering 0 the entries corresponding to the other matrices.
For example, starting from the eigenvector u_{kj}, i.e. the eigenvector k of matrix B_j, for matrix B we get the eigenvector:

$$[\underbrace{0 \ldots 0}_{1} | \underbrace{0 \ldots 0}_{2} | \ldots | \underbrace{\mathbf{u}_{kj}}_{j} | \ldots | \underbrace{0 \ldots 0}_{m}] \tag{4.6}$$

Starting from each eigenvector of matrices B_i, we get n eigenvectors of B with this structure which form a basis of $\mathbb{R}^n$. Theorem 4.2.2 assures us that each matrix B_i has an orthonormal basis consisting of its eigenvectors; this is the basis we will consider in the following part of this proof, so that even B will have an orthonormal basis. Finally, since each matrix B_i is real and symmetric, from corollary 4.2.1 we get that B_i is orthogonally diagonalizable, meaning that an orthogonal matrix Q_i such that:

$$Q_i^{-1} B_i Q_i = \begin{bmatrix} \lambda_{1i} & 0 & \dots & 0 \\ 0 & \lambda_{2i} & \dots & 0 \\ \vdots & & \ddots & \\ 0 & 0 & \dots & \lambda_{k_i i} \end{bmatrix} \tag{4.7}$$

does exist, where $\lambda_{1i}, \lambda_{2i}, \dots, \lambda_{k_i i}$ are the k_i real and nonnegative eigenvalues of Bi.

Starting from this result, let us suppose that $M^T M$ has $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_n \geq 0$ as eigenvalues belonging to the eigenvectors $v_1, v_2, \dots, v_n$. Each of these eigenvectors is related to a particular diagonal block of matrix B, so that it has entries 0 corresponding to the other matrices, as shown in formula 4.6. If more than one matrix B_i has a strictly dominant eigenvalue equal to λ_1, i.e. the dominant eigenvalue of B, then we have an example of the more general case we are interested in.
Let us suppose $\lambda_1 = \lambda_2 = \dots = \lambda_r > \lambda_{r+1} \geq \lambda_{r+2} \geq \dots \geq \lambda_n 0$. The eigenvectors $v_1, v_2, \dots, v_r$ are related to the r different matrices
$B_{h_1}, B_{h_2}, \dots, B_{h_r}$, so that v_j, $1 \leq j \leq r$, has positive entries corresponding to the diagonal block B_{h_j}, and 0 otherwise.
We can express vector $W^T u$ as:

$$W^T \mathbf{u} = \alpha_1 v_1 + \dots + \alpha_r v_r + \alpha_{r+1} v_{r+1} + \dots + \alpha_n v_n.$$

Since $\mathcal{B} \triangleq v_1, \dots, v_n$ is an orthonormal basis,

$$\alpha_j = \left\langle W^T \mathbf{u}, v_j \right\rangle, j = 1, \dots, n$$

where $\left\langle W^T \mathbf{u}, v_j \right\rangle$ is the scalar product between $W^T \mathbf{u}$ and v_j.
From formula 4.4, we can obtain:

$$
\begin{aligned}
a^{(k)} &= (M^T M)^{(k-1)} M^T \mathbf{u} = \\
&= (M^T M)^{(k-1)} (\alpha_1 v_1 + \alpha_2 v_2 + \cdots + \alpha_n v_n) = \\
&\text{since } Ax = \lambda x \\
&= \alpha_1 \lambda_1^{k-1} v_1 + \alpha_2 \lambda_2^{k-1} v_2 + \cdots + \alpha_n \lambda_n^{k-1} v_n \\
&k = 1, 2 \ldots
\end{aligned}
$$

Let be $\lambda \triangleq \lambda_1 = \lambda_2 = \cdots = \lambda_r$, then

$$
\begin{aligned}
a^{(k)} &= \lambda^{k-1} \left(\alpha_1 v_1 + \alpha_2 v_2 + \cdots + \alpha_r v_r + \sum_{i=r+1}^{n} \frac{\alpha_i \lambda_i^{k-1} v_i}{\lambda_i^{k-1}} \right) = \\
&= \lambda^{k-1} (\alpha_1 v_1 + \alpha_2 v_2 + \cdots + \alpha_r v_r + v(k) \\
&k = 1, 2 \ldots
\end{aligned}
$$

where

$$
v(k) \triangleq \sum_{i=r+1}^{n} \frac{\alpha_i \lambda_i^{k-1} v_i}{\lambda_i^{k-1}},\ k = 1, 2 \ldots
$$

is a sequence of vectors so that $\lim_{k \to +\infty} v(k) = 0$. All the entries, in fact, vanish as $k \to +\infty$, since $\lambda_i / \lambda < 1$ when $r + 1 \leq i \leq n$.
The algorithm makes us compute

$$
\lim_{k \to +\infty} \frac{a^(k)}{\|a^{(k)}\|}
$$

where

$$
\begin{aligned}
\left\| a^{(k)} \right\| &= \left\| \lambda^{k-1} (\alpha_1 v_1 + \alpha_2 v_2 + \cdots + \alpha_r v_r + v(k)) \right\| = \\
&= \lambda^{k-1} \left\| (\alpha_1 v_1 + \alpha_2 v_2 + \cdots + \alpha_r v_r + v(k)) \right\|.
\end{aligned}
$$

Since every vector norm $\|x\|$, with $x = [x_1, x_2, \ldots, x_n]^T$, is a continuous function of the variables $x_1, x_2, \ldots, x_n$. then

$$
\begin{aligned}
lim_{k \to +\infty} &\left\| \alpha_1 v_1 + \alpha_2 v_2 + \cdots + \alpha_r v_r + v(k) \right\| = \\
&= \left\| \alpha_1 v_1 + \alpha_2 v_2 + \cdots + \alpha_r v_r \right\|
\end{aligned}
$$

so that

$$
\lim_{k \to +\infty} \frac{a^{(k)}}{\left\| a^{(k)} \right\|} = \frac{\alpha_1 v_1 + \alpha_2 v_2 + \cdots + \alpha_r v_r}{\left\| \alpha_1 v_1 + \alpha_2 v_2 + \cdots + \alpha_r v_r \right\|}
$$

From this last result we find that, under whatever norm, the HITSx-ONTO algorithm, as both HITS and the revised HITS presented by Agosti et all., converges to a unit authority vector which is a linear combination of all the dominant eigenvectors of $M^T W$.

Example 4.3 *Running Example - Part 3: Hub and Authority rankings.*
Let us apply the HITSxONTO algorithm to the adjacency matrix in figure 31 by using the following setting for the required parameters:

Hub Threshold: $Ht = 0.0$;
Authority Threshold: $At = 0.2$.

The Hub and Authority vectors (*hScore* and *aScore*) are:

$$hScore = [0.0, 0.5682, 0.0, 0.0, 0.0, 0.0, 0.8151, 0.1131, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]$$
$$aScore = [0.2585, 0.0, 0.1523, 0.2585, 0.4109, 0.2585, 0.0, 0.2585, 0.1523, 0.2585, 0.3027, 0.2585, 0.2585, 0.2585, 0.2585, 0.3027]$$

As one can image, since the matrix contains many 0s, the concepts with higher hubness are `Company` and `MPSQuestionnaire`, i.e. the concepts that actually have, in the ontology, more out-going edges w.r.t. the others. The same holds for the authority: all the concepts, except `Company` and `MPSQuestionnaire`, have the same number of incoming edges, so the ranking is, more or less, uniform.
Based on the *hScore* and *aScore*, the two ordered lists of the most important concepts are generated, ready to be "filtered" in agreement with the acceptance thresholds (see part 4 of the running example).

◁

4.3 Giving a Structure to the "Influence Rules"

The use of the HITSxONTO algorithm for finding the most relevant concepts in the ontology permits us also to exploit another property for constructing the basic schema of the IRs. In fact, the separation of authoritative and hubness concepts allows us to identify and assign roles to the concepts in the IRs. In a natural way, we associate the implicant position to the authoritative concepts and the implicated position to the hubness concept in a schema rule based on the fact that, by definition, the former attract incoming links and the latter are a sort of source for out-going links.

According to the strategy sketched at the beginning of this chapter, the construction of the schemas involves the use of the "original" object properties, i.e. the object properties existing in the ontology under analysis. The object properties are, in fact, the elements that connect the concepts and permit one to create links between them. In this case, they are used for creating the pairs < $Implicant, Implicated$ >. $Implicant$ is the ontology concept the implication starts from, while $Implicated$ is the concept influenced by the implicant concept.

The pseudo code below (Algorithm 2) shows the simple strategy.

```
 INPUT: The list of concepts:  CL
        The Adjacency Matrix:  M
        The length of the paths:  L
        The hub threshold:  Ht
        The Authority threshold:  At
 OUTPUT: A set of Influence Rules Schemas:  IRSchemas
1: M̄ = computeIndirectConnections(L, M);
2: foreach (implicant ∈ CL) {
3:    if (implicant.hubValue >= Ht)
4:    then {
5:       foreach ((implicated ∈ CL) and
                  (connected(M̄, implicant, implicated)) {
6:          if (implicated.authorityValue >= At)
7:          then updateIRSchema(implicant, implicated);
8:       }
9:    }
10: }
11: Return IRSchemas;
```

Algorithm 2: Pseudo code of the IRs schema generation procedure.

Algorithm 2 Description. The procedure takes as input the list of concepts and the adjacency matrix. From them, the information about the concepts and their relationships (in term of connections) are collected. Ht and At represent the thresholds of acceptance and define the minimum score that a concept must obtain in order to be considered "important", thus becomes a candidate for a IR Schema (the corresponding acceptance

tests are made at steps 3 and 6).
Choosing the thresholds is not simple, because they strongly depend on the ontology connectivity and on the number of concepts, and they cannot be fixed a-priori. Since the process must be automatic (and it is not guided by an expert in this phase), an evaluation function (heuristic) has been created. This function depends on the number of the concepts and on the distribution of scores, and it is based on the mean of the values. In general, the threshold is the arithmetic mean, but when more then 70% of the scores are "0", the threshold is set to "0" and all the concepts are considered as candidate.

Each rule schema is created by using the candidate implicant concepts (step 2) and connecting them with the candidate implicated concepts reachable directly or indirectly by object properties (step 5). In general, two concepts are connected if an object property or a path (composed by more than 1 object property) exist among them. Starting from the original adjacency matrix M it is possible to quickly determine if an indirect connection exists by multiplying M by itself many times, depending on the intermediate steps we want to consider. In fact, from graph theory, we know that the matrix of paths of length n are generated by multiplying the matrix of paths of length $n-1$ by that of length 1. The method `computeIndirectConnections(`L`,` M`)` at step 1, computes this new matrix $\bar{M}$ that contains all the paths from 1 to L. Formally:

$$\bar{M} = \sum_{i=1}^{L} M^i = M + M^2 + M^3 + ... + M^L \tag{4.8}$$

The set of associations $< implicant, implicated >$ created at step 7 is the result (step 11).

Detailed descriptions of `computeIndirectConnections`, `connected` and `updateIRSchema` subroutines are available in appendix A.3.1. The pseudo codes of `computeIndirectConnections` and `updateIRSchema` are also reported in Algorithm 4 and 5, respectively.

Example 4.4 *Running Example - Part 4: The Influence Rules Schemas.*

Considering the acceptance thresholds that we set for the hub and authority, the concepts candidates that can be part of rules are the following:

Implicant set = {ManagementTeam,
Company,
MPSQuestionnaire}.
Implicated set = {CapitalizationStrategy,
DiversificationOfProduction,
LevelOfCompetition,
ManagementTeam,
OrganizationalStructure,
QualitativeScore,
QualityCertificate,
RelationshipWithTheBankingSystem,
ResearchAndDevelopment,
PreviousAchivements,
StrategicVisionAndQualityManagement,
FinancialDebt}.

Each item in the Implicant Set obtained a hub value greater than 0.0, while each item in the Implicated Set obtained an authority value greater than 0.2, according to the thresholds we set (see example 4.3).
Each rule schema is then created by using the candidate implicant concepts and connecting them with the candidate implicated concepts reachable directly or indirectly by object properties. The result set of IRs Schemas is the following:

1. Company → CapitalizationStrategy
2. Company → DiversificationOfProduction
3. Company → FinancialDebt
4. Company → LevelOfCompetition
5. Company → ManagementTeam
6. Company → OrganizationalStructure
7. Company → QualitativeScore
8. Company → QualityCertificate
9. Company → RelationshipWithTheBankingSystem
10. Company → ResearchAndDevelopment
11. MPSQuestionnaire → CapitalizationStrategy
12. MPSQuestionnaire → DiversificationOfProduction
13. MPSQuestionnaire → FinancialDebt
14. MPSQuestionnaire → LevelOfCompetition
15. MPSQuestionnaire → ManagementTeam

```
16.  MPSQuestionnaire → OrganizationalStructure
17.  MPSQuestionnaire → PreviousAchivements
18.  MPSQuestionnaire → QualitativeScore
19.  MPSQuestionnaire → QualityCertificate
20.  MPSQuestionnaire →
             RelationshipWithTheBankingSystem
21.  MPSQuestionnaire → ResearchAndDevelopment
22.  MPSQuestionnaire →
             StrategicVisionAndQualityManagement
23.  ManagementTeam → PreviousAchivements
24.  ManagementTeam →
             StrategicVisionAndQualityManagement
```

As we stated earlier, from this set all possible implications (candidates to became IRs) can be generated.

◁

4.4 Influence Rules Characterization: the Use of the Instances

Until now, the only information we used for constructing the IRs have been extracted at "schema level", i.e. exploiting only what the ontology structure provides (concepts and relationships). What we have obtained is a set of rules schemas that identify only the important items and how they are related, but do not supply information about the values and the relations strength. At this stage of the process, the ontology instances play a fundamental role: they, in fact contain all the information necessary for supplementing (characterizing) the rules schemas.

The idea is to analyse the ontology instances by using a patterns discovery strategy in order to extract the frequent itemsets. In particular, we apply the PATTERNIST algorithm that has been described in section 3.2. In this context, the frequent itemsets are sets of instances of ontology elements that appear (together) more frequently and whose support is greater than some user-specified minimum support. A crucial point here is to be able to organize the set of ontology instances as transactions as

requested by PATTERNIST (and by all the pattern discovery algorithms).

In general, this type of organization is suggested by the kind of rules we want to extract and, more general, by the kind of analysis we want to perform. This action is possible by querying, in the right way, the ontology and handling the results as they were tuples of a relational database. The pseudo code (Algorithm 3) describes the steps for the identification of the values for characterizing the IRs schemas and producing the actual IRs set.

Algorithm 3 Description. This procedure aims at producing the candi-

```
 INPUT: The ontology instances set:  I
         The minimum support:  minS
         The IRs Schemas set:  IRSchemas
 OUTPUT: The set of Influence Rules:  IRsSet
1: FIsTemp = PATTERNIST(I, minS);
2: FIs = filterFrequentItems(FIsTemp);
3: IRsSet = characterizeIRSchemas(IRSchemas, FIs);
4: Return IRsSet;
```

Algorithm 3: Pseudo code of the IRs characterization.

date IRs by merging the IRs Schemas with the information extracted from the instances.

The instances I, together with the minimum support value $minS$ that acts as threshold of acceptance, are passed to the PATTERNIST algorithm for computing the frequent itemsets $FIsTemp$ (step 1). The ontology instances set I must be organized (in form of transactions) and formatted in agreement with the format accepted by the algorithm: the FIMI file format[7].

$FIsTemp$ is then passed to another utility procedure that filters the results and returns back all the itemset of interest (step 2). For the moment, the parameter used for the selection is the number of items that each itemset has to contain. In this first prototype the IRs we want to extract are "simple", i.e. composed of only one implicant and only one

[7]The FIMI file format is adopted for coding the input file of many algorithms for frequent itemset mining. Some specification are available in the following website: `http://fimi.cs.helsinki.fi/util/`

implicated. The set FIs and the $IRsSchemas$ are merged together (step 3) to obtain the final set of characterized IRs $IRsSet$. The technique is implemented in the `characterizeIRSchemas(., .)` procedure.
The rule structure drives the construction of the IR since it defines what the involved concepts are and how they are related. The FI instead, identify for each concept what the important attributes are (for the ontology concepts the attributes are represented by the associated datatype properties). Moreover, the pattern discovery process gives us the information about the values of these attributes that correspond to the datatype properties values. The support is, finally, the last component that completes the IRs schemas supplying a measure of the IR strength, the one we call probability of the IRs.

Detailed descriptions and pseudo-codes of `filterFrequentItems` and `characterizeIRSchemas` subroutines are shown in appendix A.3.2 (refer to Algorithms 6 and 7, respectively).

Example 4.5 *Running Example - Part 5: Influence Rules Schema characterization.*

Now it is time to use the instances for characterizing the IRs schemas found at the end of example 4.4. As required by the specifications discussed above, the instances are stored in a file in form of "transactions", that is, in tabular form. Each row represents a particular instance. Since the fragment of the ontology under analysis describes a questionnaire, an instance is a set of the answers for each questions. An example of a file from which we can obtain the FIMI format is the following *arff*[8] file.

```
@relation DatiMPS
@attribute DiversificationOfProduction.hasDivOfProdValue
           {1,2,3,0}
@attribute CustomerBase.hasDiversification {1,2,3,0}
@attribute ManagementTeam.hasYearOfExperience {1,2,3,0}
@attribute PreviousAchievements.hasPrevAchievements {1,2,3,0}
@attribute StrategicVisionAndQualityManagement.hasRate
           {2,3,4,1,0}
@attribute OrganizationalStructure.hasType {1,2,3}
@attribute MarketState.hasTypeOfPhase {2,1,4,3,0}
```

[8]arff is a file format adopted by the famous framework for DM: weka - `www.cs.waikato.ac.nz/ml/weka/`. The arff documentation can be found here: `http://weka.wikispaces.com/ARFF`

```
@attribute ResearchAndDevelopment.isACompanyInvestment {1,2,0}
@attribute LevelOfCompetition.competitionRate {2,1,3,0}
@attribute QualityCertificate.numberOfQCAchieved {3,1,2,0}
@attribute RelationshipWithTheBankingSystem.hasTypeOfRelationship
          {2,1,4,3,0}
@attribute FinancialDebt.hasFinancialDebt {1,2,0}
@attribute CapitalizationStrategy.isTheIncreasingForeseen
          {2,1,0}
@attribute CD_SCORE_TOT {1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,
                         16,17,18,19,20}
@data
1,1,1,1,2,1,2,1,2,3,2,1,2,13
1,1,1,1,3,1,2,1,2,1,2,1,2,13
2,1,1,1,2,1,2,2,1,3,1,1,2,12
3,2,1,1,4,2,2,1,2,1,2,2,1,9
2,2,1,2,2,1,2,1,1,3,2,1,2,10
2,1,1,1,3,2,2,2,1,1,1,1,1,10
2,2,1,2,2,2,2,1,1,3,2,1,2,9
...
2,1,1,1,2,1,2,1,2,3,2,1,2,12
2,2,1,1,2,1,2,2,1,3,2,1,2,11
3,2,1,1,2,1,1,1,1,1,2,1,2,12
2,2,1,1,2,1,3,2,1,3,4,2,2,8
2,1,1,2,3,2,2,1,1,1,2,1,2,9
```

For running PATTERNIST we set the minimum support $minS$ = 40%. The intermediate step, i.e. the output of PATTERNIST, is a set of frequent items in which support is greater than $minS$. The output, before the merging step, is in the following form:

```
FI1:LevelOfCompetition.competitionRate=1
                  FinancialDebt.hasFinancialDebt=1(41.7%)
FI2:StrategicVisionAndQualityManagement.hasRate=2
                  ManagementTeam.hasYearOfExperience=1(62%)
...
FIn:  ...
```

The characterization step merges the IRs Schema and the frequent patterns returning the following IR.

$$\texttt{ManagementTeam.hasYearOfExperience=1} \xrightarrow{62\%} \texttt{StrategicVisionAndQualityManagement.hasRate=2}$$

This rule is the result of the characterization of the IR Schema number 24 in example 4.4 after using the frequent items $FI2$. As one can see, the structure of the rule (the roles of the concepts as implicant or implicated and the types of the concepts) is dictated by the structure of the IRs, while the values (the interesting datatype properties and their values, and the

probability measure) are suggested by the data.
$FI1$ does not produce any IR because there are no Schemas that fit with it.

We observe that no rule that has `MPSQuestionnaire` as implicant, originates an IR. This is because that concept is not interesting, and is not part of the collected data even if it is an important concept in the ontology since it binds many other concepts and is part of the ontology representation model for the questionnaire.

◁

4.5 Validation

In common usage, validation is the process of checking if something satisfies a certain criterion or pre-defined requirements. In computer terminology, validation can also refer to the process of data validation, ensuring that the data inserted into an application satisfy pre-determined formats or comply with stated length and character requirements and other defined input criteria. It may also ensure that only data that is either true or real can be considered.

Also in our context, we intend the verification of the quality of the results: we look at the set of IRs and we discard the "wrong" or "not trustworthy" or not compliant ones. Obviously, the best way to validate the rules set is to ask a domain expert that is able to judge the consistency of the set w.r.t. the domain of interest, and the use one wants to make. In general, what we have to guarantee is that the IRs do not conflict each other, i.e. they do not contain opposite information. The need of validating the IRs is suggested by the fact that the rules are extracted automatically without supervisors (human or artificial agent) and without knowing the final use.
It is important to point out that the validation procedure is for evaluating the semantic interest of the extracted IRs. For this reason we do not define criteria and measures as usually done in a rigorous validation methodology.

A more detailed description of the validation procedure is presented in section 5.1.1, w.r.t. the case study we performed.

Chapter 5

Case Study: the MUSING Project

In these last four chapters we have covered, step by step, the path towards the realization of a new methodology for extracting "knowledge" out of an ontology. We started from the study of the literature at several levels of detail and continued with the investigation of related works, up to the description of the designed and adopted solutions. We have already mentioned the importance of having automatic tools able to retrieve high-level information from various sources (texts, repositories, ontologies, ...), the many applications that one can build by means of these tools, and the several uses that one can do. As you have seen, our approach is completely general and adaptable to different domains and contexts. In this chapter, we describe an actual application of the methodology in the context of MUSING, a European project in which we are involved as a scientific partner. The MUSING platform has been a good environment for developing our research prototype and for testing it with real data. The cooperation with other scientific and technological partners has been indispensable for the success of the work.

5.1 MUSING Project

MUSING,
"**MU**lti-industry, **S**emantic-based next generation business **IN**telli**G**ence" is a European Project started in 2006 that involves several international partners (Mus06). MUSING aims at developing a new generation of Business Intelligence (BI) tools and modules based on semantic knowledge and content systems. It integrates Semantic Web and Human Language technologies and combines declarative rule-based methods and statistical approaches for enhancing the technological foundations of knowledge acquisition and reasoning in BI applications.
It provides exclusive services for three BI areas:

Financial Risk Management. Development and validation of next generation (*Basel II* and beyond) semantic-based BI solutions, with particular reference to Credit Risk Management and access to credit for enterprises, especially Small and Medium Enterprises (SMEs). Figure 33 gives a view of the service.

Internationalisation. Development and validation of next generation semantic-based internationalisation platforms. Internationalisation is the process that allows an enterprise to evolve its business from a local to an international dimension, hereby expressly focusing on the information acquisition work concerning international partnerships, contracts and investments.

IT-Operational Risk & Business Continuity. Development and validation of semantic-driven knowledge systems for measurement and mitigation tools, with particular reference to operational risks faced by IT-intensive organisations. Management of business continuity and operational risks of large enterprises and SMEs impact positively on the related user communities in terms of service levels and costs.

In general, MUSING provides an international framework where partners share data, information, procedures and technical solutions, and enables the partners to use cross-border online services and platforms in

order to support clients in their international and domestic trade activities. It is important to point out that national and international laws and directives, legal practices on privacy and data protection, data sharing and public data management, affect the exchange of data at international levels.
We were mainly involved in the first vertical stream, the Financial Risk Management (FRM), where we shared our experience in data mining and knowledge discovery as well as our skills in ontology analysis.

Figure 33: MUSING services in FRM.

Our main contribution is the development of a (self) assessment tool for the analysis of economic plans and the prediction of a score expressing the quality and worthiness of the company under analysis (MFT+09). The tool is supposed to fit a dual purpose. First, it can be used by Banks and Financial Institutions to support their customer prospecting process and their decision making process. Second, they can be used by Small Medium Enterprises (SMEs) to approach banks on a transparent basis and to receive indications of which strengths and weaknesses they should fo-

cus on, while trying to extend their access to credit (financial inclusion, etc.).

The tool is made concrete by a web-access service that, by providing answers to a questionnaire with qualitative and quantitative questions on business and organisations, delivers back to the user a "Self Assessment Card" containing indications on the company quality and on the credit worthiness. The first version of the Online Self Assessment service, in Italian, is available at:
`http://musing-dev.metaware.it:8380/SelfAssessment/form.html`.
A general idea of the system and the data flow is depicted in figure 34. As

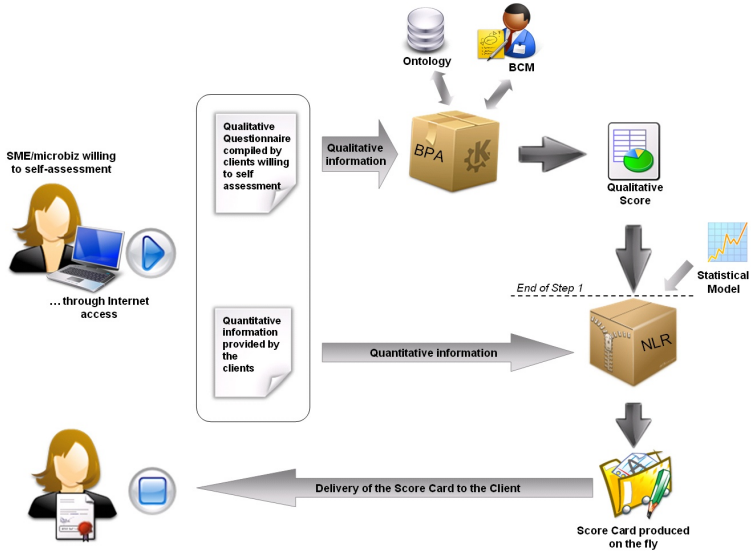


Figure 34: Logic schema of the Online Self Assessment tool.

shown, there are two main tools involved in the system: the Business Plan Analyser (BPA) and the Non-Linear Rating (NLR) . The former is the one we developed for the prevision of a qualitative score, while the latter is

the one developed by the University of Pavia[1] which implements a non-linear rating model for the computation of the company rating based on quantitative and qualitative data. The two modules are logically placed in sequence (the BPA first) following the order of the operations on the data flow. The dotted line, labelled by "End of Step 1" in figure 34, defines the logical separation of the two modules. Data collected by using an online questionnaire originate two separate flows, the qualitative information passes through the BPA and the result of the computation, to the NLR module, while the quantitative ones are sent directly to the NLR module.

The other important source of information is the ontology that contains data and metadata and with which the BPA interacts many times. Without loss of generality, let us focus on the BPA and let us see what the main components are, how they work and where the method proposed in this thesis is applied.

5.1.1 The Extraction and the Use of the IRs in MUSING

The BPA was designed in the attempt of providing SMEs with good self assessment tools not only because of increasing market competition, but also because of new rules in granting credit, as for example the ones referred to as Basel II. One of the critical issues in designing supporting tools is the quality of the knowledge embedded in them. We maintain that a better quality of decisions can be obtained by exploiting not only quantitative, but also qualitative information and expert knowledge. The BPA answers to this need by merging together these data and exploiting new data mining techniques and the ontologies (BBF+08). There are two major issues to be dealt with: finding a suitable representation for data and domain knowledge, and automating the evaluation process followed by the domain expert. Ontologies help us with the first issue because they are, nowadays, the standard way of representing the domain knowledge and storing the corresponding data. To handle the second problem,

[1]The University of Pavia (Italy) participates in the project belong with the two main Departments: the Department of Statistics and Applied Economics, and The Department of Business studies.

we utilized "material" from our previous work on classification (BFT05). There, we presented a methodology for driving the building of classification trees by means of probabilistic rules coded in a Bayesian Causal Map (BCM) and extracted from it during the classification tree building.

Starting from this scenario, the extraction of IRs out of an ontology is applied to the MUSING ontology (in particular to the subset of ontology that describes the qualitative questionnaire), and the IRs are used to enrich the set of Expert Rules (ERs) provided by an expert in economics, and coded in a BCM.

Figure 35 (that is complaint to the logic schema in figure 34) shows the

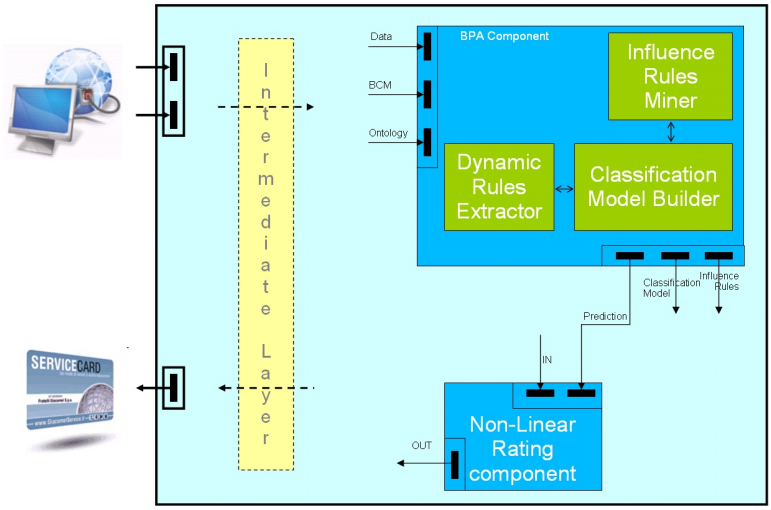


Figure 35: Technical view of the Online Self Assessment service focused on the BPA component.

components of the BPA tool in more detail.

The **Classification Model Builder** is the module for creating the classification and prediction model by using prior external knowledge and the IRs mined from the ontology. The **Influence Rules Miner** implements the extraction of IRs out of the ontology. The **Dynamic Rules Extractor**, instead, finds the rules (both the ones provided by the expert and the ones

mined from the ontology) that are applicable during the construction of the classification tree model with reference to the current construction stage (the path already built). The BPA, in realizing its task, interacts both with the ontology, the interface and with the NLR model. The ontology is the provider of the historical data needed for training the classification engine, while the GUI is the provider of the new data (a new instance of questionnaire to be analysed). The main output is the prevision of the qualitative score that the NLR component takes as input and then uses with the rough quantitative data for computing the final ranking.

Now we will describe in depth how the IR extraction procedure has been customized, starting from the description of the data and the structures, up to the customization of the four steps of the analysis (FTB+09; FTL09).

The Data. The dataset used to train and test the models (both the BPA model and the NLR model) has been provided by the bank Monte dei Paschi di Siena (MPS) [2]. The data set, composed of 6000 records, has a time extension of two years and contains the following information:

- 13 Qualitative Variables representing a subset of the questions included in the Qualitative Questionnaire performed by MPS to assess the credit worthiness of a third party, and in particular utilised to calculate the Qualitative Score of a Company.
- The Qualitative Score (target item of the classification task).
- 80 Financial/Economic indicators calculated from the Balance Sheets and representing a part of the information utilised to foresee the default of a company.

[2]MPS is the Monte dei Paschi di Siena Group's (Italy's fourth largest banking group) Centre of Excellence for eBanking, eBusiness and models innovation. Working closely with the Bank's marketing and strategy functions, MPS is responsible for designing the eBanking and web-based platforms of the Group, as well as managing the implementation process through the internal IT division. MPS has strong competencies and experience in Financial Risk Management (with particular reference to the major Basel II-related issues), Internationalisation, and Operational Risk fields.

- Status of the company. A variable that defines whether the company is currently in *bonis* or in default.

Data Representation and Storage. In agreement with the MUSING specification, all the data have been coded in an ontology.
The logical structure is depicted in figure 36, while the diagram in figure 37 presents the import relationships. As one can see, MUSING is built on a solid net of ontologies each one specialized in a particular area (vertical stream or domain) but strongly inter connected.

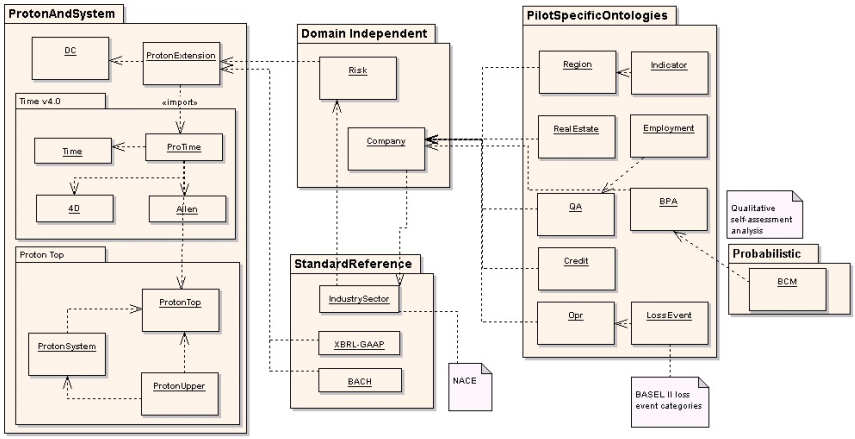


Figure 36: The MUSING Ontologies structure.

The set of *ProtonAndSystem* ontologies comprises generic and axiomatic ontologies. The temporal ontologies are Protime, Allen, 4D and Time; Proton ontologies are System, Top and Upper. The set of *Domain Independent* ontologies is now divided into the Company and Risk ontologies.
The emerging *StandardReference* level consists of IndustrialSector (representing the NACE code[3]), BACH and XBRL-GAAP. The *PilotSpeci-*

[3]The NACE Code is an European classification system which groups organisations ac-

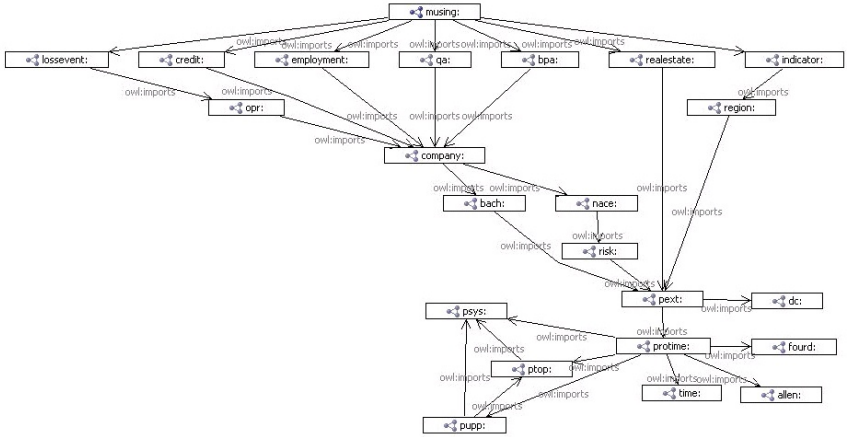


Figure 37: Ontologies imports diagram.

ficOntologies are related to the various pilots of the project (LBS$^+$09). The subset of ontology we are interested in is the *BPA Ontology*, part of the Pilot specific Ontologies. The concepts coded there are related to the qualitative questionnaire (please see Appendix B.2 for the complete list of questions-answers). This ontology represents the domain of application for the IRs extraction engine.

Input Data. The ontology that describes our domain is the BPA ontology, nevertheless, we loaded the whole set of MUSING ontology, so that, starting from the BPA ontology, we can extend the analysis also to the other related concepts allowing the system to exploit the indirect connections between the target concepts (i.e. the concepts related to the qualitative questionnaire).

Extraction of the relevant concepts. The HITSxONTO algorithm has been applied to the MUSING ontologies obtaining a list of 552 concepts and a corresponding adjacency matrix $AdjM$ of size 552x552. The

cording to their business activities (`http://ec.europa.eu/competition/mergers/cases/index/nace_all.html`).

simple $AdjM$ has 2123 non-zero entries (the 6.97% of the total entries). In agreement with the strategy, the exponentiated adjacency matrix $ExpoAdjM$ is computed and adopted for the analysis. This matrix corresponds to:

$$ExpoAdjM = AdjM + \frac{AdjM^2}{2!} + \frac{AdjM^3}{3!},$$

so that paths up to length 3, are exploited. $ExpoAdjM$ has 4415 non-zero entries, 14.49% of the total entries.

The HITSxONTO computation ends after four iterations, returning a list of 5 concepts with $hScore$ greater than 0 and a list of 14 concepts with $aScore$ greater than 0.

Since there are many not sufficiently connected concepts (and the $ExpoAdjM$ has a low number of non-zero entries) involved in the computation, we obtain few concepts with hub and authority ranking greater than 0. Therefore, to obtain a significant number of IRs Schemas we are forced to set low thresholds for both hub and authority.

Without relaxing Ht and At no IR is found because no concept of the questionnaire dataset is part of an IR Schema just obtained.

Construction of the IRs Schemas. The IRs Schemas building action considers the list of candidate implicant concepts with $hScore >= Ht$. If a direct or indirect (up to a maximum path length of 3 steps) connection exists among the candidate implicant concept and the candidate implicated concepts with $aScore >= At$, an IR Schema is generated. Under this parameter setting, the result is a set of 2097 IRs Schemas with exactly one implicant and one implicated.

Characterization of the IRs. As stated earlier, the instances used for this test are about the qualitative variable of the questionnaire. In this case, the instances were already stored separately w.r.t. the ontology; in particular we have a file with the instances organized as transactions, i.e. in tabular form, where each row corresponds to

an instance of the questionnaire (a set of answers coming from the same interviewed subject and related to his company). The number of instances is 5757.

Having set the minimum support $minS$ = 20%, PATTERNIST returned a set of 56 frequent itemset (pairs of concepts).
The step of characterization builds the IRs based on the IR Schemas; in fact, the rule structure drives the construction of the IR since it defines what the involved concepts are and how they are related. The frequent items instead identify, for each concept, what the important attributes and their values are while the support values instead provide a measure of the IR strength.
The result of the characterization of the IRs Schemas by using the set of frequent itemsets, is the following set of 14 IRs:

```
ResearchAndDevelopment.isACompanyInvestment=1 --26%-->
       PreviousAchievements.hasPrevAchievements=1
ResearchAndDevelopment.isACompanyInvestment=1 --30%-->
      CapitalizationStrategy.isTheIncreasingForeseen=2
ResearchAndDevelopment.isACompanyInvestment=2 --28%-->
      PreviousAchievements.hasPrevAchievements=2
StrategicVisionAndQualityManagement.hasRate=2 --28%-->
      CapitalizationStrategy.isTheIncreasingForeseen=2
CapitalizationStrategy.isTheIncreasingForeseen=2 --36%-->
      PreviousAchievements.hasPrevAchievements=2
ManagementTeam.hasYearOfExperience=1 --32%-->
      PreviousAchievements.hasPrevAchievements=2
ResearchAndDevelopment.isACompanyInvestment=2 --31%-->
      PreviousAchievements.hasPrevAchievements=1
StrategicVisionAndQualityManagement.hasRate=2 --42%-->
      PreviousAchievements.hasPrevAchievements=1
CapitalizationStrategy.isTheIncreasingForeseen=2 --48%-->
      PreviousAchievements.hasPrevAchievements=1
ManagementTeam.hasYearOfExperience=1 --54%-->
      PreviousAchievements.hasPrevAchievements=1
ResearchAndDevelopment.isACompanyInvestment=2 --54%-->
      CapitalizationStrategy.isTheIncreasingForeseen=2
StrategicVisionAndQualityManagement.hasRate=2 --60%-->
      CapitalizationStrategy.isTheIncreasingForeseen=2
ManagementTeam.hasYearOfExperience=1 --62%-->
```

```
      StrategicVisionAndQualityManagement.hasRate=2
ManagementTeam.hasYearOfExperience=1 --73%-->
      CapitalizationStrategy.isTheIncreasingForeseen=2
```

In order to interprete correctly these rules, please refer to the description of the qualitative questionnaire and its codification, reported in appendix B.2.
For example, the meaning of the last IR,

```
ManagementTeam.hasYearOfExperience=1 --73%-->
      CapitalizationStrategy.isTheIncreasingForeseen=2
```

is:

> *If the management team has more than* 10 *years of experience in the industrial sector, then the company does not foresee to increase its capital, with a probability of* 73%.

This IR, in agreement with what we just stated, belongs to the following:

```
ManagementTeam → CapitalizationStrategy
```

which is one of the 2097 schemas extracted in the previous phase. Here it is clear that the schema supplies the structure of a set of future IRs; it defines the direction of the implication and what are the involved concepts. The frequent itemset, instead, identifies the interesting datatype properties (related to the considered concepts) and assigns the weight (i.e. the support), making one of the possible instances compatible with that schema.

Validation. As stated in section 4.5, the validation should be done by an expert in the economic field that judges the trustworthiness, the correctness and the compatibility of the IRs. Nevertheless, in the context of MUSING, a sort of "automatic" validation can be done taking into account the final objective of the extraction, i.e. how the IRs are planned to be used. At the beginning of this chapter we said

that in MUSING the IRs are used inside the BPA module for enriching the set of ERs, provided by an expert, that are used for driving the construction of the classification model. The BCM, that contains the ERs, is a cognitive map with associated probabilities; we use it mainly as a representation model rather than a computation model for the rules. The main issue here is to merge IRs and ERs in the same BCM without losing the general coherence of the set, i.e. without creating contrasts (or incompatibilities) among rules.

We can distinguish two main situations:

1. Internal conflict in the IRs set.
2. Conflict in the integration of ERs set with the IRs set when we enrich the BCM.

We can have conflicts inside the IRs set because they are mined automatically from the ontology by using a process that is not driven by expert knowledge and does not follow economic theories. Furthermore, since the generation is influenced by the structure of the ontology, if the ontology suffers some weaknesses the rules can reflect this status. In general, the generation procedure exploits some techniques that should guarantee the consistency of the rules, but the coherence of the rules does not depend only on this process.

The second situation concerns the integration of the ERs set with that of IRs. In this case the conflict is between rules of two different original sets. In fact, once we resolve the conflicts inside the IRs, we are sure that ERs are free from conflicts because an expert provides them and they are validated before their use by the expert itself.

In general, we can have incompatibility when the rules involve same items but different probabilities or they semantically collide (they provide opposite information). In these cases, an action is needed for maintaining the final set of rules consistent. The general idea at the basis of the conflict resolution is to impute more importance to the ERs than to the IRs. The reason is that the IRs set results from an automatic computation that is not strictly driven by the economic

theory (some economic principles are yet hidden in the ontology), while the ERs represent the current literature, the expert belief and the consolidated economic theory.

We identified four main conflict cases, and we propose an action for each one. The details follow (consider the ERs already coded in the BCM).
Cases:

1. Same rules, different probabilities
ER: $A = a \xrightarrow{0.6} B = b$
IR: $A = a \xrightarrow{0.9} B = b$
Action: We keep the rule of the expert.
Result: $A = a \xrightarrow{0.6} B = b$

2. Same precondition, different post-condition (different items)
ER: $A = a \xrightarrow{0.6} B = b$
IR: $A = a \xrightarrow{0.9} C = c$
Action: We add a new rule (the IR).
Result: $A = a \xrightarrow{0.9} C = c$

3. Different precondition (different items), same post-condition.
ER: $A = a \xrightarrow{0.6} B = b$
IR: $C = c \xrightarrow{0.9} B = b$
Action: We add a new rule (the IR).
Result: $C = c \xrightarrow{0.9} B = b$

4. Same precondition, same post-condition variable with different values.
ER: A=a $\xrightarrow{0.6}$ B=b1
IR: A=a $\xrightarrow{0.9}$ B=b2
Action: For solving this case, an input from the domain expert is essential: he or she has to provide a list of attribute-value pairs that, implicated by the same implicant form rules that can coexist in the same map. The action is to add the IR if it does not collide in accordance with the information given by the expert.

In general, if a precondition of a rule is subsumed by another rule (i.e. when there is a rule more general w.r.t. another one), we maintain both the rules.
An example of additional information given by the expert, in the context of MUSING, is provided in Appendix B.1.

5.2 Other Tests

As a demonstration of how the IR extraction system works, we perform several tests on both subsets of the MUSING ontologies and on other ontologies.
Unfortunately, in the latter case, owing to the lack of instances associated to the ontologies, the experimentation stops at the second step, i.e. at the IRs schema extraction.
Table 8 shows the results of the first set of tests in which we use MUSING ontologies subsets. We run the experiments by "playing" with different collections of ontologies, by using different parameter settings and by using the same set of instances (the only available) i.e. the set from the MPS Qualitative questionnaire.

For sake of space and readability, we use labels in the table for identifying tests and metrics. $S1, \ldots, S11$ represent the descriptions of the tests, the metrics, the settings of the experiments, and the results. The meaning is the following:

S1: The dimension of the matrix, $NRows$ x $RColums$.

S2: Percentage of the non-zero entries of the simple adjacency matrix, w.r.t. the total number of the entries.

S3: Percentage of the non-zero entries of the exponentiated adjacency matrix, w.r.t. the total number of the entries.

S4: Number of concepts with hub score greater than zero.

S5: Number of concepts with authority score greater than zero.

S6: Setting: hub threshold (Ht).

S7: Setting: authority threshold (At).

S8: Number of IRs Schemas.

S9: Setting: Minimum support.

S10: Number of Frequent itemset.

S11: Number of IRs.

$T1, \ldots, T3$ are the references to the different tests. It follows the description:

T1: The loaded MUSING ontology is shown in figure 37. The instances are those corresponding to the qualitative questionnaire.

T2: The loaded ontology is a subset of the MUSING ontology where *Temp* is missing. The instances are those corresponding to the qualitative questionnaire.

T3: The loaded ontology is a subset of MUSING: besides *BPA* only *Indicator*, *Employment*, *Company*, *QA*, *Credit* are loaded. These latest are the closest to the *BPA* ontology. The instances are those corresponding to the qualitative questionnaire.

T4: travel.owl available at
`http://www.dcs.bbk.ac.uk/~michael/sw/slides/travel.owl`,
is a travel agency ontology. It includes concepts like Hotel, Restaurant, Activities, Accommodations, ... and corresponding relations for describing the activity of a travel agency.

T5: The New Testament Names ontology is a semantic knowledge base describing each named object in the New Testament. It is available at
`http://www.semanticbible.com/index.html`.

T6: gfo-bio, available at
`http://onto.eva.mpg.de/ontologies/gfo-bio.owl`,
is a biomedical ontology.

Stats	T1	T2	T2	T3
S1	552x552	218x218		126x126
S2	6.97%	2.13%		5.2%
S3	14.49%	4.27%		10.5%
S4	5	60		58
S5	14	63		57
S6	0.0	0.0		0.0
S7	0.0	0.0		0.0
S8	2097	1001		523
S9	40%	40%	20%	20%
S10	56	56	150	150
S11	14	7	14	13

Table 8: Statistics on Tests (part 1).

Stats	T4	T4	T5	T5	T6	T6	T7	T7
S1	34x34		48x48		171x171		1745x1745	
S2	33.91%		8.11%		18.32%		0.25%	
S3	47.75%		12.32%		35.72%		0.46%	
S4	32		26		169		569	
S5	21		16		60		54	
S6	0.0	0.2	0.2	0.0	0.0	> 0.0	> 0.0	> 0.08
S7	0.0	0.2	0.2	0.0	0.0	> 0.0	> 0.0	> 0.0
S8	383	117	12	179	5345	5287	6355	3657

Table 9: Statistics on Tests (part 2).

T7: e-response, available at
`http://e-response.org/ontology/2006/20060406/e-response.owl`,
is an ontology created for the purpose of describing an emergency and the response to that emergency.

Taking a general look at the results, we can say that when the ontology is particularly big, and especially when it is a taxonomy rather than an ontology (i.e. there are few object properties between the concepts) we are forced to relax the constraints to ensure that we obtain some results. In particular, we have to bring down the authority and hub thresholds of acceptance so that the system can return back a sufficient number of "important" concepts for creating the IRs Schemas.
In the tests reported in figure 8, the sets of ontologies we used are typical examples of big ontologies with few object properties (see metrics S1, S2

and S3). In this case, we have been forced to set the thresholds to 0.0 (see metrics S6 and S7), i.e. to consider all the concepts for the construction of the IRs Schema. This has been also induced by the fact that the instances used to characterize the IRs Schemas covered only a small subset of the concepts (the one related to the BPA ontology) included in the ontologies.

From test T1, starting from 2097 IRs Schemas and by using 56 frequent itemsets, the system returns 14 influence rules.
In test T2 we loaded a smaller ontology, i.e. a subset of the ontologies used in test T1, but we maintained the same settings for the thresholds and the minimum support. As one can expect, the number of IRs schemas decreases as well as the number of the final IRs. The number of frequent itemsets is obviously the same, since the instance set does not change.
In test T3, slowing down the minimum support w.r.t. test T2, we obtain a greater number of IRs.
In test T4 we consider a subset of the MUSING ontology that is strongly closed to the collected instances. We don't have loss of information, and under the same conditions w.r.t. the previous cases, we obtain the same number of IRs.

The set of tests in table 9 stops at the construction of the IRs Schemas because in the ontologies we used the instances were missing. In any case, these tests demonstrate that the methodology we developed is fully general and can be applied to whatever domain.

Chapter 6

Conclusions

This thesis has dealt with the issue of extracting new implicit knowledge from an ontology by using both deductive and inductive techniques. This problem is set in a multidisciplinary research context: knowledge discovery and the semantic web, and inherits from both of them useful analysis tools and the most "modern" and expressive representation languages, but also, the whole set of problems related to them. The main issues we faced were to find a point of contact between the two disciplines already explored, a non-trivial task that lead to interesting theoretical and practical results.

Keeping in mind the objective and this series of issues, we organized this dissertation trying to give, at first, a general context of the research, and then presenting the main topics in form of questions.

By exploiting our experience in the data mining field, but especially by taking inspiration from the current literature, we have given an (exhaustive, we hope) answer to each question.

A case study ends this thesis with the aim of showing that the proposed methodology is not only a scientific result, but that it also finds applications in actual contexts.

6.1 What We Did and What Can Still Be Done

Knowledge extraction from databases is a consolidated practice that continues to evolve in parallel with the new storage systems. It is based not only on querying systems, but above all, on complex reasoning tools. Today, with the coming of the Web 2.0, the semantic web, new methods for representing, storing and sharing information are going to replace the traditional systems. Roughly speaking, ontologies "could substitute" in many applications the DBs.
The interest is moving toward the research of new methods for handling these structures and to efficiently obtain information from them.

In the thesis, we have handled the problem of extracting interesting and implicit knowledge out of an ontology, presenting the results in form of influence rules. Our idea was to drive the extraction process by using the ontology structure, and to exploit the instances only in a second step. The main problem was to understand if and how to use traditional methods for data mining in the context of the ontology. Obviously, the traditional systems can be used only as models, but they are not directly applicable to the ontologies.
The solution came quite slowly and in a natural but not trivial way when we decomposed the original problem into subproblems. In this way, we succeeded in finding a methodology taking inspiration from the consolidated theories and recent developments.

We divided the extraction process into 4 main steps:

[Step 1] Identification of the concepts.

The first step is the analysis of the ontology schema and the extraction of the most relevant concepts.
For the extraction, we exploited the possibility of representing the ontology as a graph with its associated Adjacency Matrix (AM). The AM points out the existence of a link between two concepts. For extracting the relevant concepts, we analysed only the schema of the ontology, and used a link analysis method very popular in the semantic web environment. We customized the HITS algorithm by

Kleinberg, implementing a new algorithm able to handle ontologies: HITSxONTO. While HITS works with web pages and hyperlinks, HITSxONTO works on concepts and object properties.

[Step 2] Influence Rule Schema building.

After having pointed out the relevant concepts, we identified the "original" connections (direct and indirect) among concepts by following their object properties. Since we were not interested in the semantics of the properties but in their existence, we were able to simplify the problem and to use the algebra and in particular the matrix theory, for producing these results. Then we associated all the related concepts, building a set of influence rules schemas.

[Step 3] Characterization of the Influence Rules Schemas.

In step 2, we produced only the "skeleton" of our knowledge; in this step we aimed at enriching the schemas with further interesting information hidden in the data (the ontology instances). In particular, we wanted to select among the set of datatype properties associated with each concept, those that were more important, and provide the influence rules with a weight.

This objective drove our research towards the pattern discovery field because it seemed to fit our problem. We analysed the instances related to the metadata in the ontology, and we extracted the frequent items with the associated support value. We did that by using PATTERNIST, an algorithm developed by colleagues at the KDD Lab in Pisa.

[Step 4] Validation.

The Validation is needed to guarantee that the IRs are consistent and do not conflict with each other. The best way for validating the rules, is to ask a domain expert; nevertheless, some *ad hoc* procedures can be implemented with reference to the domain under analysis and the foreseen use.

The first two steps are essentially deductive; they are a sort of "top-down" approach where they start from the theory and try to find some-

thing. The third step is inductive, a sort of "bottom-up" approach; we move from the observations (the instances) to the results (the IRs).
By studying a way of capturing information from the ontologies we have caught many similarities with existent methodologies, as for example graph analysis, social network analysis, link analysis and multi-relational data mining.

In steps 1 and 2, graph theory and linear algebra supplied us with the basis for representing the problem and making it computationally tractable. The link analysis gave us the ideas of how to explore the ontology structure and how to rank the concepts.
In step 3, it was mainly the data mining that supplies the tools. The way of combining all these "ideas" together is certainly part of our original work.
Besides the theoretical results, we had the opportunity of testing our system in an actual case exploiting our involvement in a European industrial research project: MUSING. In this way, we had at our disposal an integrated framework and a real set of data.

As is well known, the problem of finding suitable (for size and coherence) data very often slows down the development and the experimentation of new technologies that need large amount of data. This is very common for data mining applications, where the algorithms build meta models but only the data realizes the applicable models (let us think for example of the classification tree, association rule mining or clustering).
Outside the specific thesis context, in MUSING our main contribution is the development of a self assessment tool for the analysis of economic plans and the prediction of a score expressing the quality and worthiness of the company under scrutiny. The tool is made available in a service accessible via the web that, by providing answers to a questionnaire with qualitative and quantitative questions on business and organisation, delivers back to the user a "Self Assessment Card" containing indications on the company quality and on the credit worthiness. A classification model performs the core component of the analysis. This model uses and merges historical collected data stored in an ontology with new domain knowledge coded in a Bayesian Causal Map and provided by an expert.

This expert knowledge, in form of rules among its items, is used during the construction of the classification model.
In this specific context the extraction of knowledge, the topic of this thesis, has been applied to the subset of the MUSING ontologies dedicated to the description of the questionnaire, and the resulting IRs have been used as feedback to the BCM.

Besides having mined a valuable set of IRs, our analysis tool solves, in this domain, also the problem of the availability of the expert knowledge. In fact, in the economic field, obtaining a cognitive net of relationships from experts is a hard task, either for the complexity of the matter, or for the lack of specific studies (very often these rules are based on the expert believes or his/her own experience).
Speaking in real terms, starting from an ontology of 552 concepts and a dataset of 5757 instances, we mined 14 IRs that involve only the more important concepts and features among the set of concepts we were interested in. This result seems promising and is gratifying to us because the rules are semantically coherent. Obviously, other tests are also necessary (hopefully in different domains) for bringing to light possible weaknesses and to understand how to improve the accuracy of the analysis.

At the end, this work supplies "some highlights" of what we did in this period of research and synthesizes (i.e. implements) some of the possible solutions among the available ones.
Obviously, more can be done, following either the undertaken direction or exploring new solutions.

Concerning the technical aspects, improvements and optimizations can be performed both for the data structures we used, and in the general implementation of the algorithms. In this first release, we probably did not give the right care to the system performance in terms of memory usage and computational speed. During the current experiments, the computational speed has not been a strict constraint and no particular latencies have been registered. For a stand-alone system which runs in local machines this aspect can be left out, but it becomes crucial for online systems where the reply time determines the success of the system itself.

A substantial re-engineerizing of the whole system has already been

planned.
The data structure we used for storing the adjacency matrix, for example, is not particularly "smart" and it is not optimized in the presence of sparse data. On an empirical basis, we can state that the adjacency matrices derived from large ontologies are sparse. For handling the ontologies we used the Protégé and Jena APIs (the last stable release of OWL 1.0) because we already used them, and we acquired some experience with the Protégé framework. Other APIs could be adopted so as to compare the performances in terms of loading speed and reasoning capabilities. More or less for the same reason, we decided to use PATTERNIST as a pattern discovery algorithm. We recall that it is a result of a research performed by colleagues (here in Pisa) in the context of a previous research project and the re-use of shared good software is a common and appreciable practice. At the same way, we could also replace PATTERNIST with alternative algorithms such as the Ferenc Bodon's efficient implementation of the A-priori algorithm or WARMR algorithm.

Bodon's algorithm (Bod03; Bod04) is a new implementation of the well known A-priori algorithm, based on *trie* data structure. This data structure allows essentially to decrease the memory occupation and to speed-up the computation.

WARMR (DT99), developed by Dehaspe and colleagues, is a general purpose Inductive Logic Programming data mining algorithm able to discover knowledge in structured data, where patterns reflect one-to-many and many-to-many relationships in the data. The input is a relational database and the output is a set of conjunctive queries that succeed frequently in that database. On the basis of these frequent queries, query extensions can be generated which are the relational equivalent of association rules obtained with A-priori.

Even if it is interesting to see how the system can change by replacing its components with alternative APIs or algorithms, it is important to point out that no particular missing functionalities have been found, for both the Protégé and Jena APIs and PATTERNIST.

Concerning the strategy for identifying the relevant ontology concepts, a valid alternative to the Kleinberg algorithm is PageRank. It should be

interesting to implement a version that handles the ontologies (as we did for HITS) and compare the results.

A final consideration deals with the application fields. In the thesis, we focused on the economic domain using the IRs for augmenting a set of "similar" (for meaning, structure and objective) rules. Nevertheless, it is important to point out that the system is fully general and can be used in several domains (i.e. in all the domains that can be described by an ontology and where data is available). An alternative and particularly relevant application can be found in the question answering systems. These systems aim at automatically answering questions posed in natural language. The answers are supplied on the basis of the results obtained by querying structured DBs and by processing documents (text or online resources). Today, ontologies are used more and more; domain specific ontologies in the closed domain question answering, and general ontologies in the open-domain question answering. Knowledge extraction from ontology could be used for finding key words or for driving the searching process.

Looking at the kind of output we supply, the IRs remind the Bayesian systems both for their structure (implications with a probability associated) and of their nature. An interesting research line to investigate is the possibility of using the IRs for the automatic generation of these systems. The construction of a BN, for example, requires large efforts and much time from the experts. Moreover, since it is domain dependent, it should be desirable that, when the domains change, the associated BN be modified accordingly. Automating the process could be useful.

A BN is a sort of acyclic graph where each node represents a domain variable, and each arc between nodes represents a probabilistic dependency between them. Moreover, each arc has a conditional associated probability, which indicates how much a node depends on its parent. From this informal definition, some similarities between IRs and a BN clearly emerge. If we consider the whole set of the IRs (concepts and implications) we can redraw a sort of graph where each concept becomes a node and each implication an edge. The probability of the rule becomes

the conditional probability between the associated nodes/concepts.

This idea, i.e. the construction of the net structure and the definition of the probabilities, that can look like a simplistic solution, is actually the general procedure to follow for the generation of a BN. Several current works, that exploit existing and consolidated data structure for constructing BNs, are present in literature (HGC95; Hec96; Nea03; Hec08). In (ZLJ+04), for example, the authors propose an approach based on a Semistructured Probabilistic Database Management System that provides them with robust storage and retrieval mechanisms for large quantities of probability distributions. The main tool, the Bayes Net Builder allows then, knowledge engineers to describe the structure of the Bayes Net model, and Probability Elicitation Tool designed to elicit conditional probability distributions from domain experts. Another case is reported in (LC05), where the authors describe a method of automatically generating BNs from scripts composing knowledge base of conversational agents. It constructs the structure of hierarchically composing nodes and learns the conditional probability distribution table using Noisy-OR model.

Once we solve the non-trivial issue related to the coherence of the structure and the learning strategy to adopt, the automation of the steps would follow a natural progress.
As stated, there are several works based on that direction and the interest in KD represented in BNs is increasing because BNs can handle incomplete data sets and facilitate the combination of domain knowledge and data. From this point of view, a set of IRs can be considered as a set of incomplete data. In general, a set of IRs related to the same pair of concepts do not provide the complete probabilities distributions for the corresponding values.

Appendix A

Technical Specifications

This section is dedicated to a short overview of the technical specifications of the implementation: the programming languages, the main APIs, the framework and support tools.
We also supply the pseudo-code and a more detailed description of the main subroutines of the algorithms we have introduced and described in chapter 4.

As we stated, the ontology miner system we have designed and described in this thesis, has been implemented and it found to work. The system will be available at:

`http://kdd.di.unipi.it/OntologyMiner`

as soon as possible.

A.1 Development Tools

The whole system has been developed by using Eclipse as IDE and Java jdk 1.6 as programming language.

The program uses the following external libraries and external calls:

[Protégé 3.4.1 **lib]** It is an open-source Java library for the Web Ontology Language and RDF(S). The API provides classes and methods to load and save OWL files, to query and manipulate OWL data

models, and to perform reasoning. Furthermore, the API is optimized for the implementation of graphical user interfaces. Both Jena, Core Protégé, Protégé-Frames and Protégé-OWL APIs have been imported. The detail, codes and the corresponding documentation can be found at:
`http://protege.stanford.edu/`.

[Jama 1.0.2 lib] JAMA is a basic linear algebra package for Java. It provides user-level classes for constructing and manipulating real, dense matrices. The class Matrix has been extended for handling the adjacency matrices we use in the system. The documentation can be found at:
`http://math.nist.gov/javanumerics/jama/`.

[Weka 3.6 lib] Weka is a collection of machine learning algorithms for data mining tasks. It contains tools for data pre-processing, classification, regression, clustering, association rules, and visualization. It is also well-suited for developing new machine learning schemes. This library has been used as support (load and filter the instances file) and not for performing data mining actions. Further details and the documentation can be found at:
`http://www.cs.waikato.ac.nz/ml/weka/`.

[PATTERNIST] PATTERNIST is the implementation of the algorithm for extracting the frequent itemset. The system is reachable by means of an external call to the corresponding executable program.

A.2 The System Structure

The system has been divided in nine packages, each one dedicated to the implementation of one of the main system parts. Figure 38 shows the class diagram in UML format; the corresponding list of packages and classes are reported in table 10 and table 11, respectively.

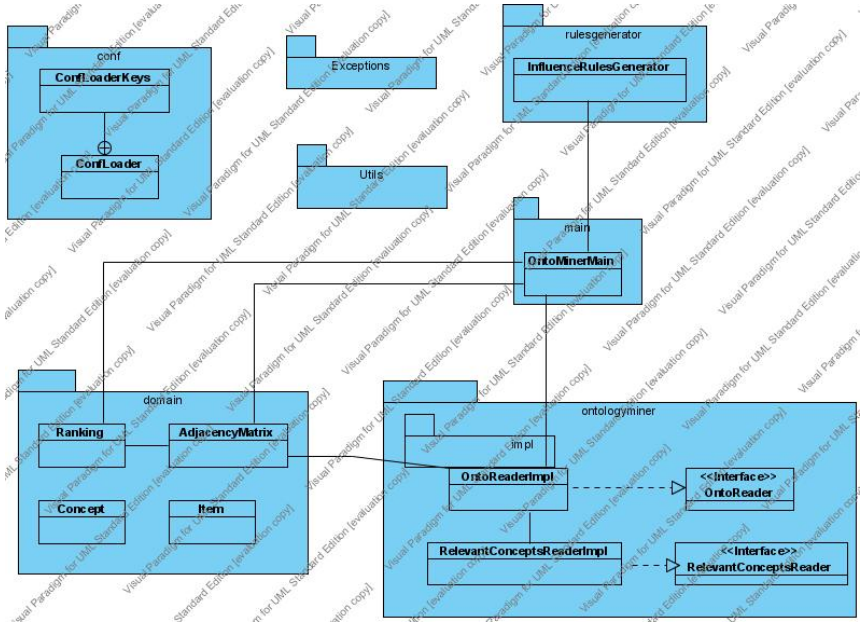


Figure 38: Ontology Miner Class Diagram

[`OntoMinerMain`] It is the main class that triggers the IRs building and composes the various steps of the process.

[`OntoReaderImpl`] It contains the methods for loading and reading the ontology, as well as for handling the lists of concepts extracted from the ontology.

[InfluenceRulesGenerator] It implements two of the main methods, dedicated to the construction of the IR Schema and the IRs.

[AdjMatrix] It extends the Jama Matrix class and implements some methods necessary for performing the operations on the adjacency matrices.

[ConfLoader] It handles the loading of the configuration parameters of the system.

Java Packages
it.unipi.ontologyminer.conceptsranking
it.unipi.ontologyminer.conf
it.unipi.ontologyminer.domain
it.unipi.ontologyminer.exceptions
it.unipi.ontologyminer.main
it.unipi.ontologyminer.ontomanager
it.unipi.ontologyminer.ontomanager.impl
it.unipi.ontologyminer.rulesgenerator
it.unipi.ontologyminer.utils

Table 10: Ontolology Miner: Java Packages.

Java Classes
it.unipi.ontologyminer.conceptsranking ↪ HITSxONTO
it.unipi.ontologyminer.conf ↪ConfLoader
it.unipi.ontologyminer.domain ↪AdjMatrix ↪Concept ↪Item ↪Ranking
it.unipi.ontologyminer.main ↪OntoMinerMain
it.unipi.ontologyminer.ontomanager.impl ↪OntoReaderImpl ↪RelevantConceptsReaderImpl
it.unipi.ontologyminer.rulesgenerator ↪InfluenceRulesGenerator
it.unipi.ontologyminer.utils ↪Constant ↪Converter

Table 11: Ontolology Miner: Java Classes.

A.3 Subroutines Specification

In this section we provide the specification of the main subroutines of algorithms 2 and 3 described in section 4.3 and 4.4, respectively.

A.3.1 Subroutines of Algorithm 2 - section 4.3

[computeIndirectConnections(L, M)]

It computes the number of direct and indirect connections among the concepts. Technically, it computes a new matrix $\bar{M}$ starting from the original adjacency matrix M by implementing the following formula:

$$\bar{M} = \sum_{i=1}^{L} M^i = M + M^2 + M^3 + ... + M^L \tag{A.1}$$

$\bar{M}$ contains the number of all the paths connecting two concepts, of length from 1 to L steps (L is a fixed integer).

```
 INPUT: The Adjacency Matrix:  M
        The length of the paths:  L
 OUTPUT: A new Adjacency Matrix:  M̄
1: exp = 2;
2: M_i = M;
3: M_partial = M;
4: while (exp ≤ L) {
5:     M_temp = multiplyMatrix(M_i, M);
6:     M_i = M_temp;
7:     M̄ = sumMatrix(M_partial, M_temp);
8:     M_partial = M̄;
9:     exp++;
10: }
11: Return M̄;
```

Algorithm 4: Pseudo code of the sub-routine `computeIndirectConnections`.

[connected($\bar{M}$, $implicant$, $implicated$**)]**

It checks if the candidate concepts that become $implicant$ and $implicated$ in a new IR Schema are directly or indirectly connected. This information is easily found in $\bar{M}$ by looking at the value of the matrix entry that corresponds to the two concepts. The method returns $true$ if the matrix entry is greater than 0, $false$ otherwise.

[updateIRSchema($implicant$, $implicated$**)]**

It inserts in the list a new IR Schema composed of the two candidate concepts just analysed.

```
 INPUT: The first concept:  implicant
        The second concept:  implicated
        The list of IR Schemas:  IRSList
 OUTPUT: The updated list:  IRSList
1: if(!(IRSList.contains(implicant, implicated)))
2: then IRSList.add(implicant, implicated);
3: Return IRSList;
```

Algorithm 5: Pseudo code of the sub-routine `updateIRSchema`.

A.3.2 Subroutines of Algorithm 3 - section 4.4

[filterFrequentItems($FIsTemp$**)]**

It filters the list of frequent itemsets generated by PATTERNIST, returning only the interesting ones. We use, as a filtering parameter the number of items each itemset is to be composed of. Since we are interested in creating IRs that contain exactly one implicant and exactly one implicated, we fix this number to 2.

[characterizeIRSchemas($IRSchemas$, FIs**)]**

It merges FIs and $IRsSchemas$ sets to obtain the final set of characterized IRs. The rule schema drives the construction of the IR since it defines what the involved concepts are and how they are related. The frequent itemset instead identifies for each concept what the important attributes are. In algorithm the method `match` finds

```
 INPUT: The list of temporary FIs:  FIsTemp
 OUTPUT: The list of FIs:  FIs
1: foreach(FI ∈ FIsTemp) {
2:    n = FI.getNumberOfItems();
3:    if(n == 2)
4:    then FIs.add(FI);
5: }
6: Return FIs;
```

Algorithm 6: Pseudo code of the sub-routine `filterFrequentItems`.

```
 INPUT: The list of IR Schemas:  IRSchemas
        The list of FIs:  FIs
 OUTPUT: The final list of IRs:  IRs
1: foreach(IRS ∈ IRSchemas) {
2:    if match(IRS.implicant, FI)
3:    then{
4:       IR.implicant = getMatch(IRS.implicant, FI);
5:       IR.implicated = getMatch(IRS.implicated, FI);
6:       IR.probability = FI.support;
7:    }
8:    IRs.add(IR);
9: }
10: Return IRs;
```

Algorithm 7: Pseudo code of the sub-routine `characterizeIRSchemas`.

if there is a correspondence between an IR schema and a frequent itemset. This matching is done by checking if the two items of FI refer to the two concepts that compose the IRS.
Method `getMatch` instead returns the right item of the frequent itemset that matches them with the IR schema implicant or implicated to construct both the implicant and implicated parts of an IR.

Appendix B

Additional Information

B.1 Additional Information Provided By the Expert.

This is an example of a document provided by the domain expert, in the context of MUSING for solving the issue of IRs consistency Case 4.
This document strongly relates to the variable used in the part of the project under analysis.

- Diversification of production.

1. The company operates in more than one sector.
2. The company operates in just one sector with flexible production processes.
3. The company operates in just one sector with no flexible production processes.

The three options are exclusive, so they cannot coexist.

- Commercial diversification.

1. Customers base well diversified, with no concentration of sales.
2. Customers base well diversified, with some key clients.

3. Most of sales directed to a few key clients.

Options 1 and 2 can coexist, while 3 collides with both 1 and 2.

- **Years of experience of the management team in the industrial sector the company operates in:**

1. > 10
2. $5 - 10$
3. < 5

The three options are exclusive, so they cannot coexist.

- **Previous achievements of the management team.**

1. Company owner / CEO with past successful achievements even in different fields from the one in which the company operates today.
2. Company owner / CEO with no relevant past experiences.
3. Company owner / CEO with one or more unsuccessful past experiences.

Options 1 and 3 collide, while 2 can coexist with the others.

- **Strategic vision and quality of management (referred to previous experiences).**

1. Excellent.
2. Good.
3. Satisfying.
4. Insufficient.

Options 1, 2 and 3 can coexist, while 4 collides with the others.

- **Organisational structure of the company.**

1. Organised in a well-articulate and efficient way.

2. Well organised even if some gaps are present, all the relevant positions are well covered.
3. The organisation is not adequate to the company dimension and some relevant positions are not presided.

Options 1 and 2 can coexist, while 3 collides with the others.

- **Market trend.**

1. Growing.
2. Stable.
3. Going toward stabilisation.
4. In recession.

Option 1 collides with 4 because they describe completely opposite market status. Options 2 and 3 are quite neutral and can coexist with the others.

- **Does the company invest in Research and Development?**

1. Yes.
2. No.

Obviously in contrast.

- **Level of competition in the market.**

1. High.
2. Medium.
3. Low.

Option 1 collides with 3 because they are opposite, while 2 is quite neutral w.r.t. to the others.

- **Quality certificate(s) achieved.**

1. The company achieved one or more quality certificates.

2. The company has one or more quality certificates requests in progress.
3. The company does not have any quality certificates.

Options 1 collides with 3, while 2 is quite neutral w.r.t. to the others.

- **Relationships with the banking system.**

1. Good margin of utilisation of the credit lines and good credit worthiness.
2. Good margin of utilisation of the credit lines.
3. Presence of some tensions.
4. Overdrafts are present.

Options 1 and 2 can coexist. Options 3 and 4 can coexist.

- **Is the company foreseeing to increase its capitalisation?**

1. Yes.
2. No.

Obviously in contrast.

B.2 Qualitative Questionnaire.

The qualitative questionnaire aims at collecting the qualitative information of the company/financial institution that accesses the Online Self Assessment service. Here is the list of questions and the corresponding option answers.
For being processed, the questionnaire has been suitable codified. In the ontology, at schema level, each question is a datatype property of a concept.
The codification, with the syntax `Concept.datatypeProperty`, is also provided.

- **Diversification of production.**

  1. The company operates in more than one sector.
  2. The company operates in just one sector with flexible production processes.
  3. The company operates in just one sector with no flexible production processes.

  `DiversificationOfProduction.hasDivOfProdValue`

- **Commercial diversification.**

  1. Customers base well diversified, with no concentration of sales.
  2. Customers base well diversified, with some key clients.
  3. Most of sales directed to few key clients.

  `CustomerBase.hasDiversification`

- **Years of experience of the management team in the industrial sector the company operates in.**

  1. > 10.
  2. $5 - 10$.
  3. < 5.

`ManagementTeam.hasYearOfExperience`

- **Previous achievements of the management team.**

  1. Company owner/CEO with past successful achievements even in different fields from the one in which the company operates today.
  2. Company owner/CEO with no relevant past experiences.
  3. Company owner/CEO with one or more unsuccessful past experiences.

`PreviousAchievements.hasPrevAchievements`

- **Strategic vision and quality of management (referred to previous experiences).**

  1. Excellent.
  2. Good.
  3. Satisfying.
  4. Insufficient.

`StrategicVisionAndQualityManagement.hasRate`

- **Organisational structure of the company.**

  1. Organised in a well-articulate and efficient way.
  2. Well organised even if some gaps are present, all the relevant positions are well covered.
  3. The organisation is not adequate to the company dimension and some relevant positions are not presided.

`OrganizationalStructure.hasType`

- **Market trend.**

  1. Growing.
  2. Stable.

3. Going toward stabilization.
4. In recession.

`MarketState.hasTypeOfPhase`

- **Does the company invest in Research & Development?**

1. Yes.
2. No.

`ResearchAndDevelopment.isACompanyInvestment`

- **Level of competition in the market.**

1. High.
2. Medium.
3. Low.

`LevelOfCompetition.competitionRate`

- **Quality certificate(s) achieved.**

1. The company achieved one or more quality certificates.
2. The company has one or more quality certificates requests in progress.
3. The company does not have any quality certificates.

`QualityCertificate.numberOfQCAchieved`

- **Relationships with the banking system.**

1. Good margin of utilisation of the credit lines and good credit worthiness.
2. Good margin of utilisation of the credit lines.
3. Presence of some tensions.
4. Overdrafts are present.

`RelationshipWithTheBankingSystem.hasTypeOfRelationship`

- **Financial requirements trend.**

    1. In line with the company dynamics.
    2. Not in line with the company dynamics.

    `FinancialDebt.hasFinancialDebt`

- **Is the company foreseeing to increase its capitalisation?**

    1. Yes.
    2. No.

    `CapitalizationStrategy.isTheIncreasingForeseen`

References

[Aba00] M. Abate. *Algebra lineare*. McGraw-Hill, 2000. 55

[ACPT99] P. Atzeni, S. Ceri, S. Paraboschi, and R. Torlone. *Database Systems - Concepts, Languages and Architectures*. McGraw-Hill, 1999. 1

[AGO97] A. Albano, G. Ghelli, and R. Orsini. *Relational and Object Databases*. Zanichelli, Bologna, 1997. 1

[AI97] T. C. Almind and P. Ingwersen. Informetric analyses on the world wide web: Methodological approaches to "webometrics". *Journal of Documentation.*, 53(4):404–426, 1997. 52

[AIS93] R. Agrawal, T. Imielinski, and A. N. Swami. Mining association rules between sets of items in large databases. In Peter Buneman and Sushil Jajodia, editors, *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, pages 207–216, Washington, D.C., 26–28 1993. 13, 69, 70

[AMS+96] R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, and A. I. Verkamo. *Fast discovery of association rules.*, pages 307–328. AAAI Press, Menlo Park, CA, 1996. 69, 70

[AP05] M. Agosti and L. Pretto. A theoretical study of a generalized version of Kleinberg's HITS algorithm. *Inf. Retr.*, 8(2):219–243, 2005. 3, 61, 95, 97, 98

[AriCE] Aristotle. *Categories*. The Internet Classic Archive, (350 B.C.E). 15, 18

[AS94] R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In *Proceedings of the Twentieth International Conference on Very Large Data Bases (VLDB'94).*, pages 487–499, 1994. 69, 70

[AS95] R. Agrawal and R. Srikant. Mining sequential patterns. In Philip S. Yu and Arbee S. P. Chen, editors, *Eleventh International Conference on Data Engineering*, pages 3–14, Taipei, Taiwan, 1995. IEEE Computer Society Press. 13

[AS02] K. Anyanwu and A. Sheth. The ρ operator: Discovering and ranking on the semantic web. *SIGMOD Record*, 31(4):42–47, 2002. 62, 63

[Bar04] S. Barton. Designing indexing structure for discovering relationships in RDF graphs. In *Proceedings of the Dateso 2004 Annual International Workshop on Databases.*, pages 7–17, 2004. 62

[BBF+08] M. Baglioni, A. Bellandi, B. Furletti, L. Spinsanti, and F. Turini. Ontology-based business plan classification. In *EDOC '08: Proceedings of the 2008 12th International IEEE Enterprise Distributed Object Computing Conference*, pages 365–371, Washington, DC, USA, 2008. IEEE Computer Society. 115

[Bel07] L. Bellini. YaDT-DRb : Yet another Decision Tree Domain Rule builder. Master's thesis, University of Pisa, Italy, 2007. 14

[BFT05] M. Baglioni, B. Furletti, and F. Turini. DrC4.5: Improving C4.5 by means of prior knowledge. In *SAC '05: Proceedings of the 2005 ACM symposium on Applied computing*, pages 474–481, New York, NY, USA, 2005. ACM Press. 14, 116

[BFT06] M. Baglioni, B. Furletti, and F. Turini. A tool for economic plans analysis based on expert knowledge and data mining techniques. In *Proceedings of the IADIS International Conference - Applied Computing 2006*, pages 421–426, 2006. 13

[BGL+06] F. Bonchi, F. Giannotti, C. Lucchese, S. Orlando, R. Perego, and R. Trasarti. Conquest: a constraint-based querying system for exploratory pattern discovery. In *ICDE*, 2006. 3, 73

[BGS02] M. Bianchini, M. Gori, and F. Scarselli. PageRank a circuital analysis. In *Proceedings of the 11th International Word Wide Web Conference (WWW 2002).*, Honolulu, Hawaii., 2002. 53

[BH06] S. Bausch and L. Han. Top 10 social networking sites, april 2006. source: Nielsen/netratings. `http://www.nielsen-netratings.com/pr/pr_060511.pdf`, 2006. 74

[BI99] J. Bar-Ilan. Search engine results over time - a case study on search engine stability. *Cybermetrics.*, 2/3, 1999. 52

[Bie03] D. Bielfeldt. Ontology. *The Encyclopedia of Science and Religion. Vol. 2*, page 632, 2003. 2, 16

[BIP04] J. Bar-Ilan and B. C. Peritz. Evolution, continuity, and disappearance of documents on a specific topic on the web: A longitudinal study of "informetrics". *Journal of the American Society for Information Science and Technology.*, 55(11):980–990, 2004. 52

[BL04] F. Bonchi and C. Lucchese. On closed constrained frequent pattern mining. In *ICDM*, pages 35–42, 2004. 3, 73

[BL05] F. Bonchi and C. Lucchese. Pushing tougher constraints in frequent pattern mining. In *PAKDD*, pages 114–124, 2005. 3, 73

[BLF97] T. Berners-Lee and M. Fischetti. *Weaving the Web*. Harper San Francisco, 1997. 23

[Bod03] F. Bodon. A fast APRIORI implementation. In Bart Goethals and Mohammed J. Zaki, editors, *Proceedings of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations (FIMI'03)*, volume 90 of *CEUR Workshop Proceedings*, Melbourne, Florida, USA, November 2003. 134

[Bod04] F. Bodon. Surprising results of trie-based FIM algorithms. In Bart Goethals, Mohammed J. Zaki, and Roberto Bayardo, editors, *Proceedings of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations (FIMI'04)*, volume 126 of *CEUR Workshop Proceedings*, Brighton, UK, November 2004. 134

[BP90] J. Bigelow and R. Pargetter. *Science and Necessity*. Cambridge University Press, 1990. 17

[BP98] S. Brin and L. Page. The anatomy of a large-scale hypertextual Web search engine. In *Proceedings of the 7th International World Wide Web Conference.*, Brisbane, Australia., 1998. 51, 52

[BRdS06] J. Bollen, M. A. Rodriguez, and H. Van de Sompel. Journal status. *CoRR*, abs/cs/0601030, 2006. 53

[Cap05] N. Capuano. Ontologie OWL: Teoria e pratica. prima parte. *Computer Programming*, 148, 2005. 27

[CB74] D. D. Chamberlin and R. F. Boyce. SEQUEL: A Structured English QUEry Language. In *SIGMOD Workshop, Vol. 1*, pages 249–264, 1974. 1

[CMH03] D. J. Cook, N. Manocha, and L. B. Holder. Using a graph-based data mining system to perform web search. 17 - n.5:705–720, 2003. 3, 65

[Cod70] E. F. Codd. A relational model of data for large shared data banks. *Communications of the ACM.*, 13(6), 1970. 1

[Com04a] W3C Community. *OWL Web Ontology Language Overview.* `http://www.w3.org/TR/2004/REC-owl-features-20040210/`, 2004. 27, 30, 87, 92

[Com04b] W3C Community. *RDF Vocabulary Description Language 1.0: RDF Schema.* `http://www.w3.org/TR/rdf-schema/`, 2004. 26

[Die00] R. Diestel. *Graph Theory (Graduate Texts in Mathematics)*, volume 173. Springer-Verlag, New York, February 2000. 3

[Dlg] Description Logics official site. `http://dl.kr.org/`. 23

[DT99] L. Dehaspe and H. Toivonen. Discovery of frequent DATALOG patterns. *Data Mining and Knowledge Discovery*, 3(1):7–36, Mar 1999. 134

[Dun92] R. I. M. Dunbar. Neocortex size as a constraint on group size in primates. *Journal of Human Evolution*, 22(6):469–493, 1992. 76

[Dze03] S. Dzeroski. Multi-relational data mining: an introduction. *SIGKDD Explor. Newsl.*, 5(1):1–16, 2003. 66

[FLM+06] A. Farahat, T. Lofaro, J. C. Miller, G. Rae, and L. A. Ward. Authority rankings from HITS, PageRank, and SALSA: Existence, uniqueness, and effect of initialization. *SIAM J. Sci. Comput.*, 27(4):1181–1201, 2006. 3, 53, 58, 60, 95, 97

[FM04] I. Fischer and T. Meinl. Graph based molecular data mining - an overview. In *IEEE International Conference on Systems, Man and Cybernetics, 2004*, volume 5, pages 4578–4582, October 2004. 65

[FTB+09] B. Furletti, F. Turini, D. Bachlechner, M. Spies, and C. Leibold. Deliverable D4.3 (WP4) - Ontology Learning Report, Rel 1. Project deliverable, March 2009. 117

[FTL09] B. Furletti, F. Turini, and C. Leibold. Deliverable D4.3 (WP4) - Ontology learning and mining report/prototype, Rel 2. Project deliverable, November 2009. 117

[Goc80] R. Gockelin. *Lexicon philosophicum*. Reprinted by Georg Olms 2 edition 1980, 1980. 16

[GW00] N. Guarino and C. A. Welty. A formal ontology of properties. In *Knowledge Acquisition, Modeling and Management*, pages 97–112, 2000. 19

[GW02] N. Guarino and C. Welty. Evaluating ontological decisions with OntoClean. *Communications of the ACM*, 45(2):61–65, 2002. 43

[HD06] N. Huang and S. Diao. Structured-based ontology evaluation. In *Proceedings of the IEEE International Conference on e-Business Engeneering (ICEBE 06).*, pages 7–17, 2006. 44

[Hec96] D. Heckerman. Bayesian networks for knowledge discovery. In *Advances in Knowledge Discovery and Data Mining*, pages 273–305. AAAI/MIT Press, 1996. 14, 136

[Hec08] D. Heckerman. A Tutorial on Learning with Bayesian Networks. In *Innovations in Bayesian Networks*, pages 33–82. 2008. 136

[HGC95] D. Heckerman, D. Geiger, and D. M. Chickering. Learning Bayesian Networks: The Combination of Knowledge and Statistical Data. *Machine Learning*, 20(3):197–243, 1995. 136

[HK00] J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers, 2000. 2, 11

[HMS01] D. Hand, H. Mannila, and P. Smyth. *Principles of Data Mining*. The MIT Press., Cambridge, Massachusetts London England, 2001. 2

[Hor05] I. Horrocks. OWL rules, ok? In *Rule Languages for Interoperability*, 2005. 34

[IA06] *Intelligenza Artificiale - Rivista Trimestrale dell'Associazione Italiana per l'IA*. Marzo–Giugno 2006. 21

[JE07] J. J. Jung and J. Euzenat. Towards semantic social networks. In *Proceedings of the European Semantic Web Conference (ESWC2007)*, volume 4519 of *LNCS*, pages 267–280. Springer-Verlag, July 2007. 77, 79

[Jia06] B. Jiang. Ranking spaces for predicting human movement in an urban environment. `http://arxiv.org/ftp/physics/papers/0612/0612011.pdf`, 2006. 54

[KAC+02] G. Karvounarakis, S. Alexaki, V. Christophides, D. Plexousakis, and M. Scholl. RQL: A declarative query language for RDF. In *Proceedings of the 11th International Word Wide Web Conference (WWW 2002).*, Honolulu, Hawaii., 2002. 62

[Kle98] J. Kleinberg. Authoritative sources in a hyperlinked environment. In *Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms.*, pages 668–677, 1998. 3, 51, 55, 95

[KMS02] B. Kemmerer, S. Mishra, and P. P. Shenoy. Bayesian causal maps as decision aids in venture capital decision making: Methods and applications. In *In proceedings of the Accademy of Management Conference*, 2002. 13

[Koe04] W. Koehler. A longitudinal study of web pages continued: a report after six years. *Information Research.*, 9(2), 2004. 52

[LBS+09] C. Leibold, D. Bachlechner, M. Spies, H. U. Krieger, and B. Kiefer. Deliverable d3.1 (wp3) - report on ontologies development & management (rel. 2). Project deliverable, 2009. 119

[LC05] S. Lim and S. Cho. Automatic Construction of Bayesian Networks for Conversational Agent. In *ICIC (2)*, pages 228–237, 2005. 136

[Lor] J. Lorhard. *Theatrum philosophicum*. Basilea, SECOND edition. 16

[LS99] O. Lassila and R. R. Swick. Resource Description Framework (RDF) model and syntax specification. `http://www.w3c.org/TR/REC-rdf-syntax`. W3c recommendation, W3C, 1999. 24

[Mea67] G. H. Mealy. Another look at data. In *Proceedings of the Fall Joint Computer Conference, Volume 31*, pages 525–534. Thompson Books, London: Academic Press, 1967. 18

[MFT+09] S. Miniati, B. Furletti, F. Turini, S. Figini, and C. Leibold. PILOT 3.1 (WP6) - Online Self-Assessment. Project working document, October 2009. 113

[Mus06] MUSING Project. `http://www.musing.eu/`, 2006. 5, 6, 112

[MW85] Z. Manna and R. Waldinger. *The logical basis for computer programming. Volume 1: deductive reasoning*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1985. 20, 21

[Nea03] R. E. Neapolitan. *Learning Bayesian Networks*. Prentice hall edition, April 2003. 136

[Nil02] N. J. Nilsson. *Intelligenza Artificiale*. Apogeo, 2002. 20

[NS93] A. Nerode and R. A. Shore. *Logic for applications*. Springer Verlag, 1993. 21

[ØAS05] P. Øhrstrøm, J. Andersen, and H. Schärfe. What has happened to ontology. In Frithjof Dau, Marie-Laure Mugnier, and Gerd Stumme, editors, *Proceedings of the 13th International Conference on Conceptual Structures (ICCS 2005)*, volume 3596 of *Lecture Notes in Computer Science*, pages 425–438. Springer, 2005. 15

[PN76] G. Pinski and F. Narin. Citation influence for journal aggregates of scientific publications: Theory, with application to literature of physics. *Information Processing and Management*, 12(5):297–312, 1976. 51

[PS08] E. Prud'hommeaux and A. Seaborne. SPARQL query language for RDF. `http://www.w3.org/TR/2008/REC-rdf-sparql-query-20080115/`. W3C Recommendation, January 2008. 2

[PSFS96] G. Piatetsky-Shapiro, U. Fayyad, and P. Smyth. From data mining to knowledge discovery in databases ontologies. 1996. 9

[Qui86] J. Ross Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986. 2

[Qui93] J. Ross Quinlan. *C4.5: programs for machine learning*. Morgan Kaufmann Publishers Inc., 1993. 14

[Rug04] S. Ruggieri. YaDT: Yet another Decision Tree Builder. In *ICTAI*, pages 260–265, 2004. 14

[SGPD+04] Y. Sure, A. Gomez-Perez, W. Daelemans, M.L. Reinberg, N. Guarino, and N. F. Noy. Why evaluate ontology technologies? because it works! *IEEE Intelligent Systems*, pages 1541–1672, 2004. 44

[SHB06] G. Stumme, A. Hotho, and B. Berendt. Semantic web mining: State of the art and future directions. *J. Web Sem.*, 4(2):124–143, 2006. 2

[Smi] B. Smith. Ontology and information systems. url: `http://ontology.buffalo.edu/smith/articles/ontologies.htm`. 2, 18

[Smi03] B. Smith. *Preprint version of chapter "Ontology"*, pages 155–166. Oxford: Blackwell, 2003. 2, 16, 18

[The06] M. Thelwall. Interpreting social science link analysis research: A theoretical framework. *Journal of the American Society for Information Science and Technology.*, 57(1):60–68, 2006. 52

[WF94] S. Wasserman and K. Faust. *Social Network Analysis: Methods and Applications (Structural Analysis in the Social Sciences)*. Cambridge University Press, 1 edition, November 1994. 51

[WGZP04] X. H. Wang, T. Gu, D. Q. Zhang, and H. K. Pung. Ontology based context modeling and reasoning using OWL., 2004. 31

[Wik] Description Logic from Wikipedia, the free encyclopedia. http://en.wikipedia.org/wiki/Description_logic. 23

[WM03] T. Washio and H. Motoda. State of the art of graph-based data mining. *SIGKDD Explor. Newsl.*, 5(1):59–68, July 2003. 3, 38, 43

[Yan] John A. Yanosy. An ontology to document relevant information about a person's career. http://protege.stanford.edu/plugins/owl/owl-library/resume.owl. 46

[ZLJ+04] W. Zhao, J. Li, E. Jessup, A. Dekhtyar, and J. Goldsmith. Building Bayes Nets with Semistructured Probabilistic DBMS. *EMISA Forum*, 24(1):28–29, 2004. 136