Association Rule Interactive Post-processing using Rule Schemas and Ontologies - ARIPSO

Claudia Marinica

Association Rule Interactive Post-processing using Rule Schemas and Ontologies - AR*IPSO*

By

Claudia Marinica

A thesis submitted to "Ecole poltechnique de l'Université de Nantes"
Department of Computer Science
for the degree of

Doctor of Philosophy

26 October 2010

Committee :

Reviewers:	Fabien Gandon	Full-time Researcher	INRIA Sophia-Antipolis
	Jean-Gabriel Ganascia	Full Professor	Université Pierre et Marie Curie Paris
Supervisor:	Fabrice Guillet	Full Professor	Polytech'Nantes
President:	Marie-Aude Aufaure	Full Professor	Ecole Centrale de Paris
Examiners:	Henri Briand	Professor Emeritus	Polytech'Nantes
	Pascale Kuntz	Full Professor	Polytech'Nantes

To my family.

Abstract/Résumé

Abstract

This thesis is concerned with the merging of two active research domains: Knowledge Discovery in Databases (KDD), more precisely the Association Rule Mining technique, and Knowledge Engineering (KE) with a main interest in knowledge representation languages developed around the Semantic Web.

In Data Mining, the usefulness of association rule technique is strongly limited by the huge amount and the low quality of delivered rules. Experiments show that rules become almost impossible to use when their number exceeds 100. At the same time, nuggets are often represented by those rare (low support) unexpected association rules which are surprising to the user. Unfortunately, the lower the support is, the larger the volume of rules becomes. Thus, it is crucial to help the decision maker with an efficient technique to reduce the number of rules.

To overcome this drawback, several methods have been proposed in the literature such as itemset concise representations, redundancy reduction, filtering, ranking and post-processing. Even though rule interestingness strongly depends on user knowledge and goals, most of the existing methods are generally based on data structure. For instance, if the user looks for unexpected rules, all the already known rules should be pruned. Or, if the user wants to focus on specific family of rules, only this subset of rules should be selected.

In this context, we address two main issues: the integration of user knowledge in the discovery process and the interactivity with the user. The first issue requires defining an adapted formalism to express user knowledge with accuracy and flexibility such as ontologies in the Semantic Web. Second, the interactivity with the user allows a more iterative mining process where the user can successively test different hypotheses or preferences and focus on interesting rules.

The main contributions of this work can be summarized as follows:

(i) A model to represent user knowledge. First, we propose a new rule-like formalism, called Rule Schema, which allows the user to define his/her expectations regarding the rules through ontology concepts. Second, ontologies allow the user to express his/her domain knowledge by means of a high semantic model. Last, the user can choose among a set of Operators for interactive processing the one to be applied over each Rule Schema (i.e. pruning, conforming, unexpectedness, . . .).

(ii) A new post-processing approach, called AR*IPSO* (Association Rule Interactive Post-processing using rule Schemas and Ontologies), which helps the user to reduce the volume of the discovered rules and to improve their quality. It consists in an interactive process integrating user knowledge and expectations by means of the proposed model. At each step of AR*IPSO*, the interactive loop allows the user to change the provided information and to reiterate the post-processing phase which produces new results.

(iii) The implementation in post-processing of the proposed approach. The developed tool is complete and operational, and it implements all the functionalities described in the approach. Also, it makes the connection between different elements like the set of rules and rule schemas stored in PMML/XML files, and the ontologies stored in OWL files and inferred by the Pellet reasoner.

(iv) An adapted implementation without post-processing, called AR*LIUS* (Association Rule Local mining Interactive Using rule Schemas), consisting in an interactive local mining process guided by the user. It allows the user to focus on interesting rules without the necessity to extract all of them, and without minimum support limit. In this way, the user may explore the rule space incrementally, a small amount at each step, starting from his/her own expectations and discovering their related rules.

(v) The experimental study analyzing the approach efficiency and the discovered rule quality. For this purpose, we used a real-life and large questionnaire database concerning customer satisfaction. For AR*IPSO*, the experimentation was carried out in complete cooperation with the domain expert. For different scenarios, from an input set of nearly 400 thousand association rules, AR*IPSO* filtered between 3 and 200 rules validated by the expert. Clearly, AR*IPSO* allows the user to significantly and efficiently reduce the input rule set. For AR*LIUS*, we experimented different scenarios over the same questionnaire database and we obtained reduced sets of rules (less than 100) with very low support.

Keywords: Data Mining, Association Rules, Ontologies, Semantic Web

Résumé

Cette thèse s'inscrit à la confluence de deux domaines actifs de recherche: l'Extraction de Connaissances à partir des Données (ECD), plus précisément la technique de fouille de Règles d'Association, et l'Ingénierie des Connaissances (KE), en s'intéressant particulièrement aux principaux langages de représentation de connaissances développés autour du Web Sémantique.

Dans le domaine de l'extraction de connaissances, l'utilité de la technique de fouille de règles d'association est fortement limitée par la quantité énorme et par la faible qualité des règles découvertes. Les différents tests montrent que les règles deviennent presque impossibles à utiliser dès que leur nombre dépasse 100. De plus, les pépites (nuggets) sont souvent représentées par ces règles d'association rares (support faible) et inattendues, surprenantes pour l'utilisateur. Malheureusement, plus le support est faible, plus le volume des règles sera important. Ainsi, il est essentiel d'aider le décideur avec une technique efficace de réduction du nombre de règles.

Pour réduire cet inconvénient, plusieurs méthodes ont été proposées dans la littérature comme les représentations concises de motifs, la réduction des redondances, le filtrage, le ranking et le post-traitement. Même si l'intérêt des règles dépend fortement des connaissances et objectifs de l'utilisateur, la plupart des méthodes existantes sont généralement basées sur la structure des données. Par exemple, si l'utilisateur cherche des règles inattendues, toutes les règles déjà connues doivent être élaguées. Ou, si l'utilisateur veut se concentrer sur une famille spécifique de règles, seul ce sous-ensemble de règles devra être sélectionné.

Dans ce contexte, nous abordons deux thèmes principaux: l'intégration des connaissances de l'utilisateur dans le processus de découverte et l'interactivité avec l'utilisateur. Le première problème exige la définition d'un formalisme adapté afin d'exprimer les connaissances de l'utilisateur avec précision et flexibilité comme les ontologies dans le Web Sémantique. Deuxièmement, l'interactivité avec l'utilisateur permet la mise en oeuvre d'un processus d'exploration plus itératif où l'utilisateur peut tester successivement des hypothèses et des préférences différentes, lui permettant ainsi de se concentrer sur les règles intéressantes.

Les principales contributions de ce travail peuvent être résumées comme suit:

(i) Un modèle pour représenter les connaissances de l'utilisateur. Premièrement, nous proposons un nouveau formalisme de règles, appelé Schéma de Règles, qui permet à l'utilisateur de définir, à travers des concepts ontologiques, ses attentes concernant les règles. Deuxièmement, les ontologies permettent à l'utilisateur d'exprimer, à l'aide d'un modèle sémantique de haut niveau, ses connaissances de domaine. Enfin, l'utilisateur peut choisir parmi un ensemble d'Opérateurs de traitement interactif celui à appliquer sur chaque schéma de règles (élagage, conforme, inattendu, ...).

(ii) Une nouvelle approche de post-traitement, nommée AR*IPSO* (Association Rule Interactive Post-processing using rule Schemas and Ontologies), qui permet à l'utilisateur de réduire le volume de règles découvertes et d'améliorer leur qualité. Il consiste en un processus interactif intégrant les connaissances et les attentes de l'utilisateur à l'aide du modèle proposé. La boucle interactive permet à l'utilisateur, à chaque étape du AR*IPSO*, de modifier les informations fournies et de réitérer la phase de post-traitement qui produit des nouveaux résultats.

(iii) L'implémentation en post-traitement de l'approche proposée. L'outil développé est complet et opérationnel, et il met en oeuvre toutes les fonctionnalités décrites dans l'approche. En outre, il fait le lien entre les différents éléments comme l'ensemble de règles et de schémas de règles stocké dans des fichiers PMML/XML, et les ontologies stockées dans des fichiers OWL et inférées à l'aide du raisonneur Pellet.

(iv) Une implémentation adaptée, sans post-traitement, nommée AR*LIUS* (Association Rule Local Interactive mining Using rule Schemas), consistant en un processus d'exploration locale et interactive guidée par l'utilisateur. Elle permet à l'utilisateur de se concentrer sur les règles intéressantes sans qu'il soit nécessaire d'extraire toutes les règles et sans une limite pour le support minimum. De cette faon, l'utilisateur peut explorer l'espace de règles progressivement, une petite quantité à chaque étape, à partir de ses propres attentes et des règles découvertes liées à ces dernières.

(v) L'étude expérimentale analysant l'efficacité de l'approche et la qualité des règles découvertes. À cette fin, nous avons utilisé une grande base de questionnaires réelle concernant la satisfaction des clients. Pour AR*IPSO*, l'étude a été réalisée en coopération complète avec l'expert de domaine. A partir d'un set de près de 400 milliers de règles d'association, AR*IPSO* a filtré, selon différents scénarios, entre 3 et 200 règles validées par l'expert. En toute évidence, AR*IPSO* permet à l'utilisateur de réduire de manière significative et efficace le set de règles. Pour AR*LIUS*, nous avons expérimenté différents scénarios sur la base de données et nous avons obtenu des ensembles réduits de règles (moins de 100) avec un support très faible.

Mots-clefs: Extraction de Connaissances à partir de Données, Règles d'Association, Mesures d'Intérêt, Ingénierie de Connaissances, Ontologies, Web Sémantique

Acknowledgments

The following dissertation, while an individual work, benefited from the insights and direction of several people.

This thesis would not have been possible unless it was financially supported by *Pays de la Loire Region* of France; *UNIVALOIRE Association* was in charge with the administration of my financial contract. Thus, I would like to thank the council of the Pays de la Loire Region for giving me the possibility to follow my dreams...

I owe my deepest gratitude to Mr. Fabrice Guillet and Mr. Henri Briand, my PhD supervisors.

Mr. Fabrice Guillet believed in me, guided me and gave me precious advices throughout this work and not only. Without him, I wouldn't be here; he gave me the possibility to get this far by always encouraging me to go further. His research and teaching qualities represent a model for me.

During two years that Mr. Henri Briand co-supervised my PhD, he provided timely and instructive comments and evaluation of my work allowing me to progress in my research. Our discussions were both constructive and enlightening, and I thank him.

I wish to express my gratitude to Mr. Fabien Gandon and Mr. Jean-Gabriel Ganascia for the honor that they made me by accepting to review my thesis and for all their constructive remarks that allowed me to improve my dissertation.

I would like also to thank Ms. Marie-Aude Aufaure for making me the honor to accept being examinator and jury president.

I had the pleasure to work in the *COnnaissances et Décisions - KnOwledge and Decisions (KOD)* research team of Nantes-Atlantique Computer Science Laboratory (LINA UMR 6241), in the Computer Science Department of Ecole polytechnique of University of Nantes. I am grateful to Ms. Pascale Kuntz for giving me the great privilege of joining the research team that she pilots. She was always there bringing me priceless answers and advices.

I would like to thank Andrei Olaru, Loïc Glimois and, more specially, Ms. Christelle Le Bouter for their implication in this work. I am grateful to Ms. Catherine Galais for reading my english papers.

My colleagues were always sources of laughter, joy, and support. They made my days less harder than they seemed to be.

In all of the ups and downs that came my way during the PhD years, I knew that I had the support of my friends, be they near or far. I will not name you, because you will know that it's all about you...Nevertheless, I would like to thank one special person – he was always there, listening and encouraging me, and always understanding; thank you.

Without you, my family, I would be nothing. Va iubesc.

Contents

List of Figures

List of Tables

1 Introduction

Looking over the implication rules generated on census data was educational. First, it was educational because most of the rules themselves were not. The rules that came out at the top, were things that were obvious

Dynamic itemset counting and implication rules for market basket data
Brin et *al.*

Contents

1.1 Research Context

Knowledge Discovery in Databases: Association Rule Mining Technique

Since the 1960s, with the apparition of computer science and databases, information volumes stored in databases have been growing exponentially. Nowadays, we are living in a world mainly defined by connections between people, devices and machines managed generally through the Internet. In this context, important amounts of data are generated in different domains such as supermarket transaction data, credit card usage records, telephone call details, government statistics and medical records [99].

Three decades ago, this specific context motivated and encouraged the development of a new research field, called *Knowledge Discovery in Databases (KDD)*, which was defined by Frawley *et al.* 1992 [64, 70] as the *non-trivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data.* The main objective is to provide new information, starting from a set of information/data, which will prove useful for the user in taking decisions/actions. The KDD process is composed of different steps which can be summarized in three main phases: first, the pre-processing phase cleans and prepares the database; second, the data mining phase consists in applying mining techniques over the database in order to discover new patterns; last, the post-mining phase deals with evaluation and visualization techniques which help the user to validate the discovered patterns.

Association rule mining [5] is one of the most important techniques of data mining. It aims to discover regularities in databases under the form of implications *$if\ X\ then\ Y$*, denoted as $X \to Y$, where X and Y are named the antecedent and the consequent, respectively, and they are defined by conjunctions of database items. This implication can be read as follows: if the items in X exist in a transaction, then, it is probable that the items in Y also exist in the same transaction; in addition, an association rule is defined by two metrics – support and confidence.

The association rule mining technique was first applied in the case of *Wal-Mart*[1] supermarket basket problem in order to study purchase trends of supermarket customers. One of the main discoveries was that *if a male customer buys diapers on a Friday afternoon, then he will be interested in buying beers.* As a result, by moving beers next to diapers, Wal-Mart noticed an important increase in beer sales. This discovery clearly explains the interest of association rule mining technique – the discovery of new surprising and useful knowledge.

But, why is this technique so popular in the data mining research field? At least two reasons can be stated. First, the model of the extracted patterns is very simple and comprehensible for a data mining non-specialist user. The reason is that implications are the core of human thinking. Second, the technique extracts the complete set of association rules without the need of an important user implication during the process.

Nevertheless, the usefulness of association rule technique is strongly limited by the huge amount and the low quality of delivered rules; thousands of rules are extracted by classical techniques from a database of several dozens of attributes and several

[1]http://www.walmart.com/

hundreds of transactions. To reduce the rule volume, one possibility is to increase the support threshold. But, as suggested by Silbershatz and Tuzilin 1996 [184], nuggets are often represented by those rare - low support - and unexpected association rules which are surprising to the user. So, the more we increase the support threshold, the more efficient the algorithms are and the more the discovered rules are obvious and, hence, the less they are interesting for the user.

As a result, it is necessary to bring the support threshold low enough in order to extract valuable information. Unfortunately, the lower the support is, the larger the volume of rules becomes, making it intractable for a decision maker to analyze the mining result. Experiments show that rules become almost impossible to use when their number exceeds 100. Thus, it is crucial to help the decision maker with an efficient technique for reducing the number of rules.

To overcome this drawback, several methods have been proposed in the literature such as itemset concise representations, redundancy reduction, filtering, ranking and post-processing. However, most of the existing methods are generally based on statistical information in the database. Since rule interestingness strongly depends on the user knowledge and goals, these methods do not guarantee that interesting rules will be extracted. For instance, if the user looks for unexpected rules, all the already known rules should be pruned. Or, if the user wants to focus on a specific family of rules, only this subset of rules should be selected.

In this context, several approaches propose to integrate the user knowledge in association rule mining process in order to reduce the rule number [122, 136, 154, 183]. The main limitations of these techniques can be summarized as:

- first, the models for user knowledge representation are limited and they do not permit to define user knowledge with accuracy and flexibility. The more the knowledge is represented in a flexible, expressive and accurate formalism, the more efficient the rule selection is;
- second, these approaches do not address the problem of interactivity with the user in depth. Interactivity would allow a more iterative mining process where the user can successively test different hypotheses or preferences, and focus on interesting rules, as stated by Blanchard *et al.* 2003 [28].

Knowledge Representation: Ontologies and Semantic Web

Knowledge Engineering (KE) research field attempts at describing a set of techniques and concepts which aim to acquire, represent and exploit domain knowledge in order to be integrated in *Knowledge-Based Systems (KBS)*. KBSs are described as systems based on knowledge and reasoning mechanisms, proposing solutions to real-world problems. In this context, *Knowledge Representation (KR)* is the area of Artificial Intelligence (AI) field which aims to propose adapted formalisms for domain knowledge representation with important facilities for knowledge reusing, sharing and inference [34].

In the past fifteen years, we saw an increase in the development of ontologies such as knowledge representation formalism. In computer science, they are used in order to gather the information (knowledge) concerning a given domain. A first notable definition was proposed by Gruber 1993 [89, 90] who defines an ontology as *a formal, explicit specification of a shared conceptualization.* This definition gathers the most important characteristics of an ontology: (1) it represents the concepts of a domain; (2) it should be described in a well-defined language so that machines should be able to interpret it; (3) it should explicitly define its elements; (4) it collects some knowledge which is common to a community.

In the meantime, information over the Web grew more and more making the Web search nearly impossible. The *Semantic Web* was introduced by Tim Berners-Lee 2001 [25] and it proposes to extend the traditional Web with well-structured annotations in order to develop a content that can be understood at the same time by humans and by machines. In the past decade, W3C aimed to propose formalisms for meta-data representation over the Web. The simpler language, XML [212], was introduced for data publishing and transmission via networks. Further, RDF [123] is the first semantic-oriented language that describes resources over the Web. Nevertheless, it has important limits such as the lack of axioms and of reasoning. In this context, OWL was introduced as the most complex representation language for ontologies taking as basis the RDF language and the class- and property-structuring capabilities of RDFS, but improving this formalism from different points of view.

The main advantage of OWL language is that it is based on Description Logics. Thus, it brings important improvements to RDF like: describing classes as logical combinations (intersection, unions, etc), describing restrictions on how properties behave locally on a specific class – it allows to define classes where particular properties are restricted –, and allowing a reasoning process. The inference process has multiple interests: first, it permits to test whether an ontology is consistent or not (i.e. two concepts are equivalent), second, it reconstructs the taxonomy of the concepts following the subsumption relation, and last, it checks whether an individual is a member of a concept.

1.2 Contributions

In order to fulfill the drawbacks in the association rule mining, this work addresses two issues: the integration of user knowledge in the discovery process and the interactivity with the user. The first issue requires defining an adapted formalism to express user knowledge with accuracy and flexibility. In this context, this work tries to benefit from the research carried out in the Semantic Web field, and more precisely from the representation languages developed in order to be used as user knowledge representation in the discovery process. Second, the interactivity with the user allows a more iterative mining process where the user can successively test different hypotheses or preferences and focus on interesting rules.

The contributions of this thesis can be summarized as follows:

1. **A model to represent user knowledge**

   We define a model to represent user knowledge in the discovery process. It is composed of three formalisms of knowledge:

   (a) **Ontologies** permit the user to express his/her domain knowledge by means of a high semantic model;
   (b) **Rule Schema** is a new rule-like formalism that we propose which allows the user to define his/her expectations regarding the discovered rules through ontology concepts;
   (c) A set of **Operators** for interactive processing can be applied over Rule Schemas. They help the user to take several actions like pruning and selection of conform or unexpected rules from the set. The user can choose an Operator to be applied over each Rule Schema. For instance, if the user chooses the Pruning Operator to be applied over a Rule Schema, all the rules conforming to the Rule Schema are eliminated.

2. **A new post-processing approach**

   We propose a new approach, called AR*IPSO* (Association Rule Interactive Post-processing using rule Schemas and Ontologies), which helps the user to reduce the volume of the discovered rules and to improve their quality. It consists in an interactive process integrating user knowledge and expectations by means of the proposed model, and different interestingness measures that assess the quality of the discovered rules. At each step of AR*IPSO*, the interactive loop allows the user to change the provided information and to reiterate the post-processing phase which produces new results.

3. **Implementation of AR*IPSO* tool**

   We implemented the proposed approach in post-processing; for this purpose, we developed a tool which is complete and operational, and which implements all the functionalities described in the approach. More particularly, it proposes a visualization interface which helps the user in editing the knowledge and validating the discovered rules. From a technical point of view, the implementation makes the connection between different elements like the set of rules and rule schemas stored in PMML/XML files, and the ontologies stored in OWL files and inferred by the Pellet reasoner.

4. **An adapted implementation without post-processing**

   Called AR*LIUS* (Association Rule Local Interactive mining Using rule Schemas), the new implementation consists in an interactive local mining process guided by the user. It allows the user to focus on interesting rules without the necessity to extract all of them and without minimum support limit. In this way, the user may explore the rule space incrementally, a small amount at each step, starting from his/her own expectations and discovering their related rules.

5. **Experimental studies**

   The experimental study analyzes the approach efficiency and the discovered rule quality. For this purpose, we used a real-life and large database questionnaire concerning customer satisfaction. For AR*IPSO*, the experimentation was carried out in complete cooperation with the domain expert. From an input set of nearly 400 thousand association rules, for different scenarios AR*IPSO* filtered between 3 and 200 rules validated by the expert. Clearly, AR*IPSO* allows the user to significantly and efficiently reduce the input rule set. For AR*LIUS*, we experimented different scenarios over the same questionnaire database and we obtained reduced sets of rules (less than 100) with very low support.

1.3 Thesis Organization

This thesis is organized as follows:

Chapter 2 is concerned with the Knowledge Discovery in Databases process, and more precisely the Association Rule Mining technique. It provides formal definitions and considers the improvements done in the frequent itemset generation and rule discovery phases in more detail.

Chapter 3 introduces the Knowledge Engineering and the Knowledge Representation domains, and gives a brief overview of the ontology notion. More specifically, it details the interest of the Semantic Web field, and it presents the languages proposed by W3C for knowledge representation.

Chapter 4 describes interestingness measures over association rules, with a particular interest in subjective measures. It provides detailed presentations of most important approaches proposed in the literature.

Chapter 5 is dedicated to the post-processing AR*IPSO* approach; in detail, it describes the model for knowledge representation along with the interactivity with the user. Further, it presents an adaptation of AR*IPSO* in local mining (without post-processing), called AR*LIUS*.

Chapter 6 provides the architecture of the AR*IPSO* tool that represents the implementation of the method. Also, it details the technical choices that were made.

Chapter 7 proposes experimental studies with AR*IPSO* and AR*LIUS* in order to analyze the approach efficiency and the discovered rule quality. For this purpose, we used a real-life questionnaire database, and the experimentation was carried out in complete cooperation with the domain expert.

Chapter 8 draws our conclusions and perspectives. One of the most important perspectives is to develop an Oracle-based implementation of AR*IPSO*.

2
Data Mining and Association Rules

Computers have promised us a fountain of wisdom but delivered a flood of data.

Knowledge Discovery in Databased
Frawley et *al.*

Contents

2.1 Introduction

Knowledge Discovery in Databases deals with the process of extracting interesting patterns starting from data. It consists of different steps starting with Data Cleaning and Data Pre-Processing till Data Mining and Post-Mining of Patterns. In Data Mining, different mining techniques can be applied among which association rule mining is one of the most popular.

Association rule mining proposes the discovery of knowledge in the form of implication *IF Antecedent THEN Consequent* that we note $Antecedent \rightarrow Consequent$. In an association rule, the antecedent and the consequent are conjunctions of attributes in a database. More particularly, an association rule $Antecedent \rightarrow Consequent$ expresses the implicative tendency (validated by statistical metrics) between the two conjunctions of attributes – from the antecedent toward the consequent.

The main advantage of association rules mining technique is that it extracts comprehensible knowledge which can be manipulated by unscientific users. Moreover, it premits the extraction of complete sets of association rules for a given scenario. Unfortunately, this important advantage represents also its main limit. On the one hand, classical algorithms require important resources and time to produce the final result. On the other hand, even when the rules can be generated, their number increases once with the number of attributes (and/or transactions) in the database, and it often overpasses hundreds of millions. In this context, it is impossible for a user to manipulate the set of discovered rules.

In order to fulfill these drawbacks, several techniques were proposed in the literature. First, for the rapidity problem, different algorithms were developed with the aim to reduce the execution time and the resources that are used. Second, to increase the efficiency of the rule generation process – to reduce the number of discovered rules – different methods were proposed with the aim to filter the rules. In this chapter we will study two of them – constraint-based association rule mining and redundancy rule reduction techniques –, and in Chapter 4 we will discuss interestingness measures.

The discovery of association rules using constraint-based techniques represents a first method for the reduction of the number of rules. Constraints are here provided by the user in different formalisms – user knowledge, data, dimensional, interestingness or rule-based. They permit to determine the content of extracted rules and to reduce the rule volume. The main characteristic of constraint-based association rule mining techniques is that the constraints are generally integrated in the mining algorithm permitting an execution time reduction also.

A second technique to reduce the rule volume consists of pruning the redundant rules from the entire set of discovered rules. A redundant rule is informally defined as a rule which does not bring important information to the complete set of rules, and thus, pruning it does not decrease the quality of the rules set. Further, we can consider that a rules is redundant regarding one or several rules.

This chapter starts with a brief presentation of Knowledge Discovery in Databases Process and, afterwords it goes deeply in the association rule mining technique – definitions and notations are presented. The second part of this chapter is dedicated to the improvements made in generating frequent itemsets and/or association rules.

Further, constraint-based association rule mining techniques and the reduction of rule redundancy are studied.

2.2 Knowledge Discovery in Databases Process

Knowledge Discovery in Databases process (KDD) was defined by Frawley *et al.* 1992 [70], and revised by Fayyad *et al.* 1996 [64], as the *non-trivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data.*

This definition regroups the main elements of the process. First, a *data* is defined as a set of facts, F, describing the application domain. Next, a *pattern* is formalized as an expression E over a language L defining facts (relationships) among a subset $F_E \subseteq F$, such as E is simpler than the enumeration of all facts in F_E.

The KDD *process* is a multi-step task with the aim to *extract patterns from data.* It is characterized as being a *non-trivial* process because it can decide which actions to perform and whether the results are interesting enough. This defines the degree of pre-processing and evaluation autonomy.

Four notions characterize the extracted patterns: validity, novelty, usefulness and comprehension by users. First, the extracted patterns should be *valid* on new data with some degree of certainty described by a set of metrics. Second, the *novelty* of patterns can be measured with respect to previous or expected values, or knowledge. Next, the patterns should be *useful* to the user, i.e. they can help the user to take benefic actions. The utility can be measured using dedicated metrics. Last, the extracted patterns should be *comprehensible* to humans in order to be able to use them to take actions.

The notion of *interestingness* is employed in order to describe the interest of a pattern. This notion is defined as a general measure over the 4 features described above: the validity, the novelty, the utility and the comprehensibility. Thus, a pattern which is interesting, i.e. it has the interestingness measures greater than a certain threshold, is called *knowledge.*

The four characteristics reveal a direct user implication in the discovery process. The interactivity in the KDD process outlines the importance of the user decisions along the KDD process. Figure 2.1 presents the main steps of the KDD process: Data Cleaning and Data Integration, Data Pre-Processing, Data Mining, and Post-Processing.

KDD is a multi-disciplinary field, being integrated in areas such as artificial intelligence, machine learning, neural networks, data base technology, information retrieval and data visualization. Furthermore, the KDD process is applied overs different research domains. In 1990s, there were only a few examples of discovery in real data. Nowadays, more and more domains benefit from the utilization of KDD techniques, such as medicine, finance, agriculture, social, marketing, military, and many others.

2.2.1 Data Cleaning and Data Integration

This step consists of two pre-mining tasks: *Data Cleaning* used in order to remove noise and inconsistent data, and *Data Integration* focused on bringing data from

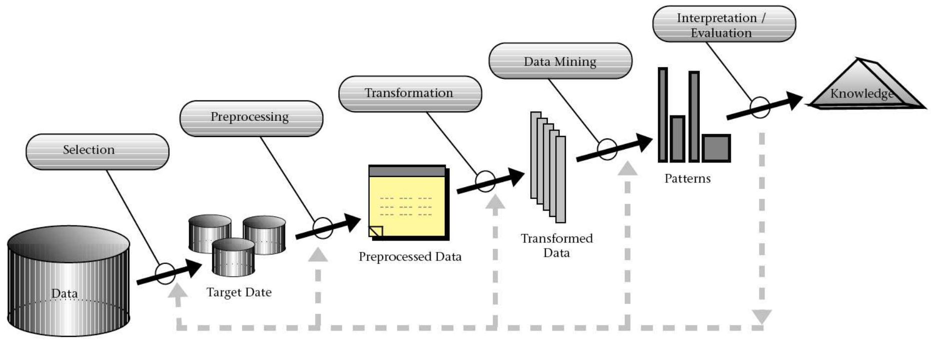


FIGURE 2.1: An overview over the steps of the KDD process [64].

multiple sources and represented in different formats.

In most of cases, real-life data contains noise and missing values, so they are considered inconsistent. Applying the knowledge discovery process over this data may extract unreliable knowledge. The *Data Cleaning* step consists in correcting inconsistencies in data appeared during the insertion phase. If it is possible, some types of inconsistencies are corrected manually by the user, but in lot of cases, automatic systems are needed in order to check the range of the attribute values.

For example, if values are missing for some attributes, this step tries to compute them by using some heuristics or to insert them manually, always keeping the same statistical information over data.

Another example is the case when some values are inserted into the data by error. In this case, a set of methods can be applied in order to determine which values are not in the range (i.e. clustering).

Data Integration step assesses the process of combining data from different sources (databases, external sources, etc). The problem consists in the incoherency of the resulting database. A valuable example for this step is the redundancy: if an attribute A can be determined from another attribute B, we can say that A is redundant comparing to B. Another type of redundancy is the existence of two attributes with different names and from different sources, but which have the same meaning. The Data Integration step proposes solutions for this type of problems.

In conclusion, the goal of this two pre-mining tasks is to generate *datawarehouses* and/or *datamarts* containing modified data making easier the futures analysis processes.

2.2.2 Data Pre-Processing

Data Pre-Processing step deals with two important problems. The first one is the verification if the datawarehouse was well developed during the data cleaning and data integration phase; if needed, data are re-cleaned. The second goal of this step is to transform (or to reduce) the data in order to be able to apply a knowledge discovery technique. Once the datawarehouse is well-developed, this step make possible the improvement the results proposed by the data mining phase.

2.2.3 Data Mining

Data Mining step is essential in the KDD process. Using intelligent methods applied over data, it extracts interesting patterns. Several techniques were developed in the literature in order to extract interesting patterns:

- *Predictive modeling - supervised classification or regression.* The classification builds a model in order to predict the class of an object whose class label is unknown. The classification is composed of two phases. The first one is represented by a learning phase - description of a set of classification rules called learning model. The second phase is the classification one - testing data are used in order to verify the precision of the classification rules generated during the first phase. The main classification techniques are: *Decision Trees*, *Bayesian Classification*, *Neural Networks.*
- *Descriptive modeling - clustering.* The clustering technique partitions the data into classes so that the intraclass similarity be maximized and the interclass similarity be minimized. In a first step, all the adequate classes are discovered, and then the data are classified according to the discovered classes. We can note that, comparing to classification, the classes are not known from the beginning, they are discovered using a set of observations. Different methods of clustering were developed among which we remind *K-means.*
- *Discovering patterns and association rules.* This technique proposes to find regularities in data, revealed to be useful in many domains, as business, medicine, etc. Finding association rules consists in finding frequent implications in data of the type *IF X THEN Y*; X and Y represent the antecedent and, respectively, the consequent, and they are propositions constructed using attributes from a database. The main advantage of this model is that association rules are easily exploitable and comprehensible by humans.

2.2.4 Post-Mining of Discovered Patterns

Usually called *post-processing* [15] or *post-mining*, this phase is the last one of the KDD process. It is composed of three important steps which make it possible for a discovered *pattern* to be considered as a *knowledge*: the validation, the evaluation and the visualization (a complete state of art of visualization is presented by Ben Said *et al.* 2010 [23]).

In a first step, after proposing for evaluation the results of a certain model, it is important that the user validates the proposed model. The validation should be made from the point of view of the comprehensibility of the extracted patterns and their possible utility.

The mining process discovers a list of patterns with a given level of interestingness. In the evaluation step, the user is able to evaluate the patterns, i.e. to determine the importance of the extracted patterns, using several user-driven methods or statistical database oriented methods. Another method of pattern evaluation is the visualization, which is related to the model of extracted patterns.

2.3 Association Rule Mining Technique

This section will outline the main motivations of association rule mining technique and it will formally describe the main notions as they are at the very foundation of this thesis. Then, the most important algorithms are detailed and the most important improvements that were proposed in the literature.

2.3.1 Motivations

The idea of discovering relationships in data started with the well-known problem of the supermarket basket discussed for the first time by Agrawal *et al.* 1993 [5]. At that moment, the supermarkets found themselves with a huge stock of shopping tickets and using KDD techniques could help them to improve the supermarket layout and, thus, to increase their sales. But, how could a set of shopping tickets produce some modifications in the supermarket layout?

In a first analysis of shopping basket data *frequent itemsets* were extracted, for example {*nappies*, *beer*} which expresses that *nappies* and *beer* appear together in a great part of shopping baskets.

Later, *association rule technique* proposes the extraction of implications, and a valuable information was discovered, that *nappies* → *beer*, which specifies that if a man customer buys *nappies*, it is very probable that he will buy *beer* too. Thus, placing *beer* products next to *nappies* products the sales of *beer* increased considerably.

2.3.2 Problem Definition

In general, association rule mining technique is applied over a database $D = \{I,\ T\}$. Let us consider $I = \{i_1, i_2, \ldots, i_m\}$ a set of m binary attributes, called *items*. Let $T = \{t_1, t_2, \ldots, t_n\}$ be a set of n transactions, where each transaction t_i represents a binary vector, with $t_i[k] = 1$ if t_i contains the item i_k, and $t_i[k] = 0$ otherwise. A unique identifier is associated to each transaction, called *TID*. Let X be a set of items in I. A transaction t_i *satisfies* X if all the items of X exist also in t_i, formally, we can say that $\forall i_k \in X,\ t_i[k] = 1$. In conclusion, a transaction t_i can be viewed as a subset of I, $t_i \subseteq I$.

Definition 2.3.1
An itemset $X = \{i_1, i_2, \ldots, i_k\}$ is a set of items $X \subseteq I$. We can denote the itemset X by $i_1, i_2, \ldots, i_k$, the comma being used as a conjunction, but most commonly it is denoted by $i_1 i_2 \ldots i_k$, omitting the commas. ■

Example 2.3.2 The set $X =$ *Milk Pear Apple* (or $X =$ *Milk, Pear, Apple*) is an itemset composed by three items: *Milk*, *Pear* and *Apple*. ■

Definition 2.3.3

An itemset X is a k-itemset if X is an itemset $X \subseteq I$ and if it contains k items: $|X| = k$. ■

Example 2.3.4 The itemset *Milk*, *Pear*, *Apple* is a *3*-itemset. ■

Definition 2.3.5

Let $X \subseteq I$ and $t_i \in T$. We say that the set of all transactions which contain the itemset X is defined by:

$$t : \mathcal{P}(I) \to T,\ t(X) = \{t_j \in T \mid X \subseteq t_j\}.$$

Similarly, the set of items contained by all the transactions in $T' \subseteq T$ is described by:

$$i : T \to \mathcal{P}(I),\ i(T') = \{i_j \in I \mid \forall t_k \in T',\ i_j \in t\}.\ ■$$

Definition 2.3.6

The support of an itemset X is defined as the number of transactions in T that support X. Thus, the fraction of transactions in T that satisfy the itemset X represents the support of X. This is denoted by:

$$supp(X) = \frac{|\{t \in T | X \subseteq t\}|}{|t \in T|}.\ ■$$

Definition 2.3.7

Un itemset X is defined as a frequent itemset if the support of X is greater than a given threshold minsupp: $supp(X) = |t(X)| \geq minsupp$. ■

The model of association rules was for the first time introduced by Agrawal *et al.* 1993 [5], and revised, one year later, by Agrawal and Srikant 1994 [6].

Definition 2.3.8

In a first attempt, an association rule was defined as an implication of the form $X => i_j$, where X is an itemset $X \subseteq I$ and i_j is an item $i_j \in I$ with $\{i_j\} \cap X = \emptyset$ [5].

Later, the definition was extended to an implication of the form $X => Y$, where X and Y are itemsets and $X \cap Y = \emptyset$ [6]. The former, X, is called the antecedent of the rule, and the latter, Y, is called the consequent of the rule.

A rule $X \to Y$ is described by two important statistical factors:

- *The support of the rule which is defined as the support of the itemset created by the union of the antecedent and the consequent of the rule*

$$supp(X \to Y) = supp(X \cup Y) = |t(X \cup Y)|.$$

It presents the ratio of the number of transactions containing $X \cup Y$. If $supp(X \to Y) = s$, $s\%$ of transactions contain the itemset $X \cup Y$.

- *The confidence of an association rule is defined as the probability that a transaction contains Y knowing X. In other words, the confidence is the ratio (c%) of the number of transactions that, containing X, contain also Y:*

$$confidence(X \to Y) = \frac{supp(X \to Y)}{supp(X)} = \frac{supp(X \cup Y)}{supp(X)}. \blacksquare$$

Definition 2.3.9

If we provide the minimum thresholds provided by the user for the support, and respectively for the confidence as minsup and $minconf$, an association rule $X \to Y$ is valid if:

- *the support of the rule is greater than minsupp: $supp(X \to Y) \geq minsup$;*
- *the confidence of the rule is greater than $minconf$: $conf(X \to Y) \geq minconf$.* ■

Definition 2.3.10

X is denoted as a maximal itemset if the itemset X is frequent and no supersets of X are frequent. ■

Definition 2.3.11

Let us consider two association rules R_1 and R_2. The rule R_1 is more general than the rule R_2, denoted $R_1 \preceq R_2$, if R_2 can be generated by adding additional items to either the antecedent or consequent of R_1. Then, a rule R_j is denoted as redundant by Zaki 2004 [220] if there exists some rule R_i, such that $R_i \preceq R_j$. Thus, the non-redundant rules in a collection of rules are those rules that are most general, i.e. those having minimal antecedents and consequents. ■

Definition 2.3.12

A rule set is defined as optimal by Li 2006 [130] with respect to an interestingness metric if it contains all rules except those with no greater interestingness than one of its more general rules. An optimal rule set is a subset of a non-redundant rule set. ■

2.3.3 CHARADE System

Long before the introduction of the association rule notion, Ganascia 1987 [73] proposed a system, named *CHARADE*, which aims to learn automatically a rule system. *CHARADE* is applied over a training set of examples described by means of conjunctions of descriptors; in our notation we can consider that examples correspond to transaction, and descriptors to items. In this context, the author argues that, if a set of examples has a conjunction of descriptors D and it has also the descriptor d, then there is possible to have an implication $D \to d$.

For this purpose, two covering functions are suggested – they describe the set of examples that covers a conjunction of descriptor (TR), or vice versa (SUB).

Example 2.3.13 Let us consider the following set of examples:

$E_1 : milk, pork, pear, apple$
$E_2 : milk, pork, grape, pear$
$E_3 : pork, pear$
$E_4 : milk, pork, pear.$

If we consider *milk* and *pork* descriptors, using TR and SUB functions we are able to find the descriptors associated to *milk* and *pork* through their common examples, as follows:

$$\begin{aligned} &SUB \circ TR(milk, pork) \\ &\qquad = SUB(E_1, E_2, E_4) \\ &\qquad = \{milk, pork, pear\}. \end{aligned}$$

In this context, we can say that *pear* can be associated with *milk* and *pork* descriptors, and, moreover, to generate the following rule:

$milk, pork \rightarrow pear.$ ■

The system proposes an exploration of the descriptor space from the more general to the more specific, and it uses a measure in order to assesses the *useless* of a set of descriptors which permits to reduce the exploring – in the case of a useless set of descriptors, the system does not access the more specific sets of descriptors.

2.3.4 *Apriori* – Classical Association Rule Mining

Generally, the problem of discovering association rules from database is composed of two main subproblems [5]:

- The first one consists in *generating all frequent itemsets*;
- Starting from frequent itemsets produced above, the second problem deals with the *generation of the association rules* having the confidence greater than *min-conf.*

For the first problem which is generally known as *frequent itemset generation* a first algorithm, named *Apriori*, was designed by Agrawal and Srikant 1994 [6] and it has its bases on the CHARADE System. *Apriori* algorithm is presented in the Table 2.1.

The algorithm gradually generates the set of itemsets from *1-itemsets* to *2-itemsets* to . . . etc. In the first pass over the data (line 1 in the algorithm), supports for the *1-itemsets* are computed in order to select only the frequent ones.

In the next steps (lines 2 to 10), starting from the *(k-1)-itemsets* and using the downward closure property *k-itemsets* are generated. Thus, starting from the frequent *(k-1)-itemsets* already generated in the previous step, the function *apriori-gen* (line

TABLE 2.1: The frequent itemset generation in *Apriori* algorithm [6].

```
Input: Database $D$
Output: The set $L$ of itemsets

1. $L_1 = \{$1-itemsets$\}$
2. forall $(k = 2; L_{k-1} \neq \emptyset; k++)$ do begin
3.     $C_k$ = apriori-gen$(L_{k-1})$
4.     forall transactions $t \in D$ do begin
5.        $C_t = subset(C_k, t)$
6.        forall candidates $c \in C_t$ do
7.           $c.count$++
8.     endfor
9.     $L_k = \{c \in C_k \mid c.count \geq minsup\}$
10. endfor

  apriori-gen$(L_{k-1})$
12. forall itemsets $c \in C_k$ do begin
13.     forall (k-1)-subsets $s$ of $c$ do begin
14.        if $(s \in L_{k-1})$ then
15.           delete $c$ from $C_k$
16.     endfor
17. endfor
```

3) generates new potentially frequent *k-itemsets*, called *candidates*. The candidates are validated during a new pass over data when the support of each candidate is computed (lines 4 to 8). The particularity of the algorithm comes from the support counting method; in fact, the function *subset* (line 5) receives the set of candidates and a transaction t of the database and returns the set of candidates satisfying the transaction. In line 7 the support of each candidate is increased. In line 9, the frequent *k-itemsets* are selected and they become the entry for the next step of the algorithm. The algorithm ends when no frequent itemset is generated.

The *Apriori* algorithm is based on a bottom-up, breadth-first search method which enumerates every single frequent itemset. Moreover, the algorithm uses the *downward closure property*. The latter is defined as the property that every frequent itemset is composed of frequent subsets (sub-itemsets). For the frequent itemset generation step the algorithm uses lattices[1].

Example 2.3.14 Let us consider a sample of the supermarket transaction database

[1]A lattice is an ordered set with several specific properties. A detailed description will be given in section 2.4.3

(Table 2.2) and a minimum support threshold of 50%.

TABLE 2.2: Supermarket database sample for the *Apriori* algorithm example.

Tuple	Transaction
1	milk, pork, pear
2	milk, pork, apple
3	pork, pear
4	milk, pork, pear

In Figure 2.2, we present the process of generating frequent itemsets by using the *Apriori* algorithm. The algorithm starts with an empty list of candidates, and, during the first pass, all *1*-itemsets are generated, but only those satisfying the support constraint (50%) become candidates. Therefore, the itemset {*apple*} with the support of 25% does not satisfy the support constraint, and, thus it is not frequent and it is not considered in the following as a candidate.

In the next passes, the *k*-itemsets which include no frequent *(k-1)*-itemsets are denoted as not frequent; thus, they are not computed in order to reduce time execution. For instance, {*pork, apple*} itemset is not generated.

In this figure the itemsets not containing {*apple*} itemset are potentially frequent. For instance, the itemset {*milk, pork, apple*} is not frequent because {*milk, apple*} and {*pork, apple*} itemsets are not frequent. On the contrary, the {*milk, pork, pear*} itemset is frequent because the three *2*-itemsets composing it are frequent ({*milk, pork*}, {*pork, pear*} and {*milk, pear*}), and its support is 50%.

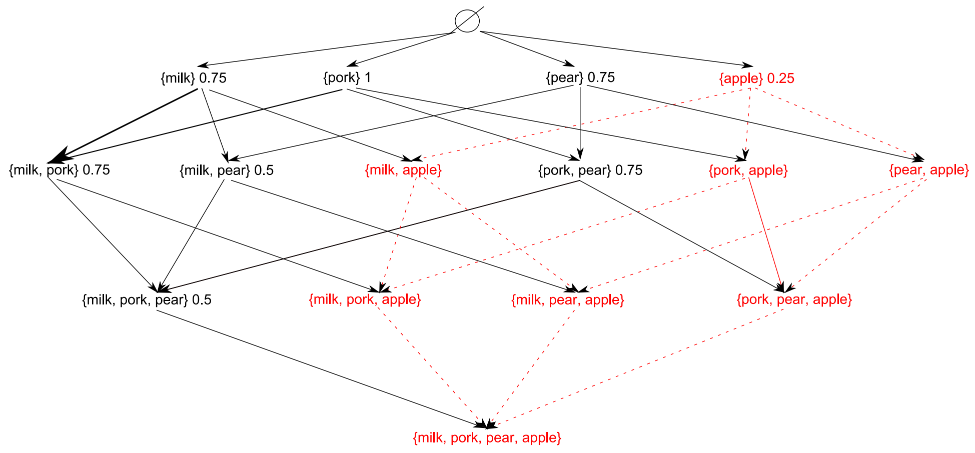


FIGURE 2.2: An example tree of the frequent itemset generation.

The second problem in the association rule mining technique is the *generation of rules*; the objective is to create association rules from the frequent itemsets generated above. The algorithm for rule generation integrated in *Apriori* is presented in Table 2.3.

TABLE 2.3: Rule generation step in *Apriori* algorithm [6].

```
Input:  Set of itemsets $l$
Output:  Set of association rules $Rules$

1. forall itemsets $l_k$, $k \geq 2$ do
2.     call $genrules(l_k, l_k)$;

3. procedure $genrules(l_k$: $k$-itemset, $a_m$: $m$-itemset)
4. $A = \{(m-1)$-itemsets $a_{m-1} \mid a_{m-1} \subset a_m \}$
5. forall $a_{m-1} \in A$ do begin
6.     $conf = support(l_k)/support(a_{m-1})$
7.     if $(conf \geq minconf)$ then
8.         $R = a_{m-1} \Rightarrow (l_k - a_{m-1})$
9.         if $(m - 1 > 1)$ then
10.            call $genrules(l_k, a_{m-1})$
11.            $Rules = Rules \cup R$
12. return $Rules$
```

The method of rule extraction is very simple. Let us consider the set L of frequent itemsets. Considering $l_i \in L$, the method finds all subsets a of l_i, $a \subseteq l_i$, and proposes a set of rule candidates of the form $a \to (l_i - a)$ which are tested against the confidence measure.

In lines (1-2) the recursive procedure *genrules* is called for each set of k-itemsets. *genrule* generates recursively the sub-itemsets level by level (line 4) in order to produce the rules which are further tested against the confidence.

Example 2.3.15 Let us consider the itemset $l_1 = \{$*milk, pork, pear*$\}$ [S = 50%]. One of the rule that we could generate is *R: milk, pork → pear.* To compute the confidence of this rule we use only the support of the complete itemset and the support of the $l_2 = \{$*milk, pork*$\}$ itemset:

$$conf(R) = \frac{supp(l_1)}{supp(l_2)} = \frac{0.5}{0.75} = 0.67. \blacksquare$$

In this section, we saw that the *Apriori* algorithm is able to extract a set of association rules starting from a database. In this context, two problems concerning the quality of the *Apriori* algorithm emerge: *the rapidity* and *the efficiency.*

The first one, *the rapidity*, deals with the capacity of the algorithm to generate the expected results in a reasonable execution time without using important quantities of resources. The generation of frequent itemsets is an exponential problem, because the search space to enumerate all the frequent itemsets is 2^m, where m is the number of items. Furthermore, the algorithm makes several passes over data depending on the length of the generated itemsets. These tasks imply an exponential growth of the resources employed during the rule mining process and an important increase of the execution time.

Rule generation is an exponential problem depending on the number of items in the frequent itemsets; for example, for a *k-itemset*, the number of possible association rules is $2^k - 2$. In this context, we can note that the execution time is not very high. First, because there is no need for new passes over the database in this step. And second, because for the cases of a large database, the execution time of this step becomes less important than that of the frequent itemsets generation step.

Consequently, the rapidity is equivalent to the rapidity of the process of frequent itemset generation. Thus, to improve this characteristic of the rule mining process, it is imperative to improve the frequent itemset generation step. For this purpose, different propositions were made in the literature, presented in Section 2.4.

Second, *the efficiency* deals with the capacity of the algorithm to produce the researched results. The main drawback in the association rule mining is that classical techniques produce a hight number of rules which are quasi unusable by the user because millions of association rules can be extracted from large databases with a reduced threshold of support. In consequence, in the last decade, an intense research was done on reducing the number of extracted rules. Unfortunately, deciding whether a set of association rules is useful is both an objective and a subjective problem. The objective one is data-oriented, and the subjective one is user-oriented – a set of rules might correspond to a user and might not correspond to another one. For this purpose, in Section 2.5 we make a survey of rule number reduction methods, that will be extended in Chapter 4.

2.4 Improvements in Frequent Itemset Generation

In this section we will study different techniques and methods developed in the literature aiming to extract rapidly and efficiently frequent itemsets. A selection of the main algorithms on frequent itemset mining is presented by Celgar and Roddick 2006 [51].

One of the first techniques for the discovery of correlations between boolean values appeared as early as 1966, the GUHA method, which introduces support and confidence metrics and which was proposed by Hajek *et al.* 1996 [97]. Several years later, the interest in the extraction of correlations from a database increased and a big volume of dedicated algorithms was developed.

2.4.1 Candidate Generation Algorithms

Candidate generation algorithms are based on the following technique: a set of candidate itemsets are identified in order to be validated with respect to the incorporated constraints. The first algorithms proposing the discovery of frequent itemsets using the candidate generation technique are AIS developed by Agrawal *et al.* 1993 [5] and SETM proposed by Houtsma and Swami 1995 [110]. These two algorithms generate candidates by scanning frequent itemset lattices. Unfortunately, they do not make an important effort in the reduction of the combinatoric explosion. As a result, an important number of improvements were brought once with new generation algorithms.

Apriori [6] is known as *the* frequent itemset generation algorithm based on candidate generation. Nevertheless, it is important to note that, at the same time (but independently), other researchers came up with the same idea of the reflexive constraint inclusion (support), and they proposed *OCD (Offline Candidate Determination)* [139].

Apriori algorithm suggested a standard for reducing the search space by the proposed heuristic support. An important number of algorithms, based on *Apriori*, were proposed with the aim to optimize the frequent itemset generation by introducing condensed representations, dataset partitioning, dataset pruning or dataset access reduction. Among the new algorithms we outline the most important ones: *Apriori-TID* [6], *Direct Hashing and Pruning (DHP)* [157], *Partition* [177], *Dynamic Itemset Couting (DIC)* [37], *Tree Projection* [4].

Using a new database for counting the itemsets support, *AprioriTID* algorithm, developed by Agrawal and Srikant 1994 [6], extends *Apriori* through decreasing the number of passes over the database. This new database has the form $< TID, C_k^i >$, where TID is the identifier of an itemset, and C_k^i represents the subsets of the itemset TID of k length. Thus, transactions are represented by the k-itemsets that describe them. Nevertheless, the problem of the high dimension of this new database for lower values of k is very important.

Direct Hashing and Pruning (DHP) algorithm, suggested by Park *et al.* 1997 [157], uses hashtables in order to reduce the number of candidates. The type of tree structures used during the candidate generation phase is important because it permits different access to elements, or different representations of the items/itemsets. Thus, we can distinguish between 3 main data structures: hash trees, enumeration-set trees and prefix trees. For instance, the difference between enumeration-set trees and prefix trees comes from the fact that, on the one hand, for prefix trees case, the nodes are items and the itemsets are constructed during depth first search in the tree, and, on the other hand, in an enumeration-set tree the nodes are itemsets. The *Tree Projection* algorithm, developed by Agarwal *et al.* 2001 [4], uses an enumeration-set tree to generate candidates.

As its name denotes, *Partition* algorithm, developed by Savasere *et al.* 1995 [177] is based on the idea of partitioning the database in order to fit into the memory.

Dynamic Itemset Couting (DIC) algorithm, introduced by Brin *et al.* 1997 [37], reduces the number of passes over the database by introducing a new interesting idea – $(k + 1)$ candidates are computed from the k pass. When a k-itemset is considered frequent, all the $(k+1)$-itemset candidates that the latter can produce are generated.

2.4.2 *FP-Growth* Algorithms

The algorithms based on candidate generation are an intuitive solution for frequent itemset extraction, but also a resources consuming one. In the last ten years, this situation led to the aim of proposing new techniques allowing frequent itemset discovery and improving the user of resources.

Starting with 2000, *Pattern Growth* algorithms were introduced; they have the specificity to use complex *hyperstructures* for data storage. Generally, a hyperstructure is composed of two principal structures:

- *Item List* – contains the list of frequent items. Each item is *linked* to the first element in the pattern frame that contains it;
- *Pattern Frame* – represents a tree structure containing items with their support. The particularity of this tree is that it is constructed in a database pass by using each transaction.

The first pattern growth algorithm is the *FP-Growth* algorithm introduced by Han and Pei 2000 [98]. Later, in 2005, Grahne and Zhu [85] developed *FP-Growth** algorithm which improves the previous algorithm performances due to *FP-array*, a new data structure. *FP-array* permits to improve the passes over the FP-tree.

Example 2.4.1 Let us consider the data base presented in Example 2.3.14 slightly modified (Table 2.4). We will consider a support threshold of 50%.

Table 2.4: Supermarket database sample for FP-Growth algorithm example.

Tuple TID	Transaction
1	pork, milk, pear
2	pork, milk, apple
3	pear, apple
4	pork, milk, pear

In the first pass over the database, the support of each item is computed, and infrequent items are discarded. Next, frequent items are sorted in decreasing order based on their support. In the Table 2.4, the items are already sorted, and we can note that none of the items is infrequent.

Figure 2.3 presents the second pass over the database. During this pass the FP-tree is constructed as follows:

- transaction 1: $\{pork, milk, pear\}$ – three nodes are created (*pork*, *milk* and *pear*) and their counts are set to 1. Also, the path $null \rightarrow pork \rightarrow milk \rightarrow pear$ is created;

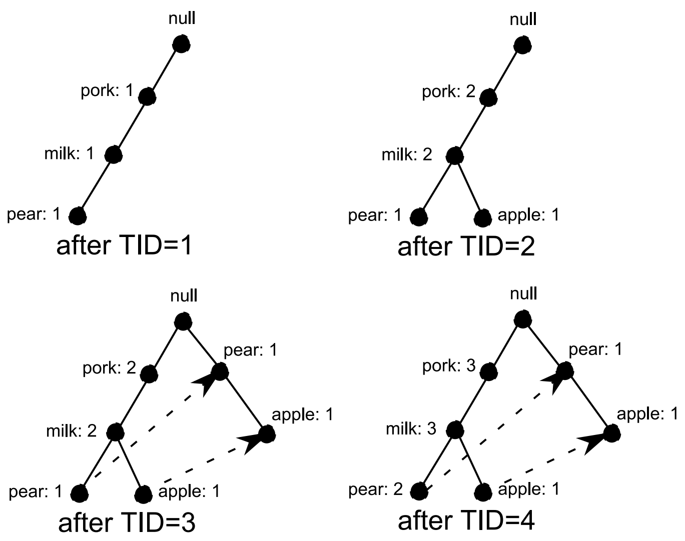


FIGURE 2.3: An example tree for the frequent itemset generation using the FP-Growth algorithm.

- transaction 2: {*pork*, *milk*, *apple*}. As the first and the second transaction share the same prefix (items *pork* and *milk*), the first part of this path is common to the previous created path. Thus, one node is created (*apple*) and its count is set to 1, while the *pork* and *milk* counts are incremented to 2;
- transaction 3: {*pear*, *apple*}. As this transaction does not share the prefix with the previous two transaction, 2 nodes are created (*pear* and *apple*), and their counts are set to 1;
- in the same manner transaction 4 is treated.

Thus, the frequent itemsets are: {*pork, milk, pear*}, {*pork, milk, apple*} and {*pear, apple*}. ■

2.4.3 Condensed Representations based Algorithms

Apriori-based algorithms perform one pass over the database for each itemset length growth. Thus, the number of passes of an *Apriori*-based algorithm depends on the length of the largest frequent itemset, growing exponentially the execution time [38]. In this context, discovering all the frequent itemsets in a large database becomes quickly an intractable problem. This exponential complexity is fundamentally restricting *Apriori*-like algorithms to discover only short patterns.

Nowadays, the literature proposes two main approaches to the long itemset mining problem. The first one is to extract only the maximal frequent itemsets, and the second one to extract frequent closed itemsets. These approaches are detailed in this section.

Since frequent itemset generation is considered as an expensive operation in terms of database passes, mining *frequent closed itemsets* was introduced in order to reduce the number of frequent itemsets. The bases of this theory were first presented by Zaki et Ogihara 1990 [219]. The authors developed a formal framework for the problem of association rules based on the Formal Concept Analysis (FCA) introduced by Wille 1982 [215]. FCA is a mathematical approach to data analysis based on the lattice theory presented in [27]. The FCA method is applied in a great variety of research fields such as psychology, ecology, library, information science, or software engineering. One of the most important goals of the FCA is to produce graphical visualizations of the conceptual structures in data.

A basic notion in FCA is the *formal context* defined as a triple $< O, A, R >$, where O is a set of objects, A is a set of attributes, and $R \subseteq O \times A$ is a binary relation between O and A which should be read as *the object* O *has the attribute* A. The definition of the formal context is quite close to the definition of the association rule mining problem. Thus, it is not very difficult to define a formal framework for association rule mining based on this method.

A *data mining context* [160] is defined as a triple $D =< O, I, R >$ where D is the database. In this context, for the association rule technique O is a set of transactions, I is a set of items and R is the binary relation between O and I. Each couple $(t_i, i_j) \in R$ describes the existence of the item i_j in the transaction t_i. Thus, we can define the formal context for the association rule technique as $D =< T, I, R >$. ■

Definition 2.4.2

Let $(P, \leq)$ be an ordered set with the binary relation $\leq$, and let S, $S \subseteq P$. An element $u \in P$ $(l \in P)$ is a upper bound (lower bound) of S if $s \leq u$ $(s \geq l)$ $\forall s \in S$. The least upper bound is called the join of S, and the greatest lower bound is called the meet of S [220]. ■

Definition 2.4.3

An ordered not-empty set $(L, \leq)$ is a lattice if, $\forall x, y \in L$ the set $\{x, y\}$ has a join and a meet [220]. An ordered not-empty set $(L, \leq)$ is a complete lattice if all the subsets $P \subseteq L$ have a join and a meet. ■

Definition 2.4.4

Let us consider the formal context presented below. For $O_i \subseteq O$ and $A_j \subseteq A$ two mappings are defined as follows. The first one, projects the objects in the set of attributes:

$$\begin{aligned} f &: \mathcal{P}(O) \to \mathcal{P}(A) \\ &f(O_i) = \{a \in A \mid \forall o \in O,\ (o, i) \in R\} \end{aligned}$$

and the second one projects the attributes in the set of objects:

$$\begin{aligned} g &: \mathcal{P}(A) \to \mathcal{P}(O) \\ &g(A_j) = \{o \in O \mid \forall a \in A,\ (o, i) \in R\} \end{aligned}$$

The couple (f, g) is the Galois connection, and the operators $h = fog$ and $h' = gof$ are Galois closure operators [77, 215]. ■

If we transpose this theory in the context of association rule mining, the set of objects are represented by the transactions, and the set of attributes, by the items in the database. Thus, using the Galois closure operators, we are able to define the notion of closed itemset.

Definition 2.4.5
A set of items $X \subseteq I$ is called closed itemset if and only if $h(X) = X$. The minimal closed itemset of X is generated by applying h to X, $h(X)$. ■

Definition 2.4.6
A formal concept is a pair $c = (O_i, I_j)$, with $O_i \subseteq O$ and I_j, if O_i and I_j are relied by the Galois operator, i.e. $h(O_i) = I_j$ and $h'(I_j) = O_i$. O_i is called extension and I_j is called intension. ■

Definition 2.4.7
A closed itemset lattice is defined as the ordered set of all closed itemsets, $P = \{X \subseteq I \mid h(X) = X\}$. The lattice is described by: $L_P = (P, \leq)$. ■

Definition 2.4.8
A closed itemset $X \subseteq I$ from the formal context $D =< O, I, R >$ is denoted as frequent if its support is higher than threshold of minimum support, $supp(X) \geq minsupp$. ■

Several algorithms were proposed in the literature in order to extract frequent closed itemsets. The first proposition was made in 1998, by Pasquier *et al.* with the *Close* algorithm [160, 161], and later, with an improved version, *A-Close* [159]. These algorithms start, as *Apriori*, from a lexicographical ordered set of *1*–itemsets, and the generation of frequent closed itemsets is an iterative process. In the first step, starting from generators (elements to which the galois closure operator h is applied) frequent closed itemset candidates are generated. Applying support constraints, frequent closed itemsets of this level are determined. In the next step, starting from frequent closed itemsets, the generators for the next level are computed. The pseudo–code of the *Close* algorithm is presented in the Table 2.5.

PASCAL algorithm was developed by Bastide *et al.* 2000 [18] as an improvement of earlier frequent closed itemset mining algorithms, but very soon it was outperformed by a new proposition. The *CLOSET algorithm* was developed by Pei *et al.* 2000 [163] as a new efficient method for mining closed itemsets. *CLOSET* uses a novel frequent pattern tree (FP-tree) structure which is a compressed representation of all the transactions in the database. Moreover, it uses a recursive divide-and-conquer and database projection approach to mine long patterns.

Introduced by Zaki and Hsiao 2002 [222], the *CHARM* algorithm expands the search space to itemset and transaction space using a representation structure - itemset-tidset tree, called *IT–tree*. Moreover, the authors suggested to use a new data structure (*diffset*) so that the frequencies of itemsets should be easier computed. The search method employed in the approach is more efficient than the other search methods proposed so far, frequent closed itemsets being discovered after less enumerations.

Table 2.5: Frequent close itemset generation algorithm [160].

```
1. Input: The database $D$
1. Output: The set of frequent closed itemsets $FC$

1. generators in $FCC_1 \leftarrow \{$1-itemsets$\}$
2. forall ($i \leftarrow 1$; $FCC_i.generator \neq \emptyset$; $i++$) do begin
3.    closures in $FCC_i \leftarrow \emptyset$
4.    supports in $FCC_i \leftarrow 0$
5.    $FCC_i \leftarrow$ Gen-Closure($FCC_i$)
6.    forall candidate closed itemsets $c \in FCC_i$ do begin
7.       if ($c.support \geq minsupport$) then
8.          $FC_i \leftarrow FC_i \cup \{c\}$
9     $FCC_{i+l} \leftarrow$ Gen-Generator($FC_i$)
10. Return $FC \leftarrow \bigcup_{j=1}^{j=i-1} \{FC_j.closure, FC_j.support\}$
```

The advantage of extracting the set of frequent closed itemsets is that no information is lost because they determine the complete set of frequent itemsets and their exact frequency.

Another solution for reducing the number of frequent itemsets is mining *maximal frequent itemsets*. A maximal frequent itemset is a frequent itemset without any frequent superset. Thus, the set of maximal frequent itemsets is more reduced comparing to the complete set of frequent itemsets and even comparing to frequent closed itemsets. Using a bottom-up and top-down search, these algorithms assure a reduced execution time.

The most important algorithm was proposed in 2001 by Burdick *et al.* [43] and it was called *MAFIA algorithm*. It is based on the depth-first traversal and on several pruning methods as Parent Equivalence Pruning (PEP), FHUT, HUTMFI or Dynamic Recording. But, the main drawback of maximal frequent itemsets methods is the loss of information because the subset frequency is not available, thus, in this case, the generation of rules is impossible.

Nevertheless, Zaki et Hsiao proved in their paper [222] that the developed algorithm, CHARM, outperforms CLOSET, Close, and Mafia algorithms.

2.5 Improvements in Rule Generation

In the previous section we studied about the improvements proposed in the literature in the generation of frequent itemsets. This section is dedicated to the improvements in rule generation – in other words, we will discuss about the solutions developed to reduce the number of discovered rules or to improve their quality.

Among available techniques for rule reduction, in this section we discuss about constraint-based association rule mining techniques and about the reduction of the redundancy of rules.

2.5.1 Constraints-based Association Rule Mining

In this particular context where huge volumes of association rules are generated by classical techniques, the paradigm of constraint-based rules mining was introduced [71]. It provides to the user the possibility to impose a set of constraints over the content of the discovered rules which could produce an important reduction of the rule volume. Moreover, an important part of computation operations are avoided, improving the execution time or decreasing the used resources. Generally, constraints ($\mathcal{C}$) are provided by means of different formalisms – user knowledge constraints, data constraints, dimensional constraints, interestingness constraints, and rule constraints.

The role of constraints is very well-defined – they generate only those association rules that are interesting to users [223], and the technique is quite trivial – the rule space is reduced so the remaining rules satisfy the constraints. If we take the example of the *Apriori* algorithm and its variants, they use two basic constraints: minimal thresholds for support and confidence. Nevertheless, even if these two constraints are basic, it is quite difficult to find the right values that produce interesting rules. Using wrong thresholds could have two consequences: first, the algorithm could miss some interesting rules and, second, it could generate trivial ones. Further, users could have difficulties to understand the meaning of data-oriented constraints and to give minimal thresholds.

Ng *et al.* 1998 [150] introduced a first classfication of two types of constraints according to two orthogonal properties: *anti-monotonicity* and *succinctness.*

Definition 2.5.1
Given an itemset X, a constraint $\mathcal{C}_{AM}$ is anti-monotone if $\forall Y \subseteq X$, $\mathcal{C}_{AM}(X) \Rightarrow \mathcal{C}_{AM}(Y)$. The most known anti-monotone constraints is the frequency (support) constraint ($\mathcal{C}_{freq}$). It is integrated in the Apriori-like algorithms with the following interpretation: if an itemset X does not satisfy the frequency constraint $\mathcal{C}_{freq}$, then no superset of X can satisfy $\mathcal{C}_{freq}$, and, they are pruned. Other $\mathcal{C}_{AM}$ constraints can easily be pushed deep down into the frequent itemset mining computation since they behave exactly as $\mathcal{C}_{freq}$:if an itemset does not satisfy the constraint, then none of its supersets satisfy it. ■

Succinct constraints are totally different from anti-monotonic constraints. While the latter are applied iteratively over sets of candidate itemsets, the former has the particularity of being able to generate itemsets without generating candidate itemsets.

Example 2.5.2 Let us consider the database in Table 2.6; each item is described by a weight as showing Table 2.7.

The constraint $sum(weight) \leq 10$ requires that for an itemset X, the total weight of the items in X must be no greater than 10. It is an anti-monotone constraint, because if an itemset, i.e. *pork*, violates the constraint, all of its supersets will violate it; as a consequence, the itemset *pork* can be removed from the candidate set during

Table 2.6: Supermarket database sample for constraints-based rule mining.

Tuple	Transaction
1	milk, pork, pear
2	milk, pork, apple
3	pork, pear
4	milk, pork, pear

Table 2.7: The weights (such as profit) of items.

Item	Weight
milk	2
pork	20
pear	10
apple	9

an *Apriori*-like frequent itemset mining process. On the contrary, the *1*-itemset *milk* has the weight of 2, so, it satisfies the constraint, and it can participate to the generation of *2*-itemsets. Further, the itemset *milk apple* is generated, but its weight is 11; in this case, this itemset does not satisfy the constraint, and it will be pruned.

The constraint $max(weight(\phi)) < 15$ requires that for an itemset X, each item must have the weight no greater than 14. It is an succinct constraint, in the sens that all the items of the itemset must satisfy the constraint, in order to be satisfied by the itemset. For instance, the itemset *milk pear* satisfies the constraint, but the itemset *milk pork* not. ■

In the following, we will outline the main approaches proposed in constraint-based association rule mining. Ng *et al.* 1998 [150] and Srikant *et al.* 1996 [190, 191] have investigated applying item constraints for the generation of frequent itemsets. They restrict the items or the combinations of items that are allowed to participate in the mining process. Earlier, in 1992, Smyth and Goodman 1992 [187] described a constraint-based rule miner integrating an interestingness constraint described by the dimension of rules, thus, long rules are considered less interesting. Padmanabhan *et al.* 2000 [156] developed a new algorithm, called *ZoomUAR*, which integrates constraints describing user knowledge expressed in a rule-like formalism of the form $A \to B$. The main idea is that, interesting rules contradict the user beliefs from a logical point of view. This approach will be detailed in Chapter 4, Section 4.4.6.

Bayardo *et al.* 1999 [20] developed an new algorithm, called *Dense-Miner* which integrates two types of constraints. First, consequent constraints help the authors to reduce the set of rules by constraining the consequent of all the rules to a certain itemset. Second, they proposed the *Minimum Improvement Constraint* in order to prune uninteresting rules. The Minimum Improvement Constraint of a rule R is computed using the entire set of general rules which can be generated starting from R.

Definition 2.5.3

Let us consider the $R_{spec} : X \rightarrow Y$ association rule and $R_{gen} : X' \rightarrow Y$ where $X' \subset X$; that is to say that the rule R_{spec} is a specification of the rules R_{gen}, or, inversely, the rule R_{gen} is a generalization of the rule R_{spec}.

The improvement of the rule R_{spec} is computed as the minimum difference between its confidence and the confidence of its more general rules. More formally, it can be defined as:

$$imp(R_{spec}) = min(conf(R_{spec}) - conf(R_{gen}) \mid \forall R_{gen}, X' \subset X). \blacksquare$$

Further, if the improvement of the rule R_{spec} is positive, then we can conclude that this rule is interesting because it brings more information than all its general rules separately. If the improvement is negative we can conclude that the rule may improve the confidence of several general rules, but not of *all* of them. In other words, the rule R_{spec} could be simplified without loosing any information.

The threshold of minimum improvement is provided by the user (i.e. 5%) with the support and confidence.

Example 2.5.4 To illustrate the minimum improvement constraint, we will consider the following set of association rules

$$\begin{aligned}
R_1 : milk, pork \rightarrow pear \quad & [S = 20\%, \;\; C = 80\%] \\
R_2 : milk, apple \rightarrow pear \quad & [S = 27\%, \;\; C = 76\%] \\
R_3 : milk \rightarrow pear \quad & [S = 25\%, \;\; C = 70\%] \\
R_4 : pork \rightarrow pear \quad & [S = 30\%, \;\; C = 72\%] \\
R_5 : apple \rightarrow pear \quad & [S = 40\%, \;\; C = 83\%]
\end{aligned}$$

where R_1 is the specialization of rules R_3 and R_4, and R_2 is the specialization of rules R_3 and R_5. Also, we consider that the threshold of minimum improvement is 5%.

Thus, we can compute the improvement of R_1 and R_2 as follows:

$$\begin{aligned}
imp(R_1) &= min(conf(R_1) - conf(R_3), conf(R_1) - conf(R_4)) \\
&= min(10, 8) \\
&= 8 \ (\%) \\
imp(R_2) &= min(conf(R_2) - conf(R_3), conf(R_2) - conf(R_5)) \\
&= min(6, -7) \\
&= -7 \ (\%)
\end{aligned}$$

and we can conclude that the rule R_1 – with an improvement of 8% which exceeds the minimum improvement threshold of 5% – is an interesting rule because it brings additional information comparing to its generalizations. On the contrary, the improvement of rule R_2 is negative, less than the threshold limit of 5%, so, the rule is not interesting. ■

2.5.2 Redundancy Rule Reduction

This second section of rule extraction improvement is dedicated to redundant rule reduction techniques. Classical algorithms propose good methods for association rule extraction, but the number of rules is too large and too many rules are redundant.

Different definitions are suggested in the literature for redundant rules; in the following we will use the definition suggested by Zaki 2000 [221].

Definition 2.5.5
Zaki 2000 [221] denotes generalization/specialization relation between two rules by $R_1 \preceq R_2$ which describes that rule R_1 is more general than the rule R_2, or, on the contrary, that the rule R_2 is more specific than the rule R_1.

A rule R_j is denoted as redundant if $\exists R_i,\ R_i \preceq R_j$, where $R_i, R_j \in R = \{R_1, \ldots, R_n\}$, R being a set of rules with the same support and confidence.

In conclusion, non-redundant rules are the most general rules in a rule set.

In 2004, Zaki [220] proposed to extract non-redundant rules using frequent closed itemsets. He suggested the first ideas concerning non-redundant rule generation in 2000 [221], but it is in 2004 that he developed a real algorithm. Let us consider two closed itemsets X and Y, where $X \subseteq Y$. In a first step, the algorithm developed by Zaki produces the set of minimal generators for each itemset. A generator X' of an itemset X is a subset of X ($X' \subset X$) that has the same support as X ($supp(X) = supp(X')$). A generator of X which has no subset generators of X is called minimal generator of X. Then, each minimal generator can be the left part or, respectively the right part of an non-redundant association rule.

Comparable with the propositions of Zaki, Pasquier *et al.* 2005 [162] introduced two condensed association bases to represent non-redundant association rules: *Min-max Approximative Basis* and *Min-max Exact Basis.* These two definitions describe different possibilities to extract non-redundant rules using closed methods. Later, Xu and Li 2007 [218] improved the definitions suggested by Pasquier *et al.* proposing a more concise association rule basis called *Reliable exact basis.*

Nevertheless, both closed and maximal itemset mining *still break down at low support thresholds.* To address these limitations, Omiecinski 2003 [153] proposed three new important interestingness measures: *any-confidence*, *all-confidence* and *bond.* All these measures are indicators of the degree to which items in an association are related to each other. The most interesting one, *all-confidence*, introduced as an alternative to support, represents the minimum confidence of all association rules extracted from an itemset. *Bond* is another measure of the interestingness of an association. It is similar to support, but with respect to a subset of the data rather than the entire database.

Li 2006 [130] proposed recently *optimal rule sets* defined with respect to an interestingness metric. An optimal rule set contains all rules except those with no greater interestingness than one of its more general rules. A set of techniques for the reduction of redundant rule number was developed and implemented by Ashrafi *et al.* 2005 [13]. The proposed techniques are based on generalization/specification of antecedent/consequent of the rules and they are divided in methods for multi-antecedent rules and multi-consequent rules. Complementary, several researches suggested that pruning redundant rules can seriously decreases rule number and the number of redundant rules is exponential in the length of the longest frequent itemset [220].

A novel reducing redundancy technique based on *rule covers* was developed by Toivonen *et al.* 1995 [204]. The notion of rule cover defines the subset of a rule set describing the same database transaction set. The authors developed an algorithm to efficiently extract a rule cover from a set of given rules.

2.6 Conclusions

In this chapter, we described Knowledge Discovery in Databases process and, more particularly, we focused on the association rule mining technique. We presented the definitions, the notations and the most important drawbacks that limit the mining potential of the technique.

In this context, we studied different existing methods that ameliorate the association rule mining process. First, for the rapidity problem, we detailed different algorithms developed in the literature with the aim to reduce the execution time and the resources that are used. Second, to increase the efficiency of the rule generation process – the reduction of rule number – different methods are proposed with the aim to filter the rules. In this chapter we studied two of them – constraint-based association rule mining and redundancy rule reduction.

The advantages of these techniques are numerous. For instance, mining algorithms based on frequent closed itemsets are at the same time able to reduce the time execution and to considerably reduce the rule volume. The same improvements are also brought by constraints-based association rule mining algorithms. Redundancy reduction methods are generally known for the important reduction of the number of rules and for the significant increase of the quality of the discovered rules.

Nevertheless, these approaches have their own limits. First, we will focus on the number of generated rules which is still important to be processed by a user. These techniques could permit to reduce the number of rules from millions to thousands, but even thousand rules a user is not able to exploit manually. In conclusion, the *useful* characteristic of patterns suggested by Fayyad *et al.* cannot be met in this case. At the same time, these techniques are generally based on database information; nevertheless, the interest and the utility of a rule is decided by the user. Thus, *new filters/techniques* are needed to extract reduced sets of rules which are useful and interesting for the user.

3 Knowledge Representation: Ontologies and Semantic Web

Knowledge is power

Francis Bacon

Contents

3.1 Introduction

In the previous Chapter, we studied the association rule mining technique and we outlined its main problem – the mining technique discovers high volumes of rules. Moreover, the generated rules are trivial, redundant or uninteresting for the user. Unfortunately, techniques such as redundancy rule reduction do not guarantee that the selected rules are interesting for the user. In this context, the integration of user knowledge in the rule discovery process could produce a notable increase of the rule quality. For this purpose, since the more the knowledge is represented in a flexible, expressive and accurate formalism, the more the rule selection is efficient, the user knowledge should be represented by means of an adequate formalism.

In this context, *Knowledge Engineering (KE)* research field attempts to describe a set of techniques and concepts which aim to acquire, represent and exploit domain knowledge in order to be integrated in *Knowledge-Based Systems (KBS)*. KBSs are described as systems based on knowledge and reasoning mechanisms which propose solutions to real-world problems. In this context, *Knowledge Representation (KR)* is the area of the Artificial Intelligence (AI) field which aims to propose suitable formalisms for domain knowledge representation with important facilities for knowledge reusing, shearing and inference [34]. The knowledge can come from different sources: documents, archives, Web and experts. In the case of expert knowledge, a step of knowledge acquisition is developed which sometimes is more difficult than the knowledge representation one.

In this Chapter we will focus on Knowledge Representation field and we will study ontology-based formalisms. A first objective is to outline the development of models from controlled vocabularies to ontologies. An ontology is a *a formal, explicit specification of a shared conceptualization* [89, 90]. Ontologies, with concepts, relations and axioms, offer a powerful and complete formalism for human knowledge. Further, reasoning techniques complete the representation by allowing ontologies to infer information in order to generate new knowledge.

Second, we will stop over the knowledge representation languages developed in the Semantic Web field. The latter has as main purpose to provide representation languages which are comprehensible by machines and enough powerful and expressive when used for user knowledge representation. We will focus on XML, RDF and OWL, and we will give a detailed presentation of the advantages of OWL language – description logic and inference engines.

3.2 From controlled vocabularies to ontologies

As suggested in [68], The word *ontology* comes from the Greek *onto*, meaning *of being*, and *logos* meaning *science, theory*, and describes a philosophical study of the nature of the existence. This idea was introduced for the first time by Aristotle, in his Metaphysics, as the categorization of things regarding the way in which they exist, but the word *ontology* will be used 1900 years later.

Since their first apparition in the philosophy branch, ontologies have evolved and have been used in several domains. But, it is at the beginning of the 20th century that philosophers studied the methods for building ontologies, while in computer science researchers tried to build them directly. The Artificial Intelligence field [142] aims to develop intelligent machines with a human reasoning; during its evolution, a great part of the research made in the philosophy branch was assimilated. In consequence, ontologies were considered as being the most appropriate solution for knowledge representation in Information Systems.

Before giving a formal description of an ontology, we will make a short presentation of their degree of formality since organizations or controlled vocabularies until ontologies. This idea was firstly discussed in [210], where Uschold and Grüninger study the degree of formality of ontologies. In this context, our presentation starts with the most informal representations and we will end with the ontologies in high formal representations as pointed out in Figure 3.1.

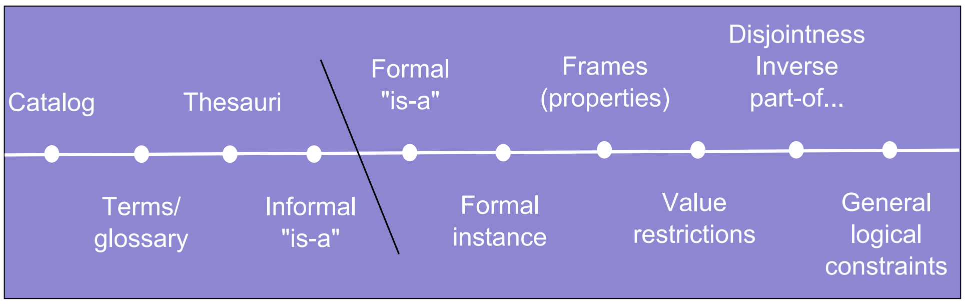


FIGURE 3.1: The degree of formality of ontologies [126].

3.2.1 Controlled vocabularies

Humans prefer to have an organized life. The nature of the human being is to organize, to structure things into groups depending on their differences and on the elements they have in common. In order to permit the interoperability of these groups, a vocabulary must be defined; generally, a controlled vocabulary registration authority is in charge of this. *Subject–based classification* [80] represents the method of grouping/classifying objects regarding their subject.

The advantage of controlled vocabularies is that they facilitate the research in a database. A controlled vocabulary is defined as a list of *terms*, where a term is a particular name for a *concept*. As already noted, the list of terms is controlled by a controlled vocabulary registration authority. It is obvious that it may be possible that the same term describes multiple concepts, and that a concept is named by several terms. That depends on how strict the controlled vocabulary registration authority is, but generally two rules are imposed [167]:

- if there is a term with the same name for different concepts, then its name is explicitly qualified to solve the ambiguity;
- if several terms are used in order to denote the same concept, one principal term is selected, and the other terms are defined as synonyms for the principal one.

For instance, a catalog can be an example of controlled vocabulary because it provides a unambiguous interpretation of terms.

Example 3.2.1 Let us consider the following two phrases *I like green apples* and *I like Apple's iPhone.* We can remark that in these phrases the word *apple* does not name the same concept, as the *Fruits* group does not use the concept *apple* in the same way as the *Phones* groups. In the first case, an *apple* is a *Fruit*, which is not related to *Apple Company.* ■

3.2.2 Glossaries and Thesauri

After controlled vocabularies, Glossaries represent the next solution for ontology specification. They consist of a list of terms with meanings. The meanings are generally described in natural language, so the humans easily understand them.

More developed, thesauri integrate additional meaning (semantics) for the relations between terms. Generally, these relations are linguistics such as synonyms, antonyms, homonyms[1], etc. In addition, thesauri do not provide an explicit hierarchy, but we could deduce one using thesaurus terms.

One of the most known thesauri for English language is WordNet[65], which is described by its authors as the lexical reference system. Its design is inspired by current psycholinguistic theories of human lexical memory. WordNet 2.0 contains a total of 203,145 word-sense-pairs, and relates terms as synonyms, hypernyms[2], hyponyms[3], holonyms[4] and meronyms[5].

3.2.3 Taxonomies

The term of taxonomy was introduce by Carl von Linné[6] who structured the system of life forms. A taxonomy is a subject-driven classification of the terms of a controlled vocabulary into an hierarchy. The process of classifying terms in a taxonomy is comparable to the one of connecting the terms using a specialization relation, called *is-a* relation. In natural language, this relation is *father to son* relation translated by: *A is-a B* or *A is the son of B* or *B is the father of A.*

Example 3.2.2 Figure 3.2 presents a sample of a taxonomy on supermarket products.

For example, the nodes are products in the supermarket, and the $\rightarrow$ arcs represent the *is-a* relation. Thus, each element of the taxonomy *is-a Fooditem.* For instance, if we consider the *Meat* element, it is in a relation *is-a* with the *Fooditem* element, and it has *three sons*: *beef*, *pork* and *chicken.* ■

[1]the same words with different senses
[2]A word that is more generic than a given word
[3]A word that is more specific than a given word
[4]A concept that has another concept as a part
[5]A concept that is part of another concept
[6]Swedish botanist, physician, and zoologist, 1707 – 1778

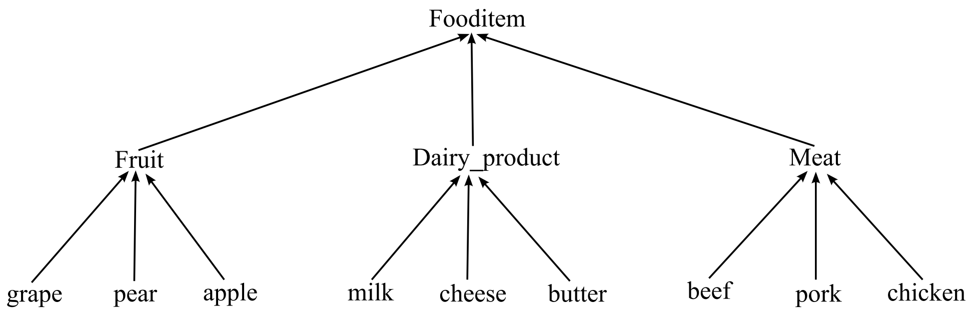


FIGURE 3.2: Example of taxonomy – supermarket taxonomy [136].

3.2.4 Ontologies

Taxonomies represent a great advancement in knowledge representation, but more complex systems are needed in order to represent the functioning of the reasoning process behind the human thinking. For example, taxonomies permit only an *is-a* relation between elements, but a more expressive structure could allow the definition of different properties (relations). For instance, in our case of supermarket food products, the *isEatenWith* property could connect two products in order to express which two products can be eaten together. Moreover, types of products can be defined by using restrictions. For example, if we add the property *hasPrice* – expressing the price of a product – between a product and a value, we can say, for instance, that if the price of a product does not overpass a x value, the product is a part of the *CheapProducts*.

3.3 Ontology – Definitions and Structure

In its very early definition, the Knowledge Engineering was viewed as a transfer process *of problem–solving from a knowledge source to a program* [100]. The transfer process is generally based on the idea that the knowledge necessary to solve a problem already exists (i.e. the expert knowledge). Thus, the KE concentrates on the acquisition of problem solving knowledge [147]. Later, several remarks came out pointing that a part of expert knowledge is hidden in expert skills which are difficult to collect. This is why the transfer process is replaced with a model construction process [145], permitting to build up and structure the knowledge. It is obvious that this type of model could not collect all expert knowledge, but it can become a good approximation of the real expert knowledge. It is also important to note that, in real life, daily modifications of expert knowledge are possible, so the model should be cyclical [195]. Three important frameworks can be cited: *CommonKADS*, by Schreiber *et al.* 1999 [178], suggests for construction a set of different models depending on the KBS specificity; *MIKE (Model-based and Incremental Knowledge Engineering)*, by Angele *et al.* 1993 [9], improves the existing models with the integration of an incremental system development process into a model-based framework; and *PROTEGE–II*, by Musen *et al.* 1995 [146], proposes to modelize KBSs using ontologies.

Knowledge Representation (KR) is the area of the Artificial Intelligence (AI) field concerned with how knowledge can be formally represented [34]. In this context, *Semantic Networks* [188] were proposed in early 1970s as a model for relational references including conceptualization. In the same line of Knowledge Representation formalisms, *Frames* [90] were introduced in 1993 as a model described using *classes*, *slots*, *facets* and *instances*. The most complex language, *Description Logic – DL* [14] was proposed as an extension to the first two ones incorporating formal logic-based semantics and allowing inference process. In the last decade, having their bases on description logics, ontologies came out as being the closest representation language to human reasoning.

Since the introduction of the notion of ontology, several definitions have been proposed in the literature. We consider that the most appropriate one is the proposition of Gruber 1993 [89, 90] which describes the ontology as *a formal, explicit specification of a shared conceptualization.* In order to understand this definition, it is important to define the notion of conceptualization. By *conceptualization* we understand an abstract model of some phenomenon in the world described by its important concepts. The *formal* notion denotes that machines should be able to interpret an ontology. Moreover, *explicit* term refers to the explicit definition of ontology elements. Finally, *shared* outlines that an ontology groups some knowledge common to a certain group, and not individual knowledge [195].

In 1995, Guarino [93] criticized the definition of Gruber which relies on an extensional notion of conceptualization. Thus, Guarino proposed in [92] a new definition of an ontology: *An ontology is a logical theory accounting for the intended meaning of a formal vocabulary, i.e. its ontological commitment to a particular conceptualization of the world. The intended models of a logical language using such a vocabulary are constrained by its ontological commitment. An ontology indirectly reflects this commitment (and the underlying conceptualization) by approximating these intended models.*

In 1998, Uschold and Grüninger described the notion of ontology as *the term used to refer to the shared understanding of some domain of interest which may be used as a unifying framework to solve the above problems in the above-described manner* [210].

Later, in 2001, another definition of ontologies was proposed by Maedche and Staab but, this time more a artificial intelligence oriented one. Thus, the authors describe ontologies as *(meta)data schemas, providing a controlled vocabulary of concepts, each with an explicitly defined and machine processable semantics* [138].

Nevertheless, how do ontologies participate to systems improvement ? The analysis made by Gandon 2006 [75] regarding Information Systems can be also applied to general systems; thus, he considers that *the introduction of an ontology in an information system aims to reduce or to exclude conceptual and terminological confusion and to tend toward a shared comprehension in order to improve communication, sharing, interoperability, and the reusability level. An ontology can be viewed as a unifying framework and it provides "primitives", basic elements helping the communication to improve between people, between people and systems, and between systems. Integrating an ontology in an information system allows to formally declare a number* *of knowledge used to describe the information managed by the system and to rely on these descriptions and on the formalization of their meaning in order to automate the information processing tasks.*

3.3.1 Concept, relations (properties) and axioms

First, it is important to note that it does not exist a single one correct way to model a domain, there are always viable alternatives. The best solution depends on the application that we have in mind and the extensions that we anticipate [152].

In addition to the definition given by Gruber, Genesereth and Nilsson 1987 [82] described the conceptualization as a set of *objects* and a set of *relationships*. The concepts are the key elements describing a knowledge domain connected between each other by means of relationships. Starting from this idea, Gruber described in [89] the *universe of discourse* as the set of objects of a domain that can be represented in a formal way. These objects are represented as taxonomies enriched by several relationships between the objects. Moreover, axioms may help users to correctly interpret the ontology elements.

This definition was extended by Maedche and Staab 2001 [138] and reviewed by Pretorius 2004 [170] presenting ontology components as follows:

- *L – lexical entries (defined as a set of strings) for concepts and relations* ;
- C – the set of *concepts* representing the elements in the domain. This can be any thing, a notion or an idea [211];
- H – the concepts taxonomy (hierarchy) structuring ontology concepts using the subsumption relation;
- R – the set of relations different from the subsumption relation. A relation connects domain concepts with range concepts;
- a set of relations relating lexical entries with concepts from C and relations from R;
- A – a set of axioms bringing additional constraints on the ontology.

To this description we add the set of *instances (I)* which represent the individuals of the concepts.

A *concept* is defined by one term, one notion (or semantical significance) and a set of objects. There are two ways of interpreting the concepts: in intension or in extension [33]. The concept is described in extension using the set of objects, generally denoted as instances of the concept, while the concept semantics (intension) is expressed by its properties and relations. If we consider the concept of *happiness* we can consider that this concept is described in intension because we cannot verify all the happiness in the world in order to correctly describe the concept. Thus, it is important not to confuse the two descriptions. On the contrary, the concept *grape* can be described in intension and extension, as the following example outlines.

Example 3.3.1 Let us consider the *grape* element in the taxonomy in Figure 3.2 on page 35. We can extend its description as follows: *a grape is a fruit that can be eaten with cheese and used to cook beef or pork. White and red grapes are varieties of grapes.* In intension, *grape* concept is defined by using the following three concepts: *cheese*, *beef* and *pork*. The Figure 3.3 presents the extension of Figure 3.2. ■

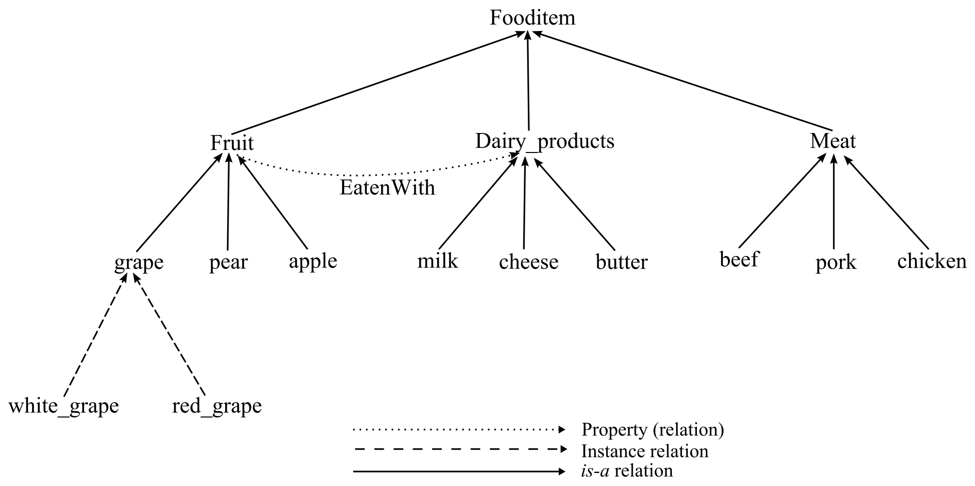


FIGURE 3.3: Example of ontology.

We can note that the development of a concept is also based on other concepts, either in terms of identity or in terms of difference. It is considered as an element located within a structured network, a network representative of a knowledge domain where concepts are linked by relationships.

These relations represent relevant associations between concepts as follows:

- *subsumses the (is-a)*: generalization/specialization relation.

  A taxonomy of concepts organizes the ontology concepts in a hierarchical structure. The concepts are connected using the *relation of subsumption*. We say that the concept C_1 is subsumed by the concept C_2 if C_1 has all the characteristics of C_2. We say also that the concept C_1 is the *child* of the concept C_2 or that C_2 is the *parent* of C_1. Thus, we will have a difference between specific/general concepts: C_1 is more specific than the general concept C_2.

  Example 3.3.2 Let us consider the ontology presented in Figure 3.3; the specific concept *Grape* is subsumed by the concept *Fruits*. ■

- is a *part of*: aggregation/composition relation.

- is *instance of*: instances are individuals of concepts.

  Example 3.3.3 In our case, the *wine_grape* and *red_grape* elements are individuals of *Grape* concept.

- is in a *relation r* with (property).

  A property connects concepts and the instances of these concepts. Generally, a relation (property) relates two concepts called *domain* and *range.* A first classification of properties is made by the type of the range:

  - *Object properties* connect two concepts; in our example *Grape* concept is connected to *Cheese* concept by the object property *EatenWith*, and to *Beef* and *Pork* concepts by the object property UsedToCook.
  - *Data properties* connect a concept to a data-type concept. They can be viewed as attributes.

  Another property classification can be outlined from the work of Guarino [94] who mainly discussed about the importance of properties for concept definition. First, we have the properties the are used for concept definition and they are essential for the concept existence. Removing these properties implies the loss of these concepts. Second, there are properties that are not crucial for the concept existence, but they bring additional information about the concept in the given domain.

Finally, axioms bring additional information to ontogies comparing to taxonomies. Thus, they allow user to define new constraints in the ontology in order to make implicit facts explicit, as presented in [192].

Designing ontologies is not an easy task. In this context, a set of criteria for ontology designing evaluation were proposed by Gruber 1993 [90] as it follows:

- *Clarity/Completeness* – Ontology terms should be described objectively and they should reveal their real meaning. As a result, definitions of concepts using necessary and sufficient conditions are preferred to partial definitions.

- *Coherence* – It represents the measure which assesses the quality of an ontology. For an ontology to be coherent, axioms should be described in such a manner permitting consistent inferences.

- *Extendibility* – One important characteristic of ontologies is the reusability and extendibility. Due to the formal representation of ontologies, one important advantage is that, for a given project in a specific domain, it is very probable to find ontologies already developed that the users can enrich with additional information.

- *Diversification of hierarchies* – The more an ontology is well detailed concerning its concepts, the easier it will be to add new concepts by using multiple inheritance.

- *Minimum semantic distance between concepts which are children of the same parent* – It is important to gather the most similar concepts, and create subclasses if necessary. It is estimated that a parent should not have more than twenty children [181].

3.3.2 Top level or domain ontologies ?

Ontologies can be classified following two directions: the level of detail and the level of dependence on a particular task or point of view. The former classifies the ontologies depending on the reference/shareable ontologies or off-line/on-line ontologies. Very detailed ontologies are more interesting because they propose a powerful specification of the vocabulary. Unfortunately, rich ontologies limit their development in collaborative tasks, while very simple ontologies can be shared between collaborators since they are easier to validate.

The second dimension allows ontologies to be classified in four classes depending on their granularity: *upper (or top-level) ontologies*, *domain ontologies*, *task ontologies* and *application ontologies* as suggested by Guarino 1998 [92]. Figure 3.4 presents the relations between the 4 classes of ontologies. We see here granularity as the measure for assessing the generalization/specialization of the concepts.

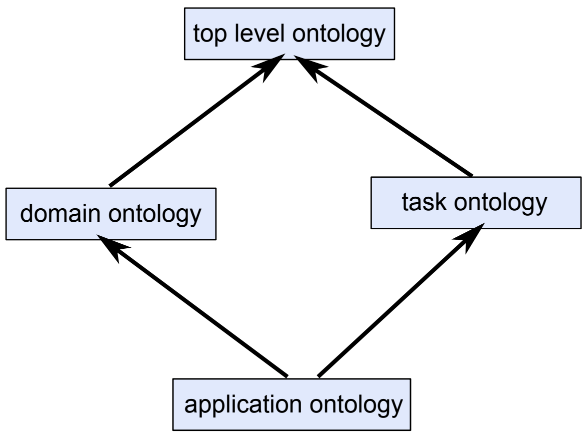


FIGURE 3.4: Classes of ontologies according to their level of dependence on a particular task or point of view [91].

Top-level ontologies define general concepts – as space, time, matter – which are independent of a particular problem or domain. *Domain ontologies* and *task ontologies* respectively describe the vocabulary of a certain domain, or a certain task. The former are composed of domain concepts and the relations that connect the concepts, while the latter describes how to solve and/or to manage a given task. The relation of generalization/specialization in the top of Figure 3.4 is explained by the fact that domain and task ontologies detail the description of concepts used in top level ontologies.

Application ontologies are specialized from domain and task ontologies because they define concepts from these two types of ontologies. Usually, the concepts correspond to roles of domain entities in executing an activity.

3.4 Semantic Web Languages – Toward the Formalization of Ontologies

In the last decade, information over the Web grew more and more making the Web information search nearly impossible. For the first time, the *Semantic Web* was introduced in [24] by the W3C[7] founder, Tim Berners-Lee, and discussed along multiple papers as [25]. Semantic Web proposes a *machine-usable content* Web [209], defined by Berners-Lee [24, 25] as *an extension of the current Web in which information is given well-defined meaning, better enabling computers and people to work in cooperation.* This definition is discussed in the Econtent Magazine and Trippe 2001 [205], suggested that *something has semantics when it can be processed and understood by a computer, such as how a bill can be processed by a software package.* This definition can be explained as follows: the Semantic Web proposes to extend the traditional Web with well-structured annotations which has as main aim to give to the Web a content that can be understood both by humans and by machines. In order to determine the meaning of a collection of documents over the Web, it is necessary to formalize the representation of these documents.

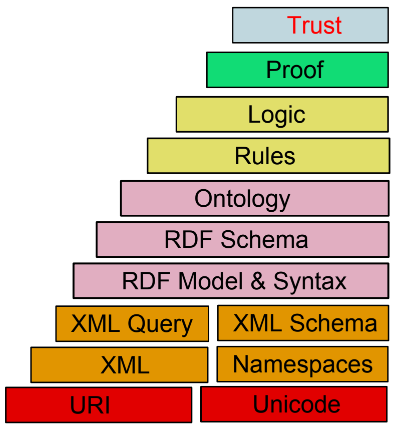


FIGURE 3.5: Semantic Web Layer Cake: enabling standards and technologies for the Semantic Web (adapted after Tim Berners-Lee, W3C).

In the last decade, W3C worked in order to propose formalisms for meta-data representation over the Web. An hierarchy – layer cake – of these representation languages/models is presented in Figure 3.5 and the main languages are detailed below.

We should note that the evolution of the Semantic Web layer cake is comparable with the evolution of knowledge representation formalisms (see Figure 3.6). In this paradigm, *Semantic Networks* [188] were proposed in early 70s as a model for relational references including conceptualization. In the same line of Knowledge Representation formalism, *Frames* [90] were introduced in 1993 as a model described using *classes*, *slots*, *facets* and *instances*. On the top of the stack, *Description Logic*

[7] http://www.w3.org/

– *DL* [14] was developed as an extension of the first two ones incorporating formal logic-based semantics and proposing an inference process.

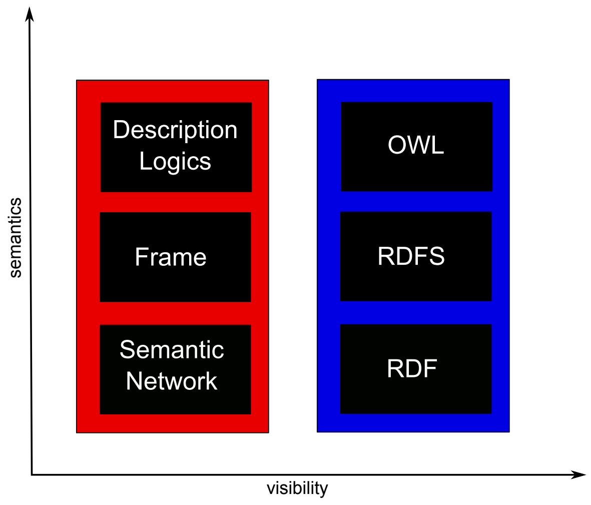


FIGURE 3.6: Comparison between the evolutions of Knowledge Representation formalisms and Semantic Web languages (adapted after [57]).

A similar evolution of representation languages is present in the field of Semantic Web. Thus, starting with *eXtensible Markup Language – XML* and *RDF* the semantic is introduced over the web. *RDFS* completed later *RDF*, and *OWL* extends RDF and RDFS in order to propose an accurate and flexible language for ontology representation.

3.4.1 XML Language and RDF Structure and Syntaxes

The most primitive representation language developed in Semantic Web is XML [212]. Suitable for transmission via networks, it is flexible and it allows users to easily insert annotations in a document. However, it does not allow users to describe semantic information in a large way. In spite of this, XML language is very present in this area and it has been chosen as the basic language for other languages proposed in the Semantic Web. The notion *eXtensible* from the large appellation of XML denotes its most important characteristic: XML is defined as a *metalanguage* – it permits to represent other languages in a standardized way [121].

Example 3.4.1 In Table 3.1 we present an example of an XML document. We can note that *fruits*, *fruit* and *from* are elements of the XML document, and *name* is an attribute of the element *fruit*.

XML is only a syntax for data structures representation, and the semantics of represented data is not available. In order to specify the vocabulary used, *Document Type Definitions – DTDs* and *XML Schemas* were proposed. Thus, in our case, a DTD can contain the vocabulary used in the XML document: *fruits*, *fruit*, *type* and *from*, but also it can impose the existence of an attribute *name* for the element *fruit*.

TABLE 3.1: Example of an XML document.

```
< ?xml version=”1.0” ? >
< fruits>
    Fruits in supermarket
    < fruit name = ”Grape” >
        < type > White < / type >
        < from > France < / from >
    < / fruit >
< / fruits >
```

Using XML language, information is represented over the Web with an implicit semantics – elements are easily understandable by humans, and, in this context, they can be easily shared. However, there is a great disadvantage of implicit semantics: the elements are not clearly defined and they are ambiguous. Consequently, it will be very useful to be able to semantically connect web pages and to develop a semantics of page knowledge in oder to help the web research tools.

In order to fulfill the drawback of XML, W3C introduced *Resource Description Framework – RDF* [123] as the first language semantic oriented that describes resources over the Web. RDF has a very simple data model that can be summed up as follows: everything is a resource that is connected with other resources via properties [214]. A resource is *anything that is identifiable by a Uniform Resource Identifier (URI) reference* [140]. Properties are also defined as resources, but they are used in order to define relations between resources.

The fundamental composition of all expressions in RDF is a collection of triplets $< subjet, property, objet >$ in which the elements can be URIs, literals or variables. The RDF triplet can be viewed as a statement, and moreover, as presented in Figure 3.7, RDF statements can be gathered in a *directed labeled graph (DLG)* with the subjects and the objects as nodes, and the properties as edges connecting subjects to objects.

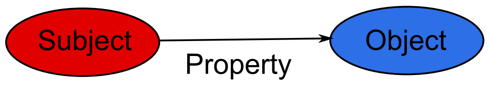


FIGURE 3.7: Graph model of an RDF statement (triplet) (adapted after [123]).

Example 3.4.2 The Table 3.2 presents an RDF document which is represented by a directed labeled graph in Figure 3.8. This example consists of a list of Fruits in a supermarket. We have several fruits characterized by a name, a type, and the origin.

TABLE 3.2: Example of an RDF document.

```
< ?xml version=”1.0”? >
< rdf:RDF xmlns:rdf=”http://www.w3.org/1999/02/22-rdf-syntax-ns#”
            xmlns:list=”http://www.w3.org/list#” >
   < rdf:Description rdf:about=”http://www.w3.org/list#Fruits >
      < list:Text > Fruits in supermarket < / list:Text >
         < rdf:Description rdf:about=”http://www.w3.org/list#Fruit/ >
            < list:Name > Grape < / list:Name >
            < list:Type > White < / list:Type >
            < list:from > France < / list:From >
         < / rdf:Description >
   < / rdf:Description >
< / rdf:RDF >
```

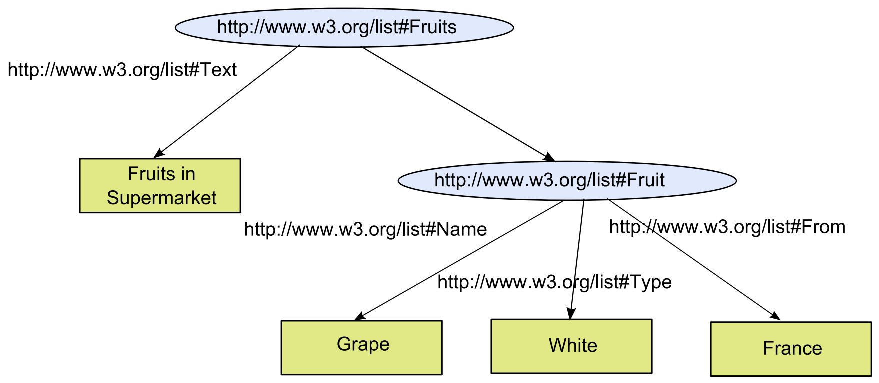

http://www.w3.org/list#Fruits
http://www.w3.org/list#Text
Fruits in Supermarket
http://www.w3.org/list#Fruit
http://www.w3.org/list#Name
http://www.w3.org/list#From
http://www.w3.org/list#Type
Grape
White
France

FIGURE 3.8: RDF graph for the XML document in Table 3.2.

RDQL [179], SeRQL [40], and SPARQL [171] are the main query languages proposed in the literature for the RDF language; nevertheless SPARQL is currently the most widely used, being defined by W3C as a standard. SPARQL is based on the notion of RDF triple patterns and its semantics is based on matching triples with RDF graphs.

3.4.2 RDFS and OWL meta-ontologies

It is important to note that RDF is designed to provide a basic object-attribute-value data model for metadata. Other than the intentional semantics described only informally in the standard RDF makes no data-modeling commitments [54]. Thus, RDF data model does not provide mechanisms for defining the relationships between properties (attributes) and resources. Just as XML Schema (DTD) provides a vocabulary definition facility, RDF Schema permits to developers to define a particular vocabulary for RDF data and specify the object to which these attributes can be applied [35]. In other words, RDF Schema (RDFS) is extending RDF vocabulary proposing to organize classes in taxonomies using *subClassOf* property, and, in the same idea, *subPropertyOf* permits to create taxonomies of properties. The second major improvement brought by RDFS is the possibility of defining domain and range for a property, i.e. the class to which the property is applied, and the class of values of the property. Last, but not least, RDFS proposes several relationships between properties enriching the representation language.

Unfortunately, RDF does not offer background for axiom representation and reasoning with axioms. In order to fulfill this drawback, several improvements on RDFS were proposed to express axioms as rules and to define classes and properties using other classes et/or properties [55].For this purpose, the language *Ontology Interchange Language – OIL* [66, 67] was developed. OIL defines new language primitives using description logics methods, and to define more complex ontology structures.

Later, OIL merged with DAML language in order to create DAML+OIL[8]. In this context, DAML - DARPA Agent Markup Language [102] became the first language proposing a basic primitives for simple inferences and simple requests. For example, let us consider the following two phrases *Parenthood is a more general relationship than motherhood* and *Mary is the mother of Bill.* DAML makes possible the extraction of new facts by inference as *Mary is the parent of Bill.* Moreover, a query system is able to answer questions such as *Who are Bill's parents?*, even if all the necessary facts are not stated effectively.

In the evolution of ontology languages, RDF/RDFS joined the last generated language, DAML+OIL, forming the new standard for knowledge representation over the web – OWL. OWL was introduced as a proposition for an accurate and flexible representation language for ontologies; it is based on the RDF language – the class- and property-structure of RDFS – and it improves it from different points of view. First, we will stop on the features common to both OWL and RDF:

- OWL integrates the class declaration and organization using the subsumption relation;
- OWL declares properties between domain and range;
- OWL organizes properties into hierarchies.

In addition to RDF/RDFS capabilities, OWL brings essential improvements:

[8]http://www.w3.org/TR/daml+oil-reference

- OWL can describe classes as logical combinations (intersections, unions, or complements) of other classes, or as enumerations of specified objects;
- OWL is able to state that a property is transitive, symmetric, functional, or is the inverse of another property;
- OWL can express which individuals belong to which classes, and what the property values are for specific individuals;
- OWL can provide restrictions on how properties behave locally on a specific class. This improvement was inherited from Description Logic (which is discussed in the following Section) and is one of the most important improvements of OWL. The feature offers the possibility to define classes where particular properties are restricted; thus, only the individuals satisfying the set of restriction are individuals of the class;
- OWL proposes to infer information in ontology with the main purpose to generate interesting new knowledge.

OWL syntax is based on the RDF one, and the two important OWL constructors are specializations of the corresponding RDF constructors – class and property constructors – *owl: Class* and, *owl: DataProperty* or *owl: ObjectProperty*. Figure 3.9 presents the relation between the main constructors in OWL and RDF.

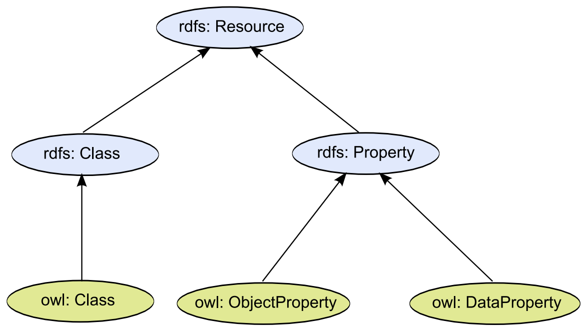


FIGURE 3.9: OWL constructors inherited from RDF constructors (adapted after [10]).

The OWL language provides three increasing expressive sublanguages [10]:

- *OWL Lite* is a very simple model. It is proposed for the organization of concepts and properties in hierarchies, but it permits also to define certain restrictions over properties. Due to the limitation of restrictions used in this sublanguage, the computability time of inference processes is limited..
- *OWL DL (Description Logic)* provides an important expressiveness, and, at the same time, it restricts the utilization of OWL and RDF constructors in order to ensure decidability, computational completeness and useful reasoning methods. It is based on Description Logics and it includes all OWL language constructors based on restrictions.

- *OWL Full* – The main advantage of OWL Full is that it provides no limits in expressiveness. For example, the type restriction is not as strict as in OWL DL; in OWL Full a defined class can be treated both as a collection of different individuals and as an individual. The great inconvenient of OWL Full is that it is so powerful in expressiveness that it became undecidable.

The differences regarding the features integrated in each of three languages are presented in Table 3.3.

Table 3.3: Differences between OWL Lite, OWL DL and OWL Full.

OWL Lite	OWL DL	OWL Full
(sub)classes, individuals		
(sub)properties, domain, range		
conjunction	negation	
(in)equality	disjunction	meta-classes
cardinality 0/1	full cardinality	modify language
datatypes	enumerated types	
inverse, transitive	hasValue	
symmetric properties		
someValuesFrom		
allValuesFrom		

3.4.3 Descriptions Logic and OWL

The aim of the knowledge representation research field was to find a suitable language to describe the world – or an application domain – that can be properly used in order to create intelligent applications. In this purpose, Description Logic (previously called *terminological logic*) was introduced and it represents a family of class-based (concept-based) knowledge representation formalisms [14]. Compared to its predecessors, its particularity comes directly from its name – *Logic* – which outlines that this set of KRs are based on a formal and logic semantic. That is the reason why it can be considered as a subset of *First-Order Logic (FOL)* [32] with a decidable inferencing process.

Generally, a Description Logic knowledge base is composed of two components:

- a *terminological box – TBox* – containing the terminology (the vocabulary) of the application domain. The vocabulary is composed of two important elements:
  - the *concepts* – corresponding to classes in OWL and represent sets of individuals (i.e. $Fuits$ or $Meat$);

  - the *roles* – corresponding to properties in OWL and represent binary relations between concepts (i.e. *isEcological* or *isDiet*).

- an *assertional box* – *ABox* – containing assertions about individuals (instances) of the concepts – $C(a)$ – or about roles – $R(a, b)$ –, where C is a concept, R a role (relationship or property), and, a and b individuals (i.e. $pear(white\_grape)$ or $isEcological(white\_grape, true)$).

Table 3.4: Acronym meaning – Description Logic expressiveness.

Symbol	Meaning
$\mathcal{ALC}$	$\mathcal{AL}$ with Complement
$\mathcal{S}$	$\mathcal{ALC}$ with transitive properties
$\mathcal{H}$	Property hierarchy (subproperties – *rdf:subPropertyOf*)
$\mathcal{O}$	Nominals – enumerated classes of object value restrictions (i.e. *owl:oneOf*, *owl:hasValue*)
$\mathcal{I}$	Inverse properties
$\mathcal{N}$	Cardinality restrictions (*owl:Cardinality*, *owl:MaxCardinality*)
$\mathcal{Q}$	Qualified property restrictions
$(\mathcal{D})$	Data types
$\mathcal{F}$	Functional properties

The $\mathcal{AL}$-*language* (attributive language) is the minimal description language. In the $\mathcal{AL}$-*language* the following concept descriptions are available: atomic concept, universal concept, bottom concept, atomic negation, intersection, value restriction, and limited existential quantification. Each new Description Logic is based on $\mathcal{AL}$-*language* and it enriches the terminology by adding to atomic concepts and atomic roles *concept constructors* in order to create new concepts.

Nevertheless, a great part of research in this field deals with the improvement of expressiveness and computational properties. Thus, the expressive power of the DL affects directly the decidability and the complexity of the inference process. It is obvious that we find two cases: first, if the DL is very expressive the probability for a undecidable inference problem is very high; second, less (minimal) expressive DLs have decidable inference process, but they are limited in expressiveness [105].

The design of OWL was significantly influenced by Description Logics, and particularly by the choice of language constructors. In fact, each OWL language is based on different expressive Description Logics, and the difference comes from the intention to create three OWL languages with three different types of expressiveness and decidability.

OWL DL and OWL Lite are based on the expressive Description Logic $\mathcal{SHOIN}(\mathcal{D})$

Table 3.5: List of OWL DL constructors.

Classes	
A	
intersectionOf($C_1 \dots C_n$)	$C_1 \sqcap \dots \sqcap C_n$
unionOf($C_1 \dots C_n$)	$C_1 \sqcup \dots \sqcup C_n$
complementOf(C)	$\neg C$
oneOf($o_1 \dots o_n$)	$\{o_1\} \sqcup \dots \sqcup \{o_n\}$
restriction R	
someValuesFrom(C)	$\exists R.C$
allValuesFrom(C)	$\forall R.C$
hasValue(o)	$R : o$
minCardinality(n)	$\geq n\ R$
maxCardinality(n)	$\leq n\ R$
cardinality(n)	$=\ n\ R$
restriction T	
someValuesFrom(D)	$\exists T.D$
allValuesFrom(D)	$\forall T.D$
hasValue(o)	$T : o$
minCardinality(n)	$\geq n\ T$
maxCardinality(n)	$\leq n\ T$
cardinality(n)	$=\ n\ T$
Data Range	
B	
oneOf($v_1 \dots v_n$)	$\{v_1\} \sqcup \dots \sqcup \{v_n\}$

and $\mathcal{SHIF}(\mathcal{D})$ respectively (see Table 3.4 for acronym meaning). OWL DL and OWL Lite provide the expressiveness of two powerful DLs, but OWL Lite limits the complexity: comparing to OWL DL, OWL Lite does not permit enumerations, individuals to occur in description or class axioms, and cardinality restrictions are limited to 0 and 1. The main advantage is that, with the loss in expressiveness power, OWL Lite gains in tractability. OWL Full is based on OWL DL, but it overpasses it. Unfortunately, the inference process is undecidable, and moreover, the abstract syntax of OWL DL is inadequate, and the RDF/XML syntax should be used.

In Table 3.5 we present the main constructors integrated in OWL DL, where A is a class name, C is a description, o is an individual, R is an object property, T is a data property, B is a data type, D is a data range, v is a value and n is a non-negative integer. In the first column, constructors are expressed in OWL abstract syntax, while in the second column, they are presented in Description Logic syntax.

In the following, we will consider restriction constructors and we will show their interest in OWL ontologies. An ontology concept is denoted as a restriction concept (*defined concept* in OWL language) if it is defined using necessary and sufficient conditions based on at least a restriction constructor. A restriction constructor describes a constraint on relationships that the individuals participates in for a given property, and it has as consequence the limitation of individuals belonging to a concept.

Three types of restrictions are available as follows [104]:

- **Quantifier Restrictions** – propose, as their name denotes, a quantification of the individuals composing the class. This type of restrictions is composed of a *quantifier*, a *property*, and a *filler*. Two types of quantifier restrictions may be used:

  - The *existential quantifier* – *someValuesFrom* – (or ($\exists$) in DL), can be read as *at least one*, or *some*;
  - The *universal quantifier* – *allValuesFrom* – (or ($\forall$) in DL), which can be read as *only*.

  There is an important difference between these two quantifier restrictions which is generally difficult to understand by novice users. The existential quantifier is employed to denote that a class contains those individuals that have at least one relationship along a certain property to an individual that is a member of a second class – the filler. On the contrary, the universal quantifier unifies all those individuals that have only relationships along a certain property to individuals from a well defined class. In other words, individuals from classes defined by the existential quantifier *can* have relationships along *the same* property to individual from *another* class, while universal quantifier does not allow it.

  Example 3.4.3 To exemplify the quantifier restrictions and to present the differences between them, we will consider an ontology with:

  - a *FoodItem* class containing all food products sold in a supermarket (as individuals), and
  - the *Country* class with two sub classes: *WesternEurope* and *EasternEurope* regrouping as individuals all countries in west and, respectively, in east of the Europe;
  - *isFromCountry* property defines the country(s) each product comes from.

  If we want to define a new class – *EasternEuropeProducts* – containing the products produced in Eastern Europe countries we can use one of the two quantifiers in function of the semantic that we want to express.

  The structure of this ontology is drawn in Figure 3.10.

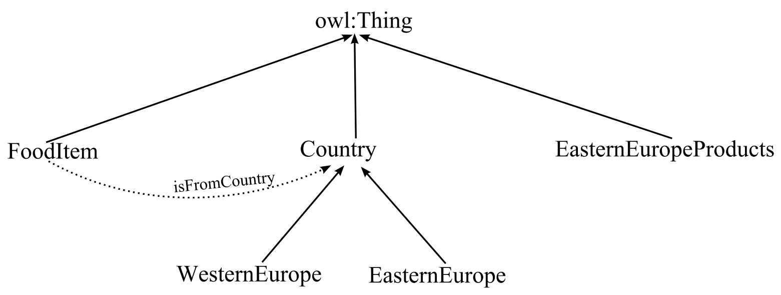


FIGURE 3.10: The ontology in 3.4.3 example.

Figure 3.11 presents the difference of using the two techniques. The upper part presents the use of existential restriction:

$EasternEuropeProducts \equiv$

$FoodItem \sqcap isFromCountry$ someValuesFrom $EasternEurope$

or

$FoodItem \sqcap \exists\, isFromCountry.EasternEurope$

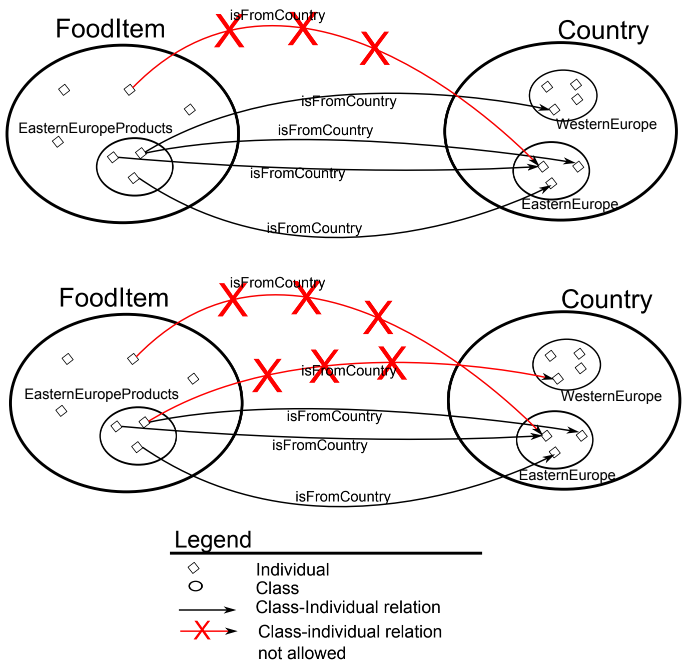


FIGURE 3.11: A semantic of using quantifiers. Upper figure presents the semantic of the Existential Restriction – ∃ *isFromCountry.EasternEurope*, while the bottom one presents a Universal Restriction – ∀ *isFromCountry.EasternEurope*.

That is to say that we consider that the $EasternEuropeProducts$ class contains those food items that have as provenance country at least one Eastern Europe country. This interpretation could help in cases when the same product could be imported from different countries. For instance, in France, strawberries are produced in France, but we import also strawberries from Spain. The particularity of this interpretation and of the existential restriction use is that individuals of *EasternEuropeProducts* are not limited to have as provenance a country from the East of Europe. It does not exist a product coming from Eastern Europe and which is not in this class.

The other part presents the use of universal restriction:

$EasternEuropeProducts \equiv$

$FoodItem \sqcap isFromCountry$ allValuesFrom $EasternEurope$

or

$FoodItem \sqcap \forall\, isFromCountry.EasternEurope$

The difference between this interpretation and the last one, is that, in this case, the single provenance possible for the products is only Eastern Europe countries.

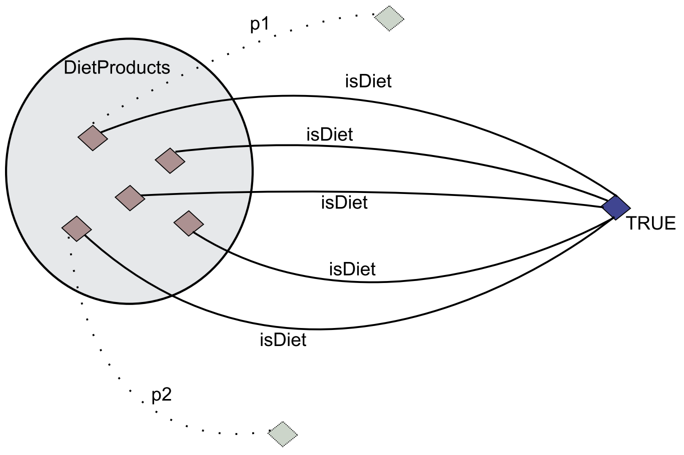


FIGURE 3.12: Graphical representation of restriction concept based on *hasValue* restrictions (adapted from [104]).

- **hasValue Restrictions** (∋) describe a class of individuals that are related to another specific individual along a specified property. The difference with the quantifier restriction is that the individuals that are described by the quantifier restriction are related to any individual from a specified class along a specified property.

  Example 3.4.4 We will base our example on the scenario given in the previous example. Let us consider that we integrate one data property in our ontology – *Fooditem.isDiet()* – a boolean property which denotes if a product is a dietetic one or not.

  Based on this data property, we are able to define a concept describing the individuals related to the boolean data *TRUE* by the *isDiet* property:

$$DietProducts \equiv FoodItems \sqcap isDiet\ hasValue\ TRUE$$

In Figure 3.12 we present graphically the restriction concept *DietProducts*. The difference between quantifiers and *hasValue* restrictions is that the latter limits the individuals of a class to those individuals that are connected along a given property to a given *individual*. It is not forbidden for the individuals to be connected to other individuals along the same property.

- **Cardinality Restrictions** are used to test the number of relationships that an individual may participate in for a given property and with different values/objects.

3.4.4 Query Languages and Inference Engines

To access the asserted and inferred facts represented in Semantic Web languages, a common, well-defined interface is needed. This is accomplished through standardized query languages, which have the advantage of a higher abstraction level.

Basically, query languages can be categorized by the storage format of the ontology on which the queries can perform [81]. As query languages are used to access ontolgies, they are based on the OWL, RDFS or DAML/OIL format. The ontology query language syntax is derived from other popular query languages, as SQL, or functional and logic programming languages, as LISP. Depending on the language, the expressive power of the queries may differ.

Although SPARQL achieves to define a standard in composing and submitting conjunctive queries to RDF documents, it does not take in charge the inference capabilities, taxonomic queries and the particularities of the OWL language. Unfortunately, it is quite difficult to provide a semantics for these query languages under OWL-DL semantics because RDF representation mixes the syntax of the language with its assertions.

In order to fulfill these drawbacks, SPARQ-DL was introduced by Sirin and Persia [186]; it is defined as a substantial subset of SPARQL, which is designed for querying certain OWL-DL based semantics. SPARQL-DL has the ability to combine queries about the schema (classes and properties) and the data (individuals) that brings new challenges to query answering. Moreover, one of its grate advantages is that it can be implemented on top of existing OWL-DL reasoners (like Pellet).

As we could see in the previous section, the description logic terminology and assertion boxes can be significantly enriched by constructors and, we can quickly arrive to inconsistent structures.

Example 3.4.5 For instance, we will consider the previous example presented in Figure 3.11 and, more precisely, the case of the universal restriction. For this example, we rename the defined concept in *NewProducts*. Thus, the *NewProducts* concept is defined as following:

$$NewProducts = FoodItem \sqcap \forall\, isFromCountry.EasternEurope.$$

Next, we will consider that we add to this definition the $\forall$ *isFromCountry. WesternEurope* restriction, thus, the definitive definition is:

$$\begin{aligned} NewProducts \equiv\ & FoodItem \\ & \sqcap \forall\ isFromCountry.EasternEurope \\ & \sqcap \forall\ isFromCountry.WesternEurope. \end{aligned}$$

Now, the concept *NewProducts* regroups those products only from Eastern Europe countries, and, at the same time, only from Western Europe countries. It is obvious that it is not possible for a product to come *only from two countries*. Thus, the definition that we proposed is inconsistent. ■

In order to easily find these types of errors, reasoners over the description logic were proposed. They attempt to find inconsistencies such as:

- *Subsumption* – to check whether a concept is more general than another: $C \sqsubseteq D$;
- *Equivalence* – to check whether two concepts are equivalent: $C \equiv D$;
- *Instantiation (membership)* – to check weather an individual i is a member of a concept C: $i \in C$;
- *Correctness* – to capture intentions of domain experts;
- *Minimal redundancy* – to capture unintended synonymous.

Furthermore, reasoners infer new information by reorganizing the concepts taxonomy and by defining the real relationships between individuals and concepts. For instance, let us consider that *pear* is the instance of the *FoodItem* concept, and that *isFromCountry(pear, Romania)* – *pear* products come from *Romania*, where *Romania* is an instance of *EasternEurope* concept (Romania is a country in the East of Europe). If we consider *EasternEuropeProducts* defined by a universal quantifier, after the inference, the *pear* individual will also be an instance of the *EasternEuropeProducts* concept because the former has all its *isFromCountry* relationships to instances of *EasternEurope* concept.

As a global consequence, a description logic-based system can be described by the architecture in Figure 3.13. The latter is composed of several levels: first, description logic defines application domain terminology and its concrete elements (individuals); second, inference engine verify the consistency of the proposed system, and also infer new information, and in the last, a user interface.

In Semantic Web, OWL DL and OWL Lite languages benefit of all the advantages of Description Logic among which DIG interface for DL reasoners is one of the most important. Thus, the inference engines developed for the Semantic Web languages were designed starting form DL reasoners.

Nowadays, a great number of inference engines free or commercial exist with the main aim to extract new knowledge such as *Racer* [96], *Pellet* [158], *Fact* [107], *Fact++* [206], *Surnia*[9], *F-OWL*[10] and *Hoolet*[11]. A great part of these engines were

[9]Surnia web site: http://www.w3.org/2003/08/surnia/
[10]F-OWL web site: http://fowl.sourceforge.net/
[11]Hoolet web site: http://owl.man.ac.uk/hoolet/

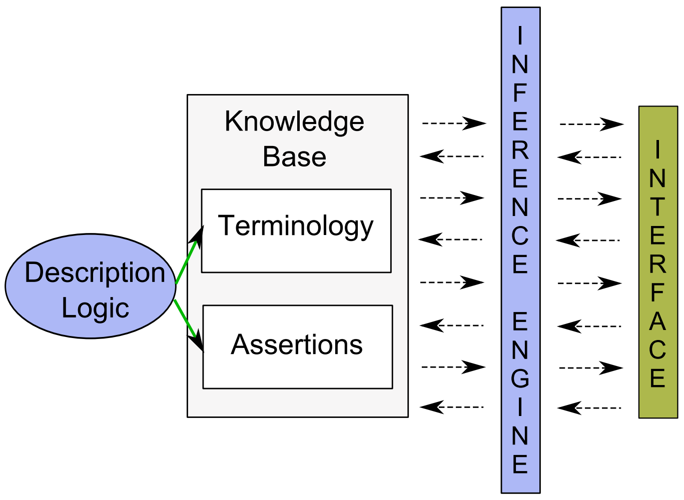


FIGURE 3.13: Architecture of a description logic system [106].

designed to reason over description logics, but they were also adapted to OWL ontologies. F-OWL, Hoolet and Surnia are based on experimental reasoning methods which point out interesting performances for simple problems; unfortunately, these three reasoners are not suitable for important projects due to their inefficacy and to the undecidability of the integrated algorithms. Thus, in this section we will concentrate on the rest of inference engines: Fact, Fact++, Racer, and Pellet. Table 3.6 presents a comparative study between these four reasoners.

TABLE 3.6: Comparative study between the main inference engines [69].

	Racer	Pellet	Fact	Fact++
Description Logic	$\mathcal{SHIQ}, \mathcal{SHF}$	$\mathcal{SHIN}(\mathcal{D})$ $\mathcal{SHON}(\mathcal{D})$	$\mathcal{SHOQ}(\mathcal{D})$	$\mathcal{SHIF}(\mathcal{D})$
Implementation	C++	Java	Common Lisp	C++
Inference	TBox/ABox	TBox/ABox	TBox	TBox
API Java	yes	native	yes	yes
OWL	OWL DL	OWL DL	OWL DL	OWL Lite
DIG Interface	yes	yes	yes	yes
Direct Connection	non	yes	non	non

We can remark that all these engines reason over TBox and ABox, apart Fact and Fact++ engines which are specialized in reasoning only over TBox. That is to say that Fact and Fact++ (the new C++ version of Fact) do not reason over individuals, but only over the terminology box. This is an important inconvenient for a reasoner, knowing that a great part of applications is based on ABox reasoning.

Further, we can note that a great part of reasoners is based on the DIG interface. DIG interface is a standard interface/protocol that was suggested in order to propose a common interface to description logic reasoners for these ones to be accessed by different applications. But, invoking reasoners through a DIG interface has important limitations – ontologies suffer different modifications when being sent to the reasoner. Pellet is the only reasoner which integrates a direct connection, and, moreover, it is the native reasoner in Jena[12]. In this context, Pellet is the more appropriate reasoner.

3.5 Conclusion

In this Chapter, we presented different formalisms for knowledge representation and we stopped over the Semantic Web languages: XML, RDF and OWL. In a first part, we studied the development of representation formalisms from controlled vocabularies to ontologies. Next, we discussed the advantages that ontologies provide as knowledge representation formalism starting with classical structure – concept, properties and axioms –, and ending with inference engines.

Web Ontology Language (OWL) was the center of the second part of this Chapter. We outlined the advantages of using this Semantic Web language, how description logic guides it, and the application of inference engines over the OWL knowledge. We concluded with a brief study over the available inference engines for OWL.

[12]Jena web site: http://jena.sourceforge.net/

4
From Data-Based to Knowledge-Based Measures

Can we preserve the full power of association rule mining without overwhelming the user?

Pruning and summarizing the discovered associations
Liu et *al.*

Contents

4.1 Introduction

In the Chapter 2, we proposed a brief description of the KDD field, and more particularly, we detailed the association rule mining technique. The two main problems outlined during this chapter were: the lack of rapidity and of efficiency of mining algorithms. The former deals with the exponential characteristic of execution time of mining algorithms, and the latter with the useless of discovered rules due to their huge number which makes impossible their analyze by a user.

In this context, interestingness measures evaluate the relevance and the interest of the discovered rules. Interestingness measures were classified by Piatetsky-Shapiro and Matheus 1994 [166] in *objective* and *subjective*. Objective or data-driven measures assess discovered rules from a data (statistical or descriptive) point of view. They have a main advantage of being autonomic and of generating a satisfactory result without user implication. Nevertheless, these measures could not ensure an important reduction of rule number, and moreover, the quality of rules for different users varies. In other words, a user can find a set of rules interesting, while another user could find it trivial. This Chapter states about the interest of objective measures and it makes a brief description and presentation of their advantages/inconvenience.

Subjective or user-driven measures attempt to select those rules corresponding to a given user. In this context, it is important to find the right representation model for the background knowledge of each user. In Chapter 3, we studied different formalisms for knowledge representation. In the second part of this Chapter we make a survey over the approaches using those knowledge representation formalisms to filter the interesting rules.

The Chapter concludes with a comparative study between post-processing and measure-embedded mining techniques.

4.2 Interestingness Measure Evolution

In the first years of their apparition, knowledge discovery techniques were considered as simple processes in the way that all extracted patterns were presented to the user. Gradually, these processes evolved and important methods came up for pattern selection. In Figure 4.1, we compare three important categories of pattern discovery techniques (here, we are not interested in the specificity of data mining techniques applied over data, so the data mining process is treated as a closed black box).

First, in Figure 4.1–a) we can observe a graphical representation of the most simple process of knowledge discovery. Its principal advantage is that it generates a complete set of patterns satisfying the given conditions which are generally very simple. Nevertheless, this advantage becomes its main drawbacks – the important volume of patterns presented to the user, and, the time consumed in a such generation process. Therefore, the selection of interesting patterns became a major problem in KDD field and interestingness measures were proposed to assess the quality of patterns.

Figures 4.1–b)/c) outline two techniques to integrate interestingness measures.

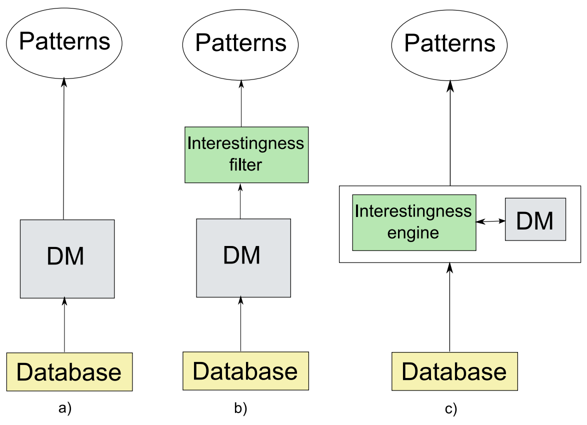


FIGURE 4.1: Different techniques for the KDD Process (proposed in [183] and revised later in [143]).

The first technique is the application of interestingness measures in a phase of post-processing of discovered knowledge (Figure 4.1–b)), and only a selection of discovered patterns is presented to the user. This solution is quite easy to develop, but not efficient due to the important execution time of the pattern generation phase. The second solution is the application of interestingness measures during the pattern generation phase (Figure 4.1–c)). This technique is more interesting than the former, being more efficient because the patterns that are not interesting are pruned early enough. Nevertheless, this technique is difficult to conceive.

In 2006, Geng and Hamilton [83] introduced a new description for the knowledge discovery process (Figure 4.2) taking into account three possible roles for the interestingness measures:

- metrics can be used *to prune uninteresting patterns* during the mining process – minimizing the search space and improving the efficiency;
- metrics can be used *to rank patterns* after the mining process;
- metrics can *filter the patterns* in the post-processing task.

Section 4.6 offers a complete comparative study between the utilization of metrics as post-processing techniques or directly in the mining algorithms.

In the last decade, interestingness measure field had an important activity and, more particularly, in the association rule mining field. A hight number of measures was proposed in the literature, and for the first time in 1994, they were classified by Piatetsky-Shapiro and Matheus [166] as follows:

- *objective (or data-oriented) measures* are metrics depending on data;
- *subjective (or user-oriented) measures* take into account the user goals and beliefs.

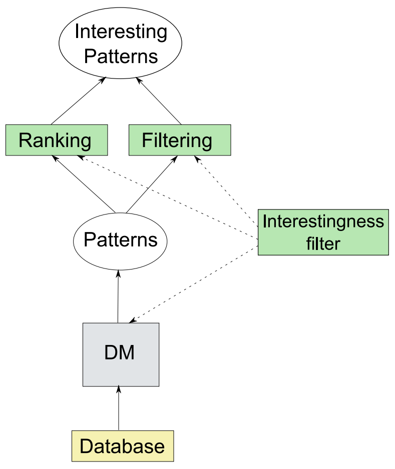


FIGURE 4.2: Roles of interestingness measures in the KDD process [83].

This classification was refined by Silberchatz and Tuzilin 1995 [183] who introduced a first classification (discussed in Section 4.4) of subjective measures in *actionability* and *unexpectedness*.

Later, in 2006, Geng and Hamilton [83] proposed a new classification, taking into consideration the researches done in the field and the proposed interestingness measures:

- Objective Measures:
  - Based on probability – these measures evaluate the generality and the reliability of a rule and generally, they are defined starting from a 2×2 contingency table;
  - Based on the form of the rules – these measures evaluate a rule comparing to its neighborhood; the more the rule is different from its neighborhood, the more it is considered interesting (i.e. peculiarity, surprisingness and conciseness).
- Subjective Measures – comparing to objective measures, these measures are not based on database properties, but on user goals and beliefs. They are classified in:
  - Surprisingness (unexpectedness);
  - Novelty and/or actionability.
- Semantic Measures – domain-based measures:
  - Utility – generally, this type of measure is based on the notion of weighted association rule mining [44];
  - Actionability - this type of measures is business-oriented and assesses the help that the mining result can bring to the user in taking decisions.

Surely, this new classification is more precise than the first one. Nevertheless, the measure *categories* are not disjoint and it is easy to confuse one with the other. For instance, let us consider the case of the *Utility* category. Speaking of the first approach proposed, weighted association rule mining, the action of weighting the patterns or the transactions is made by the user. Thus, we can consider that this is a subjective measure and not a semantic one. In the same way, we can view actionability measures as user or business oriented, so, in some cases, it should be more appropriate to consider actionability as a subjective measure.

4.3 Objective Interestingness Measures

Objective measures are defined as metrics capturing dependencies among variables in a dataset with the main goal to assess the association rules. Their principal characteristics are as follows:

- they are domain-independent – they do not assess the rules by using information from the domain field;
- they are partial user-dependent – understanding by this that the user is requested to give threshold levels for the measures, but he/she is not interactively involved in the mining process;
- they are fully data-dependent – giving the interestingness in terms of statistics or information theory applied over the database.

4.3.1 Presentation

An objective measure is generally computed on the contingency table of itemsets frequency over the database, D. In Table 4.1 we show an example of contingency table for the X, Y binary variables, $X, Y \subseteq I$ and $X \cap Y = \emptyset$, where I is the set of attributes in the database. Let us note with N_X and N_Y the number of transactions verifying X and, respectively, Y over the total set of N transactions. In this context, we can define $\bar{X}$ as the negation of X; in other words, $\bar{X}$ verifies the complementary set of transactions verified by X, thus it verifies $N_{\bar{X}} = N - N_X$ transactions. Now, we can define the frequency of X as: $P(X) = \frac{N_X}{N}$.

TABLE 4.1: Contingency table for X and Y binary variables

	Y	$\bar{Y}$	
X	$N_{X,Y}$	$N_{X,\bar{Y}}$	N_X
$\bar{X}$	$N_{\bar{X},Y}$	$N_{\bar{X},\bar{Y}}$	$N_{\bar{X}}$
	N_Y	$N_{\bar{Y}}$	N

In this context, we define the *examples* of the rule as the set of transactions verifying XY, i.e. $N_{X,Y}$, and the *counterexamples* of the rule as the set of transactions verifying X but not Y, thus verifying $X\bar{Y}$, i.e. $N_{X,\bar{Y}}$. The more a rule has fewer counterexample and numerous examples, the more it is interesting.

Proposed with the introduction of association rules, support and confidence [5] are the most basic and widely used interestingness measures. *Support* evaluates the rule from the generality point of view and it is equal to the frequency of the two itemsets (antecedent and consequent) in the data:

$$support(X \rightarrow Y) = \frac{N_{X,Y}}{N}.$$

Example 4.3.1 Let us take the example of supermarket basket. The following association rule *apples* → *milk* is discovered by the mining process and its support is 52%. That is to say that 52% of baskets contains at the same time *apples* and *milk*. ■

The *confidence* is viewed as the strength rate of an association rule, assessing the validity of the rule. This metric is computed as the fraction between the frequency of the two itemsets and the one of the antecedent. This metric can be formalized as follows:

$$confidence(X \rightarrow Y) = \frac{N_{X,Y}}{N_X}.$$

Example 4.3.2 Let us consider that the association rule *apples* → *milk* has the confidence of 83%. That is to say that 83% of baskets containing *apples*, contain also *milk*. ■

Unfortunately, a great part of past research [19, 202] agreed on the limitation of the rule evaluation by only these two metrics. The limit can be viewed at several levels. First, the huge amount of rules limits the utility of the technique, and second, the triviality of a great part of the generated rules implies a real necessity of new metrics.

As a consequence, in the last two decades, a surprising number of interestingness measures was developed in order to fulfill the drawback of the basic metrics. Nevertheless, the problem is far from being solved because new questions have been raised: *How could someone know which are the best metrics to use in a specific situation and a particular application field?*. Or, pushing this issue further, it is reasonable to question whether the measures can produce similar results when applied to a set of association rules [193].

A great number of surveys attempt to characterize, to differentiate, to classify and to rank the interestingness measures [19, 30, 83, 95, 111, 143, 202]. In this section we will briefly resume the main propositions and we will discuss the classification of measures suggested by Blanchard and al. 2009 [30].

In 1991, Piatetsky-Shapiro proposed a first set of principles concerning the value of a metric applied over an association rule [165]. If we consider the association rule $X \rightarrow Y$, Piatetsky-Shapiro outlined that:

- the metric is equal to 0, if X and Y are statistically independent $\Rightarrow P(X,Y) = P(X)P(Y)$;
- the metric is monotonically increasing with $P(X,Y)$, when $P(X)$ and $P(Y)$ remain the same;
- the metric is monotonically decreasing with $P(Y)$.

Starting from these three principles, several new properties/classifications are suggested. In a first attempt, Hilderman and Hamilton 1991 [103] introduced five interesting principles: minimum value, maximum value, skewness, permutation invariance, transfer. Later, Tan *et al.* 2004 [202] pointed out five important properties: symmetry on variable permutation, invariance on row or column scaling, antisymmetry on row/column permutation, invariance on row and column permutation, no relationship with transaction not dealing with the rule. But, symmetric/antisymmetric properties cannot be defined for all the interestingness measures (i.e. the confidence), thus, the authors proposed to transform each asymmetric measure into a symmetric measure.

Lenca *et al.* 2004 [128] suggested five properties to evaluate an interestingness measure: the metric has a constant value if there is no counterexample; convex decrease is desirable when few counterexamples are added; increase with the total number of records; facility to fix a threshold; and facility of its comprehension. For more details on principles of interestingness measure you can refer to [83, 111].

4.3.2 Semantics-Based Classification

In the following, we will focus on the methods for classifying interestingness measures, and, more precisely we will discuss the method introduced by Blanchard et *al.* in 2009 [30]. The proposed method classifies the interestingness measures according to the *subject*, the *scope* and the *nature* of the measure. In other words, a metric is evaluated according to the notion that it measures, the elements regarding its results concern, and the type of the metric: descriptive or statistical.

Subject-based classification. According to the subject, there are two situations when the rule interestingness takes extreme values $N_{X,\bar{Y}} = min(N_X, N_{\bar{Y}})$ and $N_{X,\bar{Y}} = max(0, N_X + N_Y - N)$. Between these two cases, there exist two important situations when rules cannot be considered interesting, being non-oriented. In the following we will discuss these two specific situations: the independence and the equilibrium. In consequence, if a rule is in independence or equilibrium, it is not interesting for the mining process, thus it should be pruned. On the contrary, the deviation cases could give interesting rules. In other words, the metrics should be able to quantify the deviation from these two situations.

A rule is in the situation of *independence* if its variables, X and Y, are independent. In consequence, each variable brings no information about the other (knowing the value of one of them does not influence the distribution of the other variable) which can be formalized as follows: $P(Y\backslash X) = P(Y\backslash\bar{X}) = P(Y)$. In the case of independence, the probability of having at the same time X and Y is $P(X \cap Y) = P(X)P(Y)$

and the number of rule examples will be $N_{X,Y} = NP(X)P(Y) = \frac{N_X N_Y}{N}$. The way of deviating from the independence is to have the two variables correlated, and two types of correlations are possible: first, the two variable could be positively correlated ($P(X \cap Y) > P(X)P(Y)$), and, second, they could be negatively correlated ($P(X \cap Y) < P(X)P(Y)$)

Definition 4.3.3

An interestingness measure, called m_i, evaluates a deviation from independence if it has a constant value at the independence, v_i: $m_i(X \to Y) = v_i$. ■

On the other hand, a rule is in the situation of *equilibrium* if it has the same number of examples and counterexamples, $N_{X,Y} = N_{X,\bar{Y}} = \frac{N_X}{2}$ [29]. That is to say that, the variable X simultaneously appears with both Y and $\bar{Y}$ in the same number. The particularity of the equilibrium is that it is not defined over the two variables X and Y, but more particularly over the variable Y, knowing the presence of X. As for the independence, two ways of deviation are possible:

- X appears simultaneously with Y in more cases than with $\bar{Y}$;
- X appears simultaneously with $\bar{Y}$ in more cases than with Y.

Definition 4.3.4

An interestingness measure, called m_e, evaluates the deviation from equilibrium if it has a fixed value at the equilibrium, v_e: $m_e(X \to Y) = v_e$. ■

These two subject-based measures evaluate association rules from different and complementary points of view. For a better comprehension, a rule having a good deviation from independence can be interpreted as "When X is *true*, then Y is *more often true*". That is to say that, Y is true more than usual, when no information about X is available. Similarly, a rule having a good deviation from equilibrium can be interpreted as "When X is *true*, then Y is *very often true*".

Scope-based classification. Since their apparition, association rules have been compared to *implications* ($X \Rightarrow Y$), *conjunctions* ($X \wedge Y$) or *equivalences* ($X \Leftrightarrow Y$). In this study, Blanchard *et al.* outlined that depending on metric focus type, association rules are described as *quasi–implication*, *quasi–conjunction* and *quasi–equivalence.*

A *quasi-implication* is a rule, noted as $X \Rightarrow Y$, with the specificity that its examples are XY and $\bar{X}\bar{Y}$, and the counterexamples are $X\bar{Y}$. Thus, a quasi-implication is equivalent with its contrapositive $\bar{Y} \Rightarrow \bar{X}$. Based on this equivalence, we can define a measure of quasi-implication as follows.

Definition 4.3.5

A quasi-implication measure, denoted as m_{qi}, is an interestingness measure verifying the following condition: $m_{qi}(X \to Y) = m_{qi}(\bar{Y} \to \bar{X})$. ■

Starting from logical conjunctions, we can say that a quasi-conjunction (denoted as $X \wedge Y$) has the examples $X \cap Y$ and the counter-examples $X \cap \bar{Y}$ and $\bar{X} \cap Y$. Thus, the quasi-conjunction $X \wedge Y$ is equivalent to $Y \wedge X$.

Definition 4.3.6

A quasi-conjunction measure, denoted as m_{qc}, is an interestingness measure verifying the following condition: $m_{qc}(X \to Y) = m_{qi}(Y \to X)$. ■

A quasi-equivalence, denoted as $X \Leftrightarrow Y$, has as examples $X \cap Y$ and $\bar{X} \cap \bar{Y}$, and as counter-examples $X \cap \bar{Y}$ and $\bar{X} \cap Y$. Thus, the quasi-equivalent rule is equivalent to its contrapositive and the reciprocal. In this context, we can define a quasi-equivalence measure, as a measure m_{qe} verifying

$$m_{qe}(X \to Y) = m_{qe}(Y \to X) = m_{qe}(\bar{Y} \to \bar{X}) = m_{qe}(\bar{X} \to \bar{Y})$$

Nature-based classification. This last classification of measure criterion concerns their descriptive or statistic nature. Thus, an interestingness measure is descriptive if it does not vary with the cardinality variation. On the contrary, a measure is stated as statistical if it varies with the cardinality variation.

In the following we will make a detailed presentation of several important interestingness measures starting from the above classification. The measures chosen for the section below are the main measures used in the approach that we propose.

Table 4.2 contains a selection of metrics organized by means of the semantic classification. It is important to note that there are few implication, inclusion and statistical measures.

Table 4.2: The main interestingness measures organized by means of the semantic classification. The measures according to the nature are indicated by the font style: statistical measures are *italic* and the others are descriptive. Also, p-value test (probabilistic) based measures are suggested by the * symbol.

	Rule $X \to Y$ & $\neg Y \to X$	Quasi-Implication $X \to Y$ & $\neg Y \to \neg X$	Quasi-Conjunction $X \to Y$ & $Y \to X$	Quasi-Equivalence both
Equilibrium	Confidence, Precision [5] Sebag-Schoenauer [180] Example and Counterexample Rate [112] *Laplace Correction* [84]			
Independence	J-Measure	Loevinger [137] *Implication Index* [87] **Implication Intensity* [87]	Lift [36] **Likelihood linkage* [129]	Leverage [127] Odds Ratio [144] Yule's Q and Y[208] *Rule interest* [165]
Other			Jaccard [116] Dice [56] Item-relatedness [182]	Rogers-Tanimoto [176]

4.3.3 Lift (or Interest)

The lift measure was firstly defined by Brin *et al.* 1997 [36]. Regarding the previous classification, we can denote that lift is a deviation from independence measure and, also, a quasi-conjunction measure. Concerning the nature of the measure, it is a descriptive measure.

Definition 4.3.7
The lift measure is defined as

$$lift(X \to Y) = \frac{P(X,Y)}{P(X)P(Y)} = \frac{Confidence(X,Y)}{P(Y)}$$

pointing out the importance of the correlation between the two variables. ■

Let us consider the deviation from independence case. We outlined that a rule is in the case of independence if its variables X and Y are independent, thus, the probability of having in the same time X and Y is equal to $P(X \cap Y) = P(X)P(Y)$. Let us compute now the value of the lift measure in case of independence:

$$lift(X \to Y) = \frac{P(X,Y)}{P(X)P(Y)} = \frac{P(X)P(Y)//independence\ case}{P(X)P(Y)} = 1.$$

In consequence, as the lift takes a constant value on independence, we can say that the lift is a measure of deviation from independence.

Example 4.3.8 Let us consider a simple association rule $Pork \to Pear$ $[C = 83\%]$ – in 83% of cases, when we have $Pork$ in a supermarket basket, we have also $Pear$. The confidence metric evaluates the rule as being interesting. On the contrary, the lift value could prove the contrary; the result depends on the support of the $Pear$ item in the database. Two cases are possible:

- *supp*$(Pear) = 83\%$ – this is to say that, alone, $Pear$ item appears in 83% of baskets, thus, it is not surprising to have a confidence of 83%, and in reality, $Pork$ does not increase its chances to be in a supermarket basket. In consequence, computing the lift as $lif(R) = 1$, the rule is in the situation of independence;
- *supp*$(Pear)! = 83\%$ – the more the support of $Pear$ is different of 83%, the more the rule is interesting. ■

4.3.4 Implication Intensity

Implication intensity metric was proposed for the first time by Gras 1996 [86] and revised later by Gras and Kuntz 2008 [87]. It is based on a probabilistic model permitting to compare the number of counter-examples observed in data, $N_{X\bar{Y}}$,with their theoretic number. Thus, we randomly create two independent subsets A and B, under the hypothesis H_0; in this context, we note $N_{A\bar{B}} = |A \cap \bar{B}|$.

The rule $X \to Y$ is accepted with a threshold of $1 - \alpha$, if the probability that the number of counter examples in the observations is greater that the number of counter examples in the theoretic distribution is less than a tolerated error α; that is to say that $Pr(N_{X\cap\bar{Y}} \geq N_{A\cap\bar{B}}) \leq \alpha$.

Several distributions can be used as the hypergeometric, binomial or Poisson distributions, but in this study we will consider that the $N_{A\cap\bar{B}}$ variable is distributed following the Poisson distribution with the parameter $\lambda = NP(X)P(\bar{Y})$. When the approximation is justified ($\lambda \geq 4$), the variable $\tilde{N}_{A\cap\bar{B}} = \frac{N_{A\cap\bar{B}}-\lambda}{\sqrt{\lambda}}$ is following the central normal distribution. The observed value of $\tilde{N}_{A\cap\bar{B}}$ is $\tilde{n}_{X\cap\bar{Y}} = \frac{n_{X\cap\bar{Y}}-\lambda}{\sqrt{\lambda}}$.

Concerning the semantic classification, the implication intensity is a statistical and deviation from the independence metric.

4.3.5 Item-Relatedness Measure

Let us consider the supermarket taxonomy. It is obvious that the *pork* concept is semantically more close to the $beef$ concept than the *pear* concept. As already states, taxonomy-like structures propose the organization of concepts by means of the *is-a* relation in function of their shared characteristics. Thus, in our case, *pork* concept is more close to $beef$ because it shares more characteristics together than it shares with the other concept: for example, we can say that $beef$ and *pork* are meals that we eat as a main dish, while *pears* are eaten as a dessert.

In this context, an important number of semantic measures were proposed in the literature. The most basic ones are defined through subsumption relations and they are described by the shortest path or by the shared information. Interesting surveys are proposed by Blanchard *et al.* 2005 [29] and Gandon 2008 [76].

Let us consider that the following association rule is extracted by a traditional technique: *milk* $\rightarrow$ *butter* with a comfortable support and confidence so that the system could consider it interesting and could show it to the user. Nevertheless, is this rule really interesting? Although a great part of objective indicators could establish that the rule is interesting, it could fail when studying the relation between its items – the *milk* item is very close semantically to the *butter* item. Generally, rules composed by related items are trivial, thus, we can conclude that this rule is trivial.

A descriptive quasi-conjunction metric, *Item-Relatedness*, was developed by Shekar and Natarajan 2004 [182] in order fulfill this drawback. Its main purpose is to measure the relatedness between the items of already discovered association rules. For this purpose, the authors proposed to use a fuzzy taxonomy in order to describe the relations between rule items. The difference between simple taxonomies and fuzzy ones is that fuzzy taxonomies allow a node to have multiple parents and also they permit weighted *is-a* relations.

Definition 4.3.9

Let us consider two items X and Y and a fuzzy taxonomy organizing the items. The item-relatedness metric is defined according to all possible paths from X to Y in the taxonomy by means of three important measures. The first measure, HM (Highest-Level Node Membership), computes the implication of each item in their common parent node; the second one, HR (Highest-Level Relatedness), measures the *level difference between the parent of X and Y, and the taxonomy root. And, finally, the last measure, NSR (Node Separation Relatedness), is defined as the length of the path connecting the two items.*

As a consequence, the item-relatedness measure of X and Y can be defined formally as follows:

$$\mathrm{IR}(X,Y) = \Sigma_{path} \frac{(1 + HR_{X,Y}(path))HM_{X,Y}(path)}{NSR_{X,Y}(path)}$$

with the variable path being all the possible paths between X and Y. ■

To qualify the interestingness of an association rule, we can compute the item-relatedness metric between each pair of items. Generally, we can distinguished three types of similarity: between the antecedent items, between the consequent items or between the antecedent and the consequent items.

Example 4.3.10 In this example we will consider that only the NSR metric forms the item-relatedness measure of two items, and that the item pairs of a rule are formed between the antecedent and the consequent.

Considering the taxonomy in Figure 4.3 and the conditions imposed above, we will compute the item-relatedness of the following association rule:

$$grape,\ pear,\ butter \rightarrow milk$$

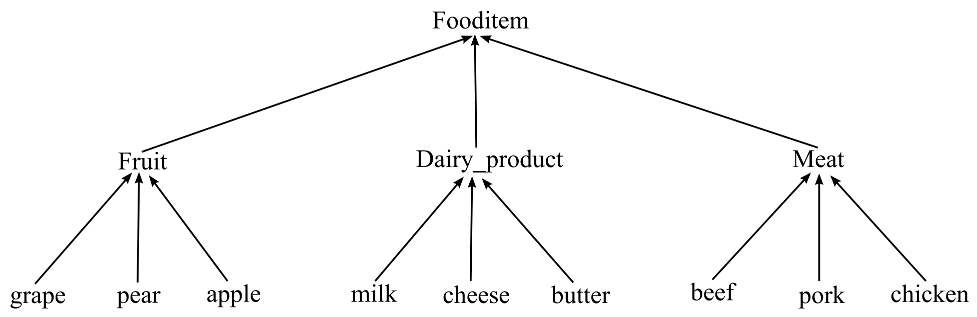


FIGURE 4.3: Supermarket taxonomy [136].

So, we will compute the metric for the following pairs: $(grape, milk)$, $(pear, milk)$ and $(butter, milk)$ and we can note that one single path is available for each pair:

$NSR_{grape,milk} = 4$
$NSR_{pear,milk} = 4$
$NSR_{butter,milk} = 2$

and the interestingness of the rule is computed as follows:

$$IR = min\{NSR_{grape,milk}, NSR_{pear,milk}, NSR_{butter,milk}\} = 2. \blacksquare$$

While Shekar and Natarajan measure the *item-relatedness* of an association rule, Garcia *et al.* developed in [79], and extended in [78], a novel technique called *Knowledge Cohesion (KC)*. The proposed metric is composed of two measures: *Semantic Distance (SD)* and *Relevance Assessment (RA)*. The *SD* one measures how close two items are semantically, using the ontology - each type of relation being weighted differently. The numerical value *RA* expresses the interest of the user in certain pairs of items in order to encourage the selection of rules containing those pairs. In this paper the ontology is used only for the *SD* computation and the authors propose a metric-based approach for itemset selection.

4.4 User-driven Interestingness Techniques

Previously, we showed the interest of using objective measures – the selected rules are interesting from statistical point of view. Nevertheless, are they interesting for the user? Generally, the rules selected by objective measures are far away from being interesting *also* from the user point of view. In this context, Carvalho *et al.* 2005 [50] tried to find a relation between the objective measures and the user interest. Thus, in the last decade, it was outlined a real need in integrating user knowledge (domain knowledge) in the discovering process of association rules [124]. Proposed as a solution for the selection of interesting rules, the user-based techniques are known as subjective interestingness measures.

The *ENIGME* system, developed by Ganascia *et al.* 1993 [74, 203], defends the idea that Machine Learning algorithms need to be connected to the Knowledge Acquisition environment. This research was proposed as a solution to the main problem of the *CHARADE* system – the huge amount of rules generated. In this context, *ENIGME* proposes to help the expert to acquire his knowledge which will guide the learning process.

ENIGME integrates a partial model of expertise proposed by the KADS methodology in order to express the expert knowledge. In this context, it uses three of the four layers of KADS: domain layer, inference layer and task layer. First, the *domain layer* covers domain knowledge such as basic facts, concepts and relations used within the domain. For example, we can have here the attributes and their possible values. Second, the *inference layer* describes abstract inferences, which can be possibly applied on domain layer knowledge, and the connections among them, forming an inference structure. The former is composed of inference steps consisting of knowledge sources describing the inference, and roles linked to domain concepts. For example, we can imagine an inference structure composed of different connections between the attributes or sets of attributes. Third, the *task layer* has the role to specify which inferences are applied for a given problem, and which is their application order. Last, semantic networks are proposed in order to describe associations between role concepts.

The system is able to learn only those relations following the inference steps given by the expert. In this context, the dimension of the search space is importantly reduced.

The problem of user integration in the KDD process was first suggested by Fayyad *et al.* 1996 [63] while presenting the importance of prior and domain knowledge in all KDD steps. In the same year, Silberschatz *et al.* 1996 [185] detailed the ways of user integration, coming up with three types of discovery process, each type describing differently the division of work between the search engines and user knowledge:

- *Automatic* – The first one is an automatic process where the search engine discovers the knowledge and no action from the user is proposed;
- *Semi-automatic* – The second one is a semi-automatic process combining user prior knowledge with the search engine power. The goal of this type of approach is to limit considerably the search space in order to generate a limited number of patterns;
- *Manual* – The last process type concerns the knowledge extraction done only by the user and by using queries.

The first and the last processes are two extreme approaches and, unfortunately, it was proved that on the one hand it is nearly impossible for a user to find alone, using queries, the information that interests her/him. On the other hand, the patterns discovered using only search engines are frequently delivered to the user in a huge volume, that makes impossible their interpretation. In consequence, the most interesting solution is the second proposition – to integrate prior and/or domain knowledge in the KDD process.

This section is dedicated to subjective measures and, in the following, we will make a briefly presentation of the important notions of unexpectedness, and actionability, we will continue by detailing the most important works in user-based association rule mining, we will present the first approaches using ontologies, and we will conclude with a comparison between post-processing and measure-based techniques.

4.4.1 Actionability vs. Unexpectedness

A first important classification of subjective measure was done by Silbershatz and Tuzilin in [183, 184]:

- *unexpectedness* – a pattern is interesting if it is surprising to the user;
- *actionability* – a pattern is interesting if it can help the user take some actions.

Comparing these measures, we can note that they are independent, and they characterize the discovered rules from two subjective different points of view [183].

First, let us consider the market basket application field; the main interest is to find important information in order to help the managers to improve sells. In consequence, the goal of this technique is to propose a set of new knowledge over the initial database that could be used in order to take useful actions. Starting from this idea, Silberschatz and Tuzhilin 1995 [183] introduced the notion of actionability as a subjective measure. In a first attempt, actionability was considered an abstract notion and defining it was not an easy task – Silbershatz and Tuzilin continued their presentation based only on unexpectedness. On the contrary, nowadays, actionability interests more and more researchers because of the increase of association rule mining applications in commercial and benefits-based databases; thus, several surveys discuss the actionability problem [101].

A first attempt in defining a process to extract actionable rules was done by Adomavicius and Tuzhilin 1997 [1] with the proposition to use a hierarchy of actions. These actions should be taken when some trigger rules are discovered. The authors proposed to assign to each node-action actionable rules/patterns by data mining queries. Later, Gamberger and Lavrac [72] suggested an expert-guided approach for knowledge discovery based on the idea of finding subgroups of rules which can be valuable information for the user.

Introduced by Ras *et al.* 2008 [173] and further investigated in [174, 207], the *action rule* notion is defined as a rule extracted from a decision system that describes a possible transition of objects (here, denoted as transactions in database) from one state to another with respect to a given attribute, called *decision attribute.* The attributes used in this type of decision systems are organized in *stable* and *flexible* attributes. *Date of Birth* or *Family Name* are examples of *stable attributes* – they do not change their values. On the contrary, *flexible attributes* can change their value; for example, in banking case, the bank can change the interest rate on an account. In a first step, the proposed algorithm generates the set of transactions corresponding to each combination of attributes. Next, classification rules are easily generated. Action rules are constructed from pairs of classification rules defined with respect to the decision attribute, as presented in the following example.

Example 4.4.1 Let us consider the following sampling database from a basket market database (Table 4.3) where *Pear* is the decision attribute, *Date of Birth* and *Family Name* are stable attributes, and *Pork* is a flexible attribute.

Table 4.3: Basket market sampling database

TID	Date of Birth	Family Name	Pork	Pear
1	26/08/1977	Martin	0	0
2	2/03/1985	David	1	0
3	26/08/1977	Martin	1	1
4	2/03/1985	Martin	0	0

The first step defines the set of transactions for each attribute combination. For example, for *Pork* attribute we have the following set of transactions: $\{t_1, t_4\}$ for $Pork = 0$ and $\{t_2, t_3\}$ for $Pork = 1$.

Next, after the first step, the following classification rules are generated:

$R_1 : FamilyName = Martin \wedge Pork = 0 \rightarrow Pear = 0$

$R_2 : FamilyName = Martin \wedge Pork = 1 \rightarrow Pear = 1$

$R_3 : FamilyName = David \wedge Pork = 1 \rightarrow Pear = 0$

...

It is important to note that the $FamilyName$ attribute is stable, so it cannot be modified. Thus, we can generate action rules using only the $Pork$ attribute. Thus, starting from rules R_1 and R_2 we generate the following action rule:
(R_1, R_2)-action rule $[(Pork, 0 \rightarrow 1)](x) \Rightarrow [(Pear, 0 \rightarrow 1)]$. ■

Later, in [172], Ras *et al.* defined *association action rules* as action-based association rules and they proposed an *Apriori*-like algorithm based on a support-like constraint. For this purpose, *atomic action sets* and *action sets* are defined, which are comparable with *frequent items* and, respectively, *frequent itemsets*, but action-oriented. This method is very interesting for business companies in order to improve their profit. Nevertheless, the use of action rules does not ensure us that the number of extracted values is more reduced compared to the other techniques.

In the same direction, Cao *et al.* [46, 49] proposed the notion of *domain-driven data mining.* The Actionable Knowledge Discovery (AKD) problem aims to discover deliverable knowledge in the form of business-friendly and decision-making actions, and can be taken over by business people seamlessly. To this end, domain driven data mining, known as D^3M, aims at an effective involvement of several intelligences: Data Intelligence, Domain Intelligence, Network Intelligence, Human Intelligence, Social Intelligence, Intelligence Metasynthesis.

Continuing this study, Cao *et al.* were interested in discovering interesting knowledge for constrained business. Thus, they proposed a new methodology, called *domain-driven in-depth pattern discovery (DDID-PD)* [47, 48] expressing a domain driven view of discovering knowledge satisfying real business needs.

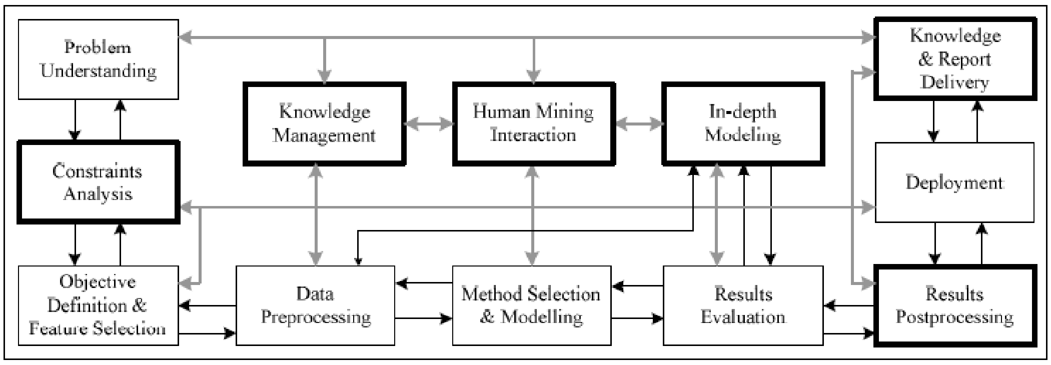


FIGURE 4.4: DDID-PD Methodology [48]

The Figure 4.4 represents the actionable knowledge discovery process steps and highlights the specific steps of the DDID-PD methodology. Every step of DDID-PD methodology could involve user/domain knowledge and expert evaluation.

The authors also outline that four user-driven procedures guide a real world data mining process and that they are necessary in order to successfully discover interesting knowledge: *constraint mining*, *in-depth mining*, *human-cooperated mining* and *loop-closed mining.*

The second type of measures – unexpectedness (or *novelty* [16] by certain researchers) – was proposed in order to solve the pattern triviality problem. Thus, unexpectedness metrics measure the surprise level of the discovered patterns regarding different notions that vary in function of the proposed approach. For example, Dong and Li [59] view the unexpectedness notion in terms of neighbourhood parameters. The authors proposed a neighbourhood based on a item difference distance, and they researched the rules the most unexpectedness regarding its neighbourhood. Another unexpectedness approach is to compare the extracted patterns (rules) with rule-based user knowledge using syntactical [132, 133, 135, 136] or logical [154, 155] techniques. The rest of this section is dedicated to user-driven techniques based on unexpectedness.

4.4.2 Templates

A first approach of template-guiding association rule mining by syntactic constraints was proposed at the same time as the association rule technique by Agrawal *et al.* [5]. Later, in 1994, extending this notion of syntactic constraints and inspired from the work in [109], Klemettinen *et al.* [122] proposed *templates* in order to allow the user to define family of rules that interests or not her/him. First, the user has to structure database attributes in a hierarchy which classifies them by means of an *is-a* relation.

Definition 4.4.2
A template is defined as follows:

$$A_1 \dots A_n \to A_{n+1}$$

where an element A_i is a class name from the hierarchy or an expression $C+$ or $C*$, where C is a class name. ■

The difference between this proposition and Agrawal's one comes from the fact that all elements in syntactic constraints are attributes of the database, while in this proposition, they are concepts of an attributes hierarchy. Moreover, two types of templates are introduced: *inclusive* and *restrictive* templates. For a rule to be considered interesting, it has to match an inclusive template. However, if it matches one of the restrictive templates it is considered uninteresting, and it will not be proposed to the user. An association rule matches a template if it is an instance of the template.

Example 4.4.3 We will exemplify this approach over the supermarket database. Suggested partially in Figure 4.5, an hierarchy is proposed by the user in order to organize the food items in the supermarket.

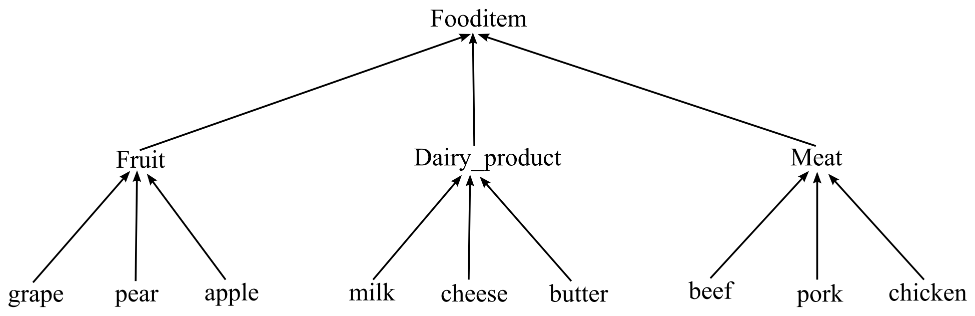


FIGURE 4.5: Supermarket hierarchy sample.

Proposing the following inclusive template:

$$(Inclusive\, Template) : \ Fruits\ , DailyProducts\ \rightarrow\ Meal$$

the user selects a set of interesting rules and two of these are presented below:

$R_1 : Pear,\ Milk\ \rightarrow\ Pork$
$R_2 : Apple,\ Milk\ \rightarrow\ Chicken$

Then, the user decides that *Pear* elements should not be included in rules, thus he/she proposes the following restrictive template:

$$(Restrictive\, Template) : \ Pear,\ DailyProducts\ \rightarrow\ Meal.$$

Thus, the system will declare the first rule uninteresting, and it will propose only the rule R_2 as being interesting. ■

The main problem of this approach resides on the method of matching rules to templates. For a rule to match a template, all its elements should be instances of elements in templates, and all template elements should have at least one instance in the rule. That is why, we consider that templates is a quite restrictive solution. Moreover, due to matching definition, restrictive templates are not so powerful that we thought. In fact, a restrictive template in order to declare a rule uninteresting, it should be composed of elements subsuming all the attributes of the rule, being in a subsuming relation with the inclusive template elements. We think that it is more interesting to describe differently the matching technique in order to use all the power of selecting/filtering templates.

Another way of using user templates is proposed by Silberschatz and Tuzhilin 1996 [185] in the *Data Monitoring and Discovery Triggering (DMDT)* system. The system limits the user feedback by guiding the system while searching for new knowledge. The main idea is that first a set of triggers and a set of templates are declared. A trigger has an IF-THEN form and it is composed of classical trigger conditions over the database in the IF clause, and a set of templates to be activated in the THEN clause. Then, considering a database D, when a trigger is fired because several significant changes suggested in the IF-clause of the trigger are detected in data, templates from

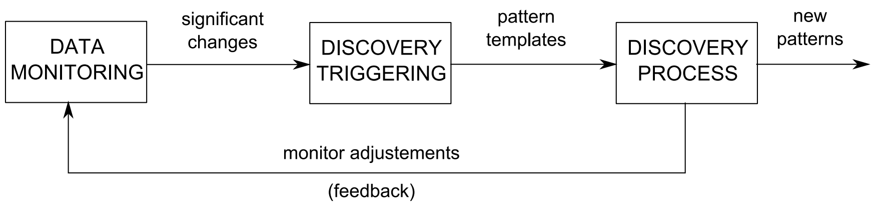


FIGURE 4.6: The DMDT System - User Feedback based on Fired Pattern Templates [143, 185].

THEN-clause are activated. Thus, the set of all templates from all the triggers fired is given in the input of the discovery process module as shown in Figure 4.6.

The set of activated pattern templates extract several patterns according to the user templates. Moreover, the discovered patterns could be used in order to modify the IF-THEN triggers or the templates, so that patterns closer to user interest could be extracted. New triggers and templates are injected and the whole process is restarted.

4.4.3 Beliefs

In 1995, with the classification of subjective measures, Silbershatz and Tuzilin [183, 184] defined user knowledge as a set of convictions – called *beliefs* – used in order to evaluate the unexpectedness of extracted patterns. Comparable with the *templates* proposition, this approach brings new interesting ideas. First, each belief is defined as arbitrary predicate formulae expressed in first–order logic, and second, a *confidence degree* is attached to each belief measuring how much the user trusts the belief.

This approach integrates two types of beliefs, in function of the character of each one:

- *Soft beliefs* are those knowledge that the user accepts to modify if the discovered patterns contradict them. In other words, the user wants to confirm and, later, to develop, this type of knowledge. In order to define the confidence degree with the extraction of a new pattern, several methods were proposed as the *bayesian*, the *Dempster–Shafer* and the *frequency approach*. The interestingness of the new extracted pattern is computed by how the new pattern changes the degrees of beliefs.

- *Hard beliefs* are those knowledge that the user will not change whatever new patterns are extracted. Thus, the confidence degree is not defined to this type of beliefs. In conclusion, patterns which contradict a hard belief are always interesting for the user.

Example 4.4.4 In order to exemplify this approach, we consider the supermarket case, and we define the following belief:

α = *for a complete day, the supermarket propose 500kg of grapes to their customers.*

First, we take the case when the previous belief is a hard belief. Assume that the following pattern is discovered by a mining technique:

$p =$ *527kg of grapes are sold each day.*

It is obvious that the discovered pattern contradicts the belief because it is not possible that 527kg of grapes are sold in a day, knowing that the supermarket offers only 500kg. This contradiction with the hard belief outlines possible errors in data and the pattern worths the attention of the users.

Second, we take the case when the proposed beliefs is a soft one. In this context, we consider the Bayesian approach proposed by the authors to compute the degree of the belief α, and we assume that the conditional probability of the belief on the initial evidence E is $P(\alpha|E) = 0.85$. If we consider a new pattern, $E_0 =$ *527kg of grapes are sold each day*, we can assume in this context that the conditional probability is $P(\alpha|E_0, E) = 0.89$. The interest of a pattern is defined by the difference between the conditional probability of the belief λ with and without the pattern. In our case, the interest is $I(p, \lambda, E) = 0.04$ and the pattern is not considered very interesting. ■

Unfortunately, this work is still in a development state, and no further advancements were done. As a result, this approach is not functional.

4.4.4 Fuzzy Rules

In [131] and later in [136], Liu and Hsu suggested to integrate user knowledge in the post-analysis of classification rules. The user knowledge are defined in a fuzzy way using the same syntax as the classification rules, as follows:

$$\text{IF} P_1,\ P_2,\ P_3,\ \ldots P_n \text{ then } C$$

where P_i and C mean *attribute OP value*, with $OP \in \{=, \neq, <, >, \leq, \geq\}$.

Liu and Hsu introduced for the first time several ideas related to the similarity between the user knowledge and the extracted rules, ideas that will be detailed in their future works. Thus, the notion of similarity and difference between the user knowledge and the discovered rules are defined as it follows:

- *Rule Similarity* – two rules are similar if their antecedents and consequents are similar;

- *Rule Difference* – two rules are different if they are dissimilar; in consequence, the discovered rule is *unexpected* regarding the fuzzy rule provided by the user. Several types of dissimilarity are proposed according the differences of antecedents and/or of consequents:

  - *Unexpected consequent* – the conditions are similar, but the consequents of the two rules are dissimilar;
  - *Unexpected antecedent* – the consequents are similar, but the antecedents of the two rules are dissimilar. Two types of unexpectedness regarding the antecedent are proposed:

  * *Contradictory antecedent* – the rule antecedent attributes are the same, but they have different values;
  * *Unanticipated condition* – the rule antecedent attributes are different.

Example 4.4.5 Let us consider the following set of classification rules:

R_1 : IF $Age > 60,\ Pear >= 3$ then $Class = Vegetarian$
R_2 : IF $Age < 50,\ Pear = 2$ then $Class = Vegetarian$
R_3 : IF $Age < 30,\ Pear = 0.5$ then $Class = NotVegetarian$
R_4 : IF $Apple = 3$ then $Class = Vegetarian$

and the following user knowledge expressed as a fuzzy rule:

IF $Age = OLD,\ Pear = HIGHT$ then $Class = GOOD$

where OLD is considered to be someone aged of more that 50 years, products are denoted with $HIGHT$ if someone bought more than 2 kilos of this product, and, finally, the $Class$ is considered to be $GOOD$ if the person is classified to be $Vegetarian$.

Thus, we can note that the rule R_4 has a unanticipated antecedent regarding the user knowledge, and that the rule R_3 has a contradictory antecedent. Also, the rule R_1 is similar to the given rule. ■

4.4.5 User Expectations – General Impressions

In the previous approach, Liu *et al.* [131] proposed fuzzy rules for user to give his/her knowledge, but they noted that it is quite difficult for a user to express exactly what he/she knows. Thus, Liu *et al.* 1997 [132] made an important classification of user knowledge:

- *General Impressions (GIs)* – user vague feelings about the domain. For example, in a supermarket, the user may know that customers may buy both pork and pears, but he/she does not know if one product sale implies the sale of the other one;

- *Reasonably Precise Knowledge (PRK)* – the user has precise ideas of concerning discovered rules. For example, if a customer buys pork, with a certain probability, he/she will also buy grapes.

In a first attempt, the authors concentrated their work on general impressions and they developed the following two syntaxes for the general impressions:

TYPE$_1$: $i_1ID_1, \dots i_wID_w \to C_j$
TYPE$_2$: $i_1ID_1, \dots i_kID_k$ AND $i_{k+1}ID_{k+1}, \dots i_wID_w \to C_j$

where i_j is an attribute, $ID \in \{<, >, \ll, |, [aset]\}$ and C_j is a class.

The formalism $i_1ID_1 \to C_j$ can be explained by: the more the attribute i_1 has a variation defined by the operator ID_1, the more there are chances to lead to class

C_j. For the second type, the second part in the antecedent of the rule is optional and helps the user to define more flexible general impressions.

For matching and ranking discovered rules, Liu *et al.* use the same method employed for fuzzy rules, composed by two important steps:

- in the first step, the user defines his/her expectations using general impressions;
- in a second step, the discovered rules are analyzed and compared to general impressions, and those interesting rules are selected.

In 1999, Liu *et al.* [136] proposed to adequate their General Impressions formalism that they developed later in [133]. Thus, a specification language was designed in order to allow the user to express his/her expectations and goals. Based on the general impression definition introduced in [132], the main idea of this specification language is to represent user feelings concerning item relations and implications in the database. Thus, three levels of specification are developed: *General Impressions* and *Reasonably Precise Concept* (already defined in [132]), and a new one – *Precise Knowledge.* The third level, Precise Knowledge, was added in the idea to allow the user to express exact prior knowledge with the vague feelings represented by the first two specifications.

Inspired from the *generalized association rules* [189] (for more details go to the Section 4.5), Liu *et al.* integrated concepts of item taxonomies as elements in their specification languages in order to generalize the rule selection. Thus, let us consider the taxonomy of items presented in Figure 4.7. In this taxonomy, the leaves such as *grape*, *pear*, ... are items from the database, and the other concepts such as *Fooditem*, *Fruit*, ... are classes of items. Thus, we can say that $\{grape, pear, apple\} \subset Fruit \subset Fooditem$.

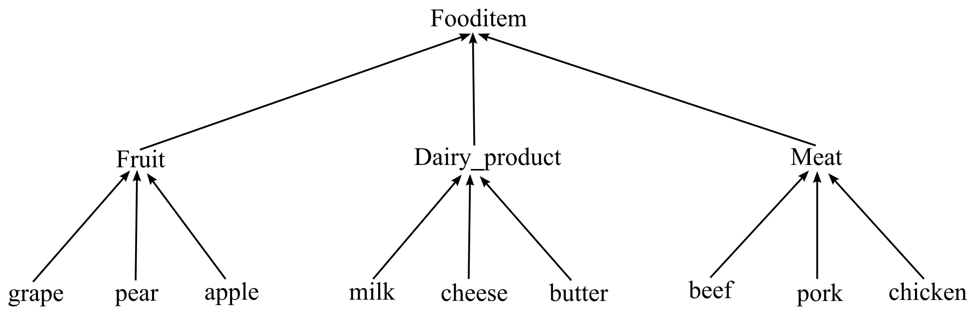


FIGURE 4.7: Supermarket taxonomy [136].

Definition 4.4.6

*The new formalism of **General Impressions (GIs)** is described as follows:*

$$gi(< S_1, \ldots, S_m >)[support,\ confidence]$$

where each S_i can be an element of the taxonomy and $*$, $+$, $\{\}$ operators can be applied over. S_i+ represents one or more occurrences of the element S_i, S_i* represents zero or more occurrences of the element S_i, $\{S_{i_1}, S_{i_2}\}$ expresses S_{i_1} OR S_{i_2}. ■

Even if the authors reworked their formalism already proposed in [132], the objectives of the General Impressions remain the same – to represent user vague feelings concerning some associations in the database.

The matching process between general impressions and association rules consists in a syntactic comparison between the antecedent/consequent elements. Thus, each element in the general impression should find a correspondent in the association rule. Moreover, there should not be any item in the rule that is not the correspondent of an element from the GI.

Example 4.4.7 Let us consider the taxonomy in Figure 4.7 and let us also consider that the user might believe that there is an association between *cheese* or *milk*, *Meat* items (zero ore more) and *pear*. In consequence, we can express these expectations using the following General Impression:

$$\text{GI}: \ gi(<\{cheese,\ milk\},\ Meat*,\ pear>)$$

Let us consider that the following association rules were discovered using classical techniques:

$R_1: \ cheese \rightarrow pear$

$R_2: \ pork \rightarrow pear,\ apple$

$R_3: \ milk,\ pear \rightarrow pork$

We can note that the rules R_1 and R_3 match the general impression, and that the rule R_2 does not match the *GI* because no element from $\{cheese,\ milk\}$ exists in this rules, and furthermore, the item *apple* is not the correspondent for any element in the *GI*. ■

Definition 4.4.8
Reasonably Precise Concept *represents the vague feelings of the user concerning the existence of some implicative associations between items, and the proposed syntax is described as follows:*

$$rpc(< S_1, \ldots, S_j \rightarrow S_{j+1}, \ldots, S_m >)[support,\ confidence]$$

with the same notations as for GIs. ■

The main difference between GIs and RPCs is that when the user expresses his/her expectations using GIs, he/she is not sure on which elements he/she should put in the antecedent, and which elements in the consequent. On the contrary, RPCs express implicative associations.

Example 4.4.9 For this example we will use the same set of rules as for the General Impression:

$R_1: \ cheese \rightarrow pear$

$R_2: \ pork \rightarrow pear,\ apple$

$R_3: \ milk,\ pear \rightarrow pork.$

In this case, the user believes that the customers buying *cheese* or *milk*, and maybe buying *Meat* products, could also buy *pears*. This can be expressed using the RPC formalism as follows:

$$\text{RPC}: \ rpc(< \{cheese, \ milk\}, \ Meat* \rightarrow pear >).$$

In this context, the rule R_1 matches the RPC, and the rules R_2 and R_3 do not match the RPC: R_2 does not contain *cheese* or *milk* products in the antecedent, and R_3 does not contain *pear* products in the consequent. ■

Precise Knowledge expresses that the user believes in a specific implicative association between items with exact thresholds of support and confidence. The main difference between PKs and RPCs is that the specification of support and confidence threshold in the PKs case is obligatory, and that in the case of GIs and RPCs it is optional.

4.4.6 Logical Contradiction and Exception Rules

A new technique to match user knowledge with discovered association rules was proposed by Padmanabhan and Tuzilin in [154], and revised and developed later in [155, 156]. The logical contradiction view of unexpectedness, slightly comparable with the exceptions idea of Suzuki [197], consists in extracting only those rules which logically contradict the consequent of the corresponding belief.

Definition 4.4.10

Let us consider an association rule $X \rightarrow Y$ and a user belief $A \rightarrow B$. The association rule is unexpected with respect to the user belief if:

- *$Y \wedge B \models FALSE$ – B and Y are in logical contradiction;*
- *$X \wedge B$ has an important support in the database – this condition eliminates those rules which could be considered unexpected, but which do not concern the same transaction in the database;*
- *$A, \ X \rightarrow B$ holds.* ■

Example 4.4.11 In order to exemplify this approach we consider the supermarket database in Table 4.4.

Further, assume that we have a belief outlining that customers who buy *pork* tend to buy also *milk*:

$$\text{Belief}: pork \ \rightarrow \ milk,$$

and we propose minimal thresholds of 30% for the support and 60% for the confidence.

In order to extract rules that contradict the previous belief, the generation starts with the itemset $pork, \neg milk$ with the aim to extract itemsets of the form $pork, \neg milk, X$,

TABLE 4.4: Supermarket database for Logical Contradiction.

Pork	Milk	Apple	Beef
1	1	1	1
1	1	1	0
1	1	1	0
1	1	0	1
1	0	0	0
1	0	0	0
1	0	0	0
0	1	0	0
1	1	1	1

where X is an itemset. Using the other two items (*apple* and *beef*), the following itemsets and rules are generated:

$pork, \neg milk, apple\ [S = 0\% - pruned]$
$pork, \neg milk, \neg apple\ [S = 30\%]$
$\qquad pork, apple \rightarrow \neg milk\ [C = 75\%]$
$pork, \neg milk, beef\ [S = 0\% - pruned]$
$pork, \neg milk, \neg beef\ [S = 30\%]$
$\qquad pork, beef \rightarrow \neg milk\ [C = 60\%]$
$pork, \neg milk, \neg apple, \neg beef [S = 30\%]$
$\qquad pork, \neg apple, \neg beef \rightarrow \neg milk\ [C = 100\%]$

■

Exception rules were introduced by Suzuki [196–200] and an extended survey was done by Duval *et al.* [60]. The search of exception rules aims at discovering a pair of rules composed of:

- a *common sense rule*, $CSR : X \rightarrow y$, where X is an itemset and y an item,
- and an *exception rule*, $ER : X, Z \rightarrow y'$, where Z is an itemset and y' is the negation of the y item.

Further, the unexpectedness of an exception rule is defined by an additional constraint – the *reference rule* $Z \rightarrow y'$ should have a low confidence. The reason is that if the reference rule has a high confidence we can consider that the itemset Z implies the y' item and not the X one, so, in this context, the exception rule is not connected to the common sense one.

The difference between this approach and the logical contradiction one mainly comes from the fact that beliefs in logical contradiction, which correspond to common sense rules, are provided by the user, while the common sense rules are discovered with the exception rules. In this context, the search algorithms are quite different: Padmanabhan and Tuzilin proposed *Apriori*-like trees in order to generate unexpected rules starting from user beliefs, while Suzuki developed a new algorithm based on a tree-like structure having as nodes the rule-pair common sense-exception.

4.4.7 Preference Model

Jiang 2006 [119, 213] introduced an approach comparable with the Logical Contradiction, but different by its diverse views over the unexpectedness and by the use of a preference model. The proposed method is designed for the extraction of classification rules.

In this context, Wang *et al.* studied what the user attempts to do with its knowledge and came with an original idea – to separate the user knowledge for each transaction in database. Thus, the novelty of this approach is the *preference model* which is a specific type of user knowledge representing how the basic knowledge of the user, called *knowledge rules ($\mathcal{K}$)*, will be applied over a given scenario or tuples of the database. Thus, the user proposes a *covering knowledge ($\mathcal{C}_t$)* for each tuple (t) – a subset of the knowledge rule set $\mathcal{K}$ that the user prefers to apply to the tuple t. At the end, the approach validates the transactions which satisfy the extracted rule, and the latter is interesting if the cover violates the transactions.

More formally, given a database D, a discovered rule r and a set of tuples S satisfying r, the following metric defines the unexpectedness (support/confidence):

$$U_{sup}(r) = \frac{\sum\{agg(\{v(t,R)|R \in \mathcal{C}_t\})|t \in S\}}{|D|}.$$

The metric is computed using a violation measure between the user knowledge $\mathcal{K}$ and each data tuple in S, and defined using agg – a well-behaved aggregate function (i.e. $min(V) \leq agg(V) \leq max(V)$ and for $V \leq V'$, $agg(V) \leq agg(V')$, where V and V' are vectors).

Wang *et al.* further detailed the *unexpectedness of a rule r* as:

$$U_{nexp}(r) = \frac{U_{sup}(r)}{supp(r)}.$$

Example 4.4.12 In order to exemplify this approach, let us consider a supermarket example. In Table 4.5 we present a supermarket database where the first column represents the transaction identifier, the following 4 columns the value of each attribute, and the last column the target-attribute – *Grape*? with possibilities 1 or 0.

Let us consider that the following two rules form the user knowledge $\mathcal{K}$:

$R_1 : Pork = Not\_1 \rightarrow Grape? = 1$

$R_2 : Milk = Not\_3 \rightarrow Grape? = 1.$

TABLE 4.5: Supermarket database for Preference Model approach.

TID	Pork	Milk	Apple	Beef	Grape?
$t_1(R_1)$	3	3	2	0	0
$t_2(R_1)$	3	3	2	1	0
$t_3(R_1)$	2	3	2	0	0
$t_4(R_2)$	1	2	2	0	1
$t_5(R_2)$	1	1	1	0	1
$t_6(R_2)$	1	1	1	1	0
$t_7(R_1)$	2	1	1	1	1
$t_8(R_1)$	3	2	2	0	0
$t_9(R_1)$	3	1	1	0	1
$t_{10}(R_2)$	3	2	1	0	1
$t_{11}(R_1)$	3	2	1	1	1
$t_{12}(R_1)$	2	2	2	1	1
$t_{13}(R_1)$	2	3	1	0	1
$t_{14}(R_2)$	1	2	2	1	0

The fuzzy values match the nearest values in the database. In this example we will consider that the depth of the covering – the number of knowledge rules – for each tuple is 1, and, in function of the matching/violating result between the user knowledge base $\mathcal{K}$ and the tuples of the database, to each tuple a knowledge rule is attached (column 1 in the Table 4.5).

Applying a minimum U_{sup} of 20%, the following unexpected rules are generated:

$r_1 : Beef = 1 \rightarrow Grape? = 0$

$r_2 : Apple = 2 \rightarrow Grape? = 0$

$r_3 : Pork = 3 \rightarrow Grape? = 0.$

Let us consider the first rule r_1. The transactions satisfying this rule are $\{t_2, t_6, t_{14}\}$, knowing that in these 3 transactions: $\{t_2\}$ violates the knowledge rule R_1 and $\{t_6, t_{14}\}$ violate R_2, each transaction satisfying r_1 violates its covering, in consequence r_1 is interesting. Formally, we can compute the three metrics of the rule: $U_{sup} = 3/14 = 21.43\%$, $U_{conf} = 3/6 = 50\%$, $U_{nexp} = 3/3 = 100\%$. ■

4.4.8 Comparative Study

In the following, we will discuss the comparative study that we made in the field of user-based interestingness measures. We studied 25 techniques for association rules or classification rules, and we could note different formalisms used in order to represent the knowledge in the mining process: templates, beliefs, meta-rules, query language, taxonomies, ontologies.

Next, we organized the 25 techniques in 3 classes:

- the techniques using implications/templates/query language;
- the techniques using both templates and taxonomies;
- the techniques using taxonomies or ontologies.

For the first class, we observe three main limits. First, the representation language is limited and static; it does not propose possibilities to evolve. Second, in the most part of cases, the user cannot choose the action to apply over the templates (pruning, selection, exception, . . .). Last, in several cases, the formalisms are difficult to understand.

For the second class, we have the limits of the first class to which we add the limits related only to taxonomies. Thus, taxonomies have a weak expressiveness; they do not allow users to improve the knowledge base with new informations.

The third class offers interesting solutions, but, unfortunately, these techniques deal with only one problem at a time (a quite marginal one) which does not permit to definitively validate the quality of discovered rules. Moreover, the techniques using ontologies do not use the power of ontologies – the reasoners.

The complete study is presented in Tables 4.6 and 4.7.

Table 4.6: Comparative study over the subjective measures. (1)

	Interestingness Measure	Year	Application	Foundation	Subjective Aspects	User Knowledge Representation
1	Shapiro and Matheus Projected Savings [141, 166]	1994	summaries		unexpectedness	Pattern deviation
2	Klementinen *et al.* Templates[122]	1994	association rules	syntactic	unexpectedness	Templates
3	Silberchatz and Tuzilin Beliefs[183]	1995		probabilistic	unexpectedness	Beliefs
4	Anand *et al.* EDM Framework [8]	1995	classification rules	syntactic	unexpectedness	Hierarchical Gen. Trees Attribute Relationship Rules Environment Constraints
5	Liu *et al.* Fuzzy Matching [131, 132]	1996	classification rules	syntactic	unexpectedness	Fuzzy rules
6	Imielinski *et al.* M-SQL [113, 115]	1996	association rules	queries		M-SQL query language
7	Kamber *et al.* Metarules [120]	1997	multi-dimensional association rules	syntactic	unexpectedness	Metarules
8	Liu *et al.* General Impressions [133, 136]	1997	association rules	syntactic	actionability unexpectedness	General Impressions Reasonably Precise Concepts Precise Knowledge
9	Baralis and Psaila Scenario Templates [17]	1997	association rules	syntactic		Scenario Templates Query Languages
10	Ng *et al.* Constrained Queries [150]	1998	association rules	syntactic	syntactic	Constrained Association Queries
11	Adomavicius and Tuzhilin Web Profiling [2, 3]	1999	profile rules	rule grouping syntactic	novelty actionability	Taxonomies Templates
12	Padmanabhan and Tuzhilin Logical Contradiction [154–156]	1999	association rules	logical statistic	unexpectedness	Beliefs
13	Wang *et al.* Preference Model [213]	2003	classification rules	violation measure	unexpectedness	Knowledge Rules Covering Knowledge

Table 4.7: Comparative study over the subjective measures. (2)

	Interestingness Measure	Year	Application	Foundation	Subjective Aspects	User Knowledge Representation
14	Chen *et al.* Raising [53, 225]	2003	association rules	generalization with higher support	actionability	Ontologies
15	An *et al.* Semantic Groups [7]	2003	association rules groups	semantic distance	unexpectedness	Taxonomy Semantic Networks
16	Jaroszewicz and Simovici Using Bayesian Networks [117, 118]	2004	patterns	support comparaison	unexpectedness	Bayesian Networks
17	Shekar and Natarajan Item-relatedness [182]	2004	association rules	similarity distance	unexpectedness	Taxonomy
18	Nazeri and Bloedorn Facts, Beliefs [149]	2004	association rules	syntactic	unexpectedness	Facts Beliefs/Preferences
19	Domingues and Rezende Taxonomy-based Generalization [58]	2005	association rules	generalization	actionability	Taxonomies
20	Faure *et al.* Using Bayesian Networks [62]	2006	association rules	dependencies comparison	unexpectedness	Bayesian Networks
21	Xin *et al.* User Interactive Feedback [217]	2006	patterns	progressive shrinking clustering	novelty	Log-linear model Biased belief model
22	Kotsifakos *et al.* [125]	2007	association rules	class membership	actionability	Ontologies
23	Antunes [11, 12]	2007	association rules	semantical distance based constraints	actionability	Ontologies Constraints
24	Bellandi *et al.* [21, 22]	2007	association rules	syntactic based constraints	unexpectedness	Ontologies Pruning Constraints Abstraction Constraints
25	Garcia *et al.* [78, 79]	2008	association rules	semantic distance relevance assessment	actionability	Ontologies Item Weight

4.5 Toward Ontologies in Association Rule Mining

In the above section we made a detailed survey on subjective measures integrating user knowledge in order to assess the quality of selected rules. In this context, the formalism for user knowledge representation has a real influence over the selection capability of the resulting interestingness measures. Rule-like formalisms allow users to express their expectations according to discovered rules by means of an intuitive formalisms.

Proposed in the philosophy branch by Aristotle, ontologies have interested researchers in Knowledge Engineering and Semantic Web fields. Ontologies have evolved over the years, from controlled vocabularies, to thesauri (glossaries), and later to taxonomies [210]. These representation structures were, step by step, proposed as user domain knowledge representations for association rule selection. Later, ontologies were considered as one of the most complex formalisms for knowledge representation, as a consequence the interest of researchers of association rule mining field toward ontologies was quite natural.

As stated by Nigro *et al.* 2007 [151], ontologies can be used in several ways in data mining as follows:

- *Ontologies for Data Mining Process*;
- *Metadata Ontologies*;
- *Domain and Background Knowledge Ontologies.*

A global vision over data mining with ontologies is presented in Figure 4.8.

Ontologies for Data Mining Process aim to codify the mining process description and to choose the most appropriate task according to the given problem. Thus, this type of ontology should contain important informations and characteristics of various techniques of data mining. As showed in Figure 4.8, these ontologies could be employed in several steps during the KDD process as it follows: the pre-processing data step in order to propose the most appropriate techniques of pre-processing, in the data mining step by choosing the right algorithms, and in the pattern generation by choosing the most interesting model for the discovered knowledge (for different interesting propositions see [26, 45]).

Metadata Ontologies aim to describe the construction process and the characteristics of items.

Domain and Background Knowledge Ontologies organize domain knowledge and/or user domain knowledge, or they could bring a basis for discovered knowledge representation. As a result, this type of ontology plays important roles at several levels of the knowledge discovery process: in pre-processing, but more important is the validation of knowledge. Further, we will detail approaches integrating two of the three types of ontologies: *Metadata Ontologies* and *Domain and Background Knowledge Ontologies*; generally, these two types of ontologies are used together because with the representation of background domain knowledge according to the database items, the construction of the database items is also described.

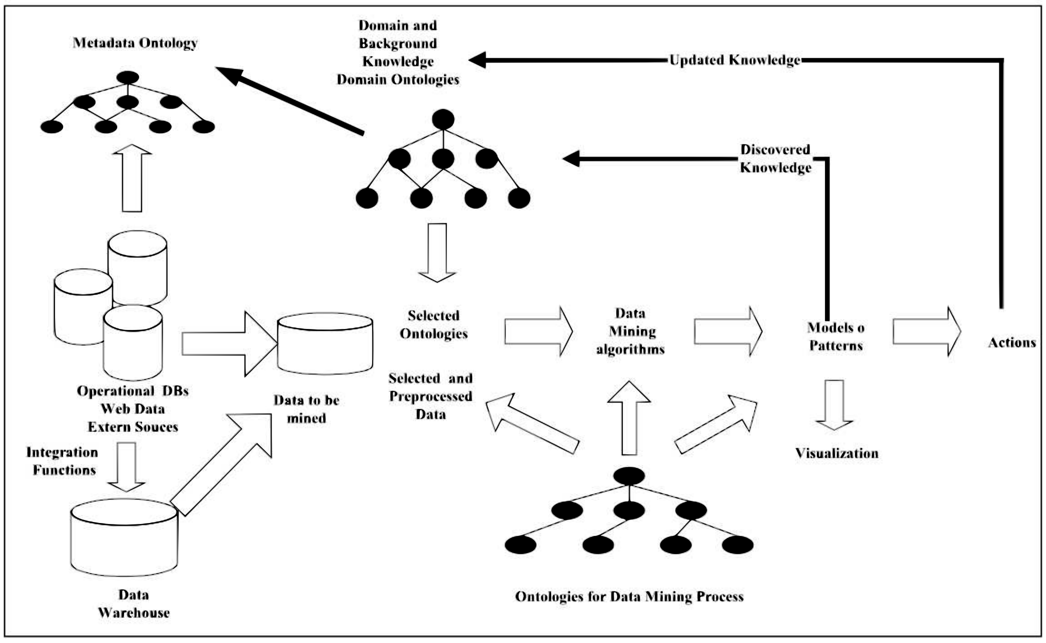


FIGURE 4.8: Data Mining with Ontologies [151].

The use of ontologies in association rule mining technique was anticipated with the use of a particular case of ontologies – taxonomies. The first ideas were introduced by Srikant and Agrawal [189] with the concept of *Generalized Association Rules (GAR).* The authors proposed taxonomies of mined data in order to generalize/specify rules. Later, taxonomies were used for item representation in General Impressions [136] or in the computing process of Item-Relatedness measure [182]. At the same time, one of the first papers to underline the utility of domain knowledge proposed also to use a taxonomy-like structure for knowledge representation. Anand *et al.* 1995 [8] suggested that the user should provide two kinds of information: domain knowledge and bias information (basic information used to guide the search). Domain knowledge is represented in three different types of structures: *Hierarchical Generalization Trees* (HG-Trees), *Attribute Relationship Rules* (AR-rules) and *Environment Based Constraints* (EBC).

Related to Generalized Association Rules, several approaches were proposed (i.e. by Won *et al.* 2006 [216]), but one of the most important suggests the notion of *raising*, developed by Chen *et al.* [53, 225]. Raising is a generalizing-like operation increasing support by keeping confidence high enough. This allows for strong rules to be discovered and also to obtain sufficient support for rules that, before raising, would not have minimum support due to the particular items they referred to. As *GARs*, this approach uses a taxonomy of items to generalize. The difference with *GARs* is that this solution proposes to use a specific level for raising and mining.

Very close to approaches developed in [189, 201], the $\mathcal{GART}$ algorithm proposed by Domingues and Rezende 2005 [58], is more user-oriented than the previous ones by proposing to the user a set of discovered association rules so that he/she could design one (or more) taxonomies of items. Recursively, taxonomies are applied over sets of consequent-equal rules with the purpose to group the generalized rules.

Pilot studies was done for integrating ontologies in the extraction of association rules with no further detailed results [168, 169]. One of the first works on ontology-driven KDD was developed by Phillips and Buchanan 2001 [164] and it proposed to use prolog ontologies. Later, a Pattern Base Management System is introduced by Kotsifakos *et al.* [125], based on a mixture of data mining process and domain ontologies for both user knowledge description and pattern storage. The selection of interesting association rules is particular, all the items of a rule should come from the same class. This is very interesting because, while a great part of subjective approaches are interested in finding unexpected knowledge, this approach is based on the idea that a rule is interesting *only if* it is conforming with the prior knowledge.

Cespivova *et al.* 2004 [52] outlined that the ontologies could be used in each step of the KDD process defined with the CRISP-DM[1] model. The role of ontologies is based on the given mining task and method, and on data characteristics. Next, the authors studied, and developed later in [201], the use of medical ontologies and other background knowledge into the process of association mining (rules are extracted using the *4ft-Miner* procedure [175]). They use UMLS (Unified Medical Language System) which provides terminological ontology to semantically group up variables. Then, the grouped variables can decompose general mining task into more specific tasks. Presumably, the mined hypotheses entering the evaluation phase will be smaller and homogeneous, hence easier to examine for a human evaluator.

A very recent approach proposed by Bellandi *et al.* 2007 [21] and developed later in [22], uses ontologies in a *pre-processing step.* Several domain-specific and user-defined constraints are introduced, grouped into two types: *pruning constraints*, meant to filter uninteresting items, and *abstraction constraints* permitting the generalization of items toward ontology concepts. The dataset is first pre-processed according to the constraints extracted from the ontology and then the data mining step takes place. The advantage of the pruning constraints is that it permits to exclude from the start the information that the user is not interested in, thus permitting to apply the *Apriori* algorithm to this new database. Let us consider that the user is not sure about which items he/she should prune. In this case he/she should create several pruning tests, and for each test he/she will have to apply the *Apriori* algorithm whose execution time is very high.

Onto4AR is a new constraint-based algorithm for association mining proposed by Antunes 2007 [11] and revised later in [12], where taxonomical and non-taxonomical (not *is-a* relation-based) constraints are defined over an item ontology. This approach is interesting in the way that the ontology offers a high level of expression for the constraints. Nevertheless, the proposed constraints are relation-based constraints, reminding the item-relatedness measure studied in [182]; the difference comes from the fact that, in this new approach, the distance is computed by using the entire set of onotological relations (*is-a* and data/object properties).

[1]CRoss Industry Standard Process for Data Mining, http://www.crisp-dm.org

The KEOPS methodology was developed by Brisson *et al.* [39, 39, 39]. It aims to guide the process of user knowledge integration in the data mining process. First, the user knowledge regarding the discovered association rules is represented by means of a rule-like formalism. Their main goal is to select the most interesting rules according to user knowledge. KEOPS approach integrates also an ontology which is used in the pre-processing phase, in order to prepare the database for mining.

4.6 Post-Processing Techniques vs. Measures-Embedded Algorithms

In the previous section, we focused over the reduction of the number of rules by means of interestingness measures and we presented a large survey on this field – on the one hand, concerning objective measures, and on the other hand, the subjective ones. For the latter, we outlined the importance of user and domain knowledge in the extraction of interesting knowledge. Nevertheless, an interesting question arises: *Which technique is more efficient: to apply the interestingness measures in a post-processing step, or to integrate them in the mining algorithm?*.

4.6.1 Post-Processing Techniques

The idea of selecting *good* rules during a post-processing step appeared as early as 1998, when Imielinski and Virmani [114] defined *rule post-processing* as a process which *involves selecting rules which are relevant or interesting, building applications which use the rules and finally, combining rules together to form a larger and more meaningful statements.* Later, in 2000, the interest in post-processing methods increased, and they were grouped as follows: *knowledge filtering* – rule truncation and rule post-pruning, *interpretation and explanation* of the acquired knowledge, *evaluation* of the mining algorithm, or *knowledge integration* [41]. But, the most accurate classification – here, with the sens of definition of post-processing tasks – was done by Baesens *et al.* 2000 [15]. The authors suggested that post-processing consists of different techniques that can be used independently or together: *pruning*, *summarizing*, *grouping* and *visualization*.

Pruning means removal of rules generated as effect of anomalies or unwanted phenomena or rules that are simply not interesting to the user. In this category we can find different approaches. Interestingness measures applied after the rule extraction are pruning techniques: objective measures – i.e. lift [36] or implication intensity [86] – remove those rules not satisfying the measure threshold, and subjective measures generally prune those rules that are not conforming with the user knowledge [7, 136]. Redundancy reduction is a well known technique of rule pruning, and different methods were proposed as subsumed rules (i.e. minimum improvement constraint [20]), transitivity of rules, or circular chains.

Summarizing generates a more compact representation of discovered knowledge by using more general or abstract concepts which are more comprehensive for the expert. Different techniques exist to summarize rules among which we remind the use of hierarchical structures and direction setting [134].

The former consists of those techniques integrating taxonomies or ontologies with the aim to generalize the rules and to summarize them [58, 189, 225]. These techniques were already presented in Section 4.5 on page 89, thus we will focus on the latter one.

The concept *direction setting (DS) rules* was proposed by Liu *et al.* 1999 [134] in order to define the set of rules which defines a behavior for the entire set of discovered rules. They represent the essential relationships (or the skeleton) of the domain, and more details are given by *non-DS* rules. To this end, Liu *et al.* first proposed to prune more specific rules by the use of chi-square test, and then, to summarize the remaining rules, generating DS rules.

Example 4.6.1 Let us consider that the following rules are discovered by classical techniques:

$R_1 : apple \rightarrow pork\ [S = 40\%,\ C = 80\%]$

$R_2 : pear \rightarrow pork\ [S = 30\%,\ C = 81\%]$

$R_3 : apple,\ pear \rightarrow pork\ [S = 22\%,\ C = 88\%].$

If we consider that chi-square test shows a positive correlation between *apple* and *pork*, and between *pear* and *pork*, this new approach considers that the rule R_3 does not bring new information to the set $R_1 + R_2$; moreover, this rule can be generated by combining the first two ones. Rules R_1 and R_2 are *DS-rules* because they show the direction (positive correlation) for the rule R_3. ■

In the same idea, Toivonen *et al.* 1995 introduced in [204] the notion of association *rule covers*. The notion of rule cover is defined as the subset of a rule set describing the same database transaction set as the rule set. The number of rules in a cover can be quite small. A greedy algorithm is proposed to find a good cover and the remaining rules are pruned. The problem with this method is that the advantage of association rules, its completeness, is lost.

Using clustering, based on economic assessment or time based patterns, *grouping* the rules helps the user to focus on the group of rules that is more interesting. In [224], Zhao *et al.* use the clustering algorithms in order to group the items and to compute distances between them. The notion of *subsumed rules*, introduced by Bayardo *et al.* 1999 [19], describes a set of rules having the same consequent and several additional conditions in the antecedent regarding another rule.

Visualization proposes specific graphical representations of discovered knowledge with the main interest of helping experts to find useful patterns. Several important elements compose this type of techniques as follows: first, the representation model should be comprehensible for the user, but also it should be able to group a hight level of information. Second, the results of the visualization technique are more significant if it is interactive and iterative – the user should be able to execute iteratively different actions over the discovered rules. In [31], Blanchard *et al.* 2007 present a 3D framework – *ARViZ* – for user-guided local mining. Another proposition of framework is made by Bruzzese and Davino 2003 [42]; the visual framework is proposed for user-guided post-treatment of discovered rules, very interesting from the point of view of graphical representation of rule support. A wide survey over the visualization techniques in data mining is proposed by Ben Said *et al.* 2010 [23].

The post-processing step is definitely necessary to the KDD process; however, generating a great number of rules and then pruning or grouping most of them is not very efficient. It has been suggested by Silberschatz and Tuzhilin 1995 [183] that a more practical approach would be to include the selection process of interesting rules in the actual rule mining process, such that the primary output already contains only the rules that are interesting to the user.

This new method and the previous one have their advantages and inconveniences. The main advantage of the post-processing methods is that working on a complete set of association rules allows to have a general view over the extracted rules, and then, a user could be able to apply different interesting methods of reduction of number of rules. The main inconvenience of post-processing techniques comes once with the necessity of generating the whole set of rules. This is due to the fact that extracting the entire set of rules takes a lot of resources, and in certain cases (i.e. low support) it is an intractable task. Next, let us consider that generating millions of rules is possible; the second main inconvenience is the impossibility to work on a huge set of rules correctly (millions of rules). This cannot be an interactive process due to execution time explosion. It is also important to note that, several post-processing techniques cannot work efficiently with a partial set of rules.

4.6.2 Measures-Embedded Algorithms

Concerning the methods that integrate interestingness measures in mining algorithm such as [28, 155], their main advantage is that they propose as result a set of rules highly limited in volume comparing to the number of rules delivered by the classical techniques. Nevertheless, their main inconvenience comes from the impossibility to integrate certain measures in the mining algorithm. For instance, to integrate interestingness measures in the *Apriori* algorithm they should satisfy the downward closure property. That is way, in the case of measure-based mining, and more specially, in the case of user-based mining, it is more interesting to use local mining techniques as it was proposed by Blanchard *et al.* 2003 [28].

In Section 4.2 on page 59 we presented in Figure 4.1 and Figure 4.2 different frameworks of knowledge discovery process. Liu *et al.* 1999 [135] proposed a new type for the KDD framework. In contradiction with the former ones, in Figure 4.9 we can note that Liu *et al.* suggest a new step in the KDD process, after the pattern extraction, consisting of tools that allow the user to rank the patterns.

It is important to note that this is the first real proposition of a framework integrating the user in the KDD process. However, we consider this point of view quite limited because it does not permit user-driven mining process. Therefore, we propose a new framework for the knowledge discovery process presented in Figure 4.10. The particularity of this framework resides in the implication of the user in the two processes: data mining and pattern analysis.

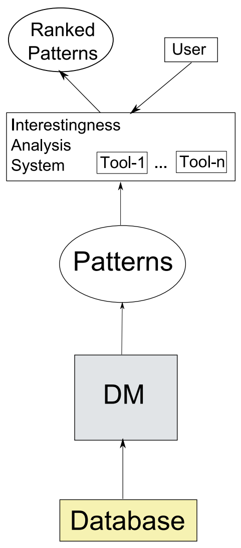


FIGURE 4.9: A new proposition for knowledge discovery process [135].

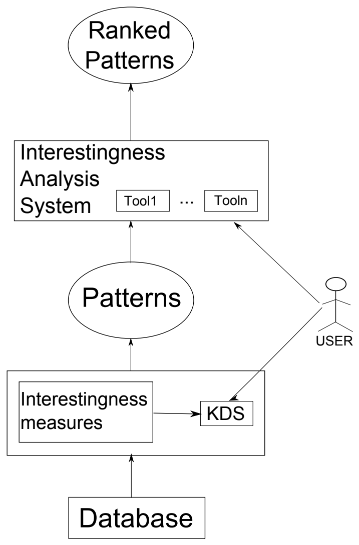


FIGURE 4.10: A new framework for knowledge discovery process.

4.7 Conclusion

In this Chapter, we studied the integration of interestingness measures in the association rule mining technique. First, we discussed objective (data-driven) measures which are defined by statistical and descriptive functions over the data. Nevertheless, the extracted rules are still in an important number, and, more important, the objective measures do not guarantee the pruning of non trivial rules. Subjective measures come to complete the objective measures. They are user-driven, based on the goals and beliefs of users.

While a great part of these approaches acts in the post-processing step by filtering interesting rules, specific approaches, such as Logical Contradiction, integrate user knowledge in the mining algorithm.

The main limitations of these techniques come from different aspects:

- volume and quality of filtered rules – the existing techniques do not guarantee an important reduction of rule number to a maximum rule number in order to be possible for a user to analyze them manually. Further, it is still possible to find some trivial rules, depending on the user. The reasons are the representation formalisms and the matching methods used in the filtering approaches.

  Using templates or rule-like formalisms could be interesting, but the language is not enough powerful to express the entire human reasoning process. Later, ontologies were proposed as domain knowledge representation, but they are not used to their entire capacity of representation. In other words, the restriction and reasoning techniques are not integrated in these techniques, even if they represent the difference between ontologies and the other hierarchy-based representation formalisms.

  In other words, the more the user domain knowledge/expectations are represented in a accurate and flexible manner, the more the filtering process is efficient, and the selected rules correspond to user knowledge and they are generated in a limited number.

- interactivity of the mining process and completeness of the mining framework – the achievement of the mining process also depends on the possibilities that the mining framework offers to the user. That is to say that, in order to filter the rules that interest him/her, the user should be able to combine objective measures with subjective ones and to refine his/her knowledge in function of partial results.

5
AR*IPSO* and AR*LIUS* Approaches

Every addition to true knowledge is
an addition to human power.

Horace Mann

Contents

5.1 Introduction

This Chapter presents AR*IPSO* (Association Rule Interactive Post-processing using Rule Schemas and Ontologies) and AR*LIUS* (Association Rule Local Interactive mining Using rule Schemas), our novative approaches for extracting reduced number of association rules. Our main motivation comes from the problem itself: we started this work with the principal purpose to transform the large and inextricable set of discovered rules in an easier analyzable one by reducing the number of rules and increasing their quality. To this end, we oriented our approach toward the user in order to help the discovery process to select only those pertinent rules. For this reason, we consider user knowledge integration in KDD process as very valuable.

First, we introduce the AR*IPSO* approach which works in the post-processing step of the KDD process. The novelty of our approach relies on supervising the association rule mining process using two different conceptual structures to represent user domain knowledge – one or several *Ontologies* – and user expectations – several *Rule Schemas* generalizing *general impressions*. In addition, the approach integrates *Operators* over Rule Schemas that aim to produce an well defined action over the set of association rules. Finally, AR*IPSO* framework provides an interactive and iterative post-mining process, which facilitates the analyzing task.

Second, we detail AR*LIUS* approach which is different from the former because it concentrates on the mining algorithm; it does not consist in a post-processing approach. It proposes a new algorithm for association rule mining integrating user knowledge by means of *Rule Schemas*. The formalism developed in this method is different from the one proposed in AR*IPSO* – it is more flexible and more expressive. Beyond the Rule Schema formalism, the novelty of this approach stands on a new set of Operators and on a new mining technique proposing a local rule mining. The main interest relies on the fact that the number of discovered rules during a research is low, and that partial results can be injected in the mining algorithm.

5.2 AR*IPSO* Approach

AR*IPSO* is an association rule filtering approach. It proposes to select only the association rules that are interesting for the user. AR*IPSO* works in the post-processing step of the KDD process. In this context, KDD process is executed in two steps: first, association rules are generated by classical techniques; next, AR*IPSO* selects only the interesting ones.

AR*IPSO* approach is composed of two main parts as shown in Figure 5.1. In a first instance, we focus on user knowledge. This part consists in a knowledge base allowing to formalize user prior knowledge. Three semantic components are integrated in the approach:

- Domain;
- Expectation;
- Action.

First, we integrate *user domain knowledge*. The user has several information concerning the domain of the database. Moreover, if the user is a domain expert, he/she should hold a complete description of the domain. Thus, user domain knowledge is integrated in our AR*IPSO* approach offering a subjective general view over the database domain.

Secondly, we integrate *user expectations*. The user might also have several information concerning the discovered rules, he/she might know the type of rules he/she wants or does not want to find. Thus, we suggest to the user to provide his/her expectations and goals representing his/her prior knowledge over the discovered rules.

Last but not least, in our approach, we give the possibility to the user to apply different *operators* over his/her expectations. Let us consider that an user knows already that a set of associations exists in the database. As these associations are trivial for the user, they should be pruned from the final set of rules. On the contrary, rules contradicting this set of associations could interest the user, having a surprise effect. For this purpose, we propose to the user to assign to each expectation an action to be taken when applied over the set of rules. In consequence, user expectations work as a filter over the entier set of rules.

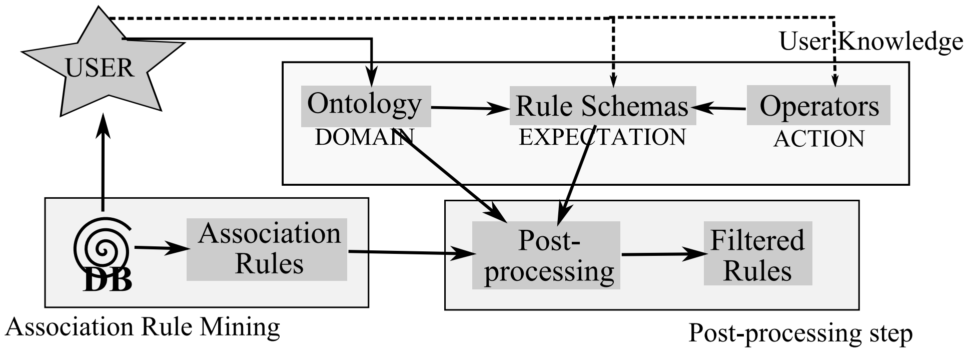


FIGURE 5.1: AR*IPSO* approach description.

The second part of our approach refers to the post-processing task. As already stated in the literature and in this manuscript, the post-processing task deals with analyzing of entire sets of association rules by using different interestingness measures, with the main goal to propose to the user a rule set highly limited in volume. In our approach, this process consists in applying iteratively a set of filters over the extracted rules in order to propose in the end of this task a small set (less than 100 rules) of rules interesting to the user. Described in the sections below, this set of filters is composed of new filters that we propose – the rule schema filters/pruning created by applying operators over rule schemas, or filters already proposed in the literature – additional filters – discussed in Section 5.2.4.3.

5.2.1 Motivations

The main motivations of our AR*IPSO* approach are described by the drawbacks of the association rule mining process outlined in Chapter 2 and Chapter 4, such as:

- the huge volume of rules, and
- the quality of rules.

The assessment of the quality of rules is both an objective and subjective problem. As outlined in Chapter 4, a great number of objective measures were proposed in the literature which are able to measure the quality/interest of discovered rules and, thus, to reduce the rule number.

Nevertheless, in this context, the user plays an important role since a rule can be interesting to an user, but not interesting to another one. This depends on user background knowledge and expectations. Subjective measures propose to integrate user knowledge in the association rule mining process and different approaches were developed. In this context, the user beliefs should be represented by means of accurate and flexible formalisms in order to express in the best way what the user knows and/or expects; thus, the selection process is more efficient.

The actual user-driven approaches suggest different formalisms for knowledge representation starting from templates, rules, taxonomies, till ontologies. As stated in Chapter 3, ontologies represent an accurate and flexible formalism for the representation of user knowledge, due to restriction and reasoning techniques. Ontologies propose to express the reasoning behind the human thinking which completes the descriptive part. Nevertheless, the few approaches using ontologies for knowledge representation do not benefit of this great advantage of ontologies – the reasoning.

Further, we could notice also that existing mining frameworks do not combine different types of interestingness measures and/or different types of knowledge representation formalisms. This could permit to generate reduced sets of association rules and could increase the rule quality. In this context, it is important to develop a complete, interactive and iterative framework allowing the user to test different scenarios till he/she arrives to the researched information.

Since early 2000's, in the Semantic Web field, the number of available ontologies has been increasing covering a wide domain of applications. This is a great advantage for a system integrating ontology-based user knowledge representations because it can limit the ontology development time.

One of our most important contributions on reducing the number of association rules relies on using ontologies as user background knowledge representation. That is to say that we extend the specification language proposed by Liu *et al.* [135] - General Impressions (GI), Reasonably Precise Concepts (RPC) and Precise Knowledge (PK), denoted in a general manner by General Impressions - by the use of ontology concepts. For now on we will refer to this specification language using the term *general impressions*.

First, let us remind one of the specifications language proposed by Liu *et al.*: the Reasonably Precise Concept one. A *RPC* represents user vague feelings concerning some associations in the database, including also the direction of the implication. The formalism of RPCs is described as follows:

$$rpc(< S_1, \ldots, S_j \rightarrow S_{j+1}, \ldots, S_m >)[support,\ confidence]$$

where each S_i is an element of the taxonomy (i.e. the taxonomy in Figure 5.2) and $*$, $+$, $\{\}$ is a set of operators that can be applied over the elements. S_i+ represents one or more occurrences of the elements of S_i, S_i* represents zero or more occurrences of the elements of S_i, $\{S_{i_1}, S_{i_2}\}$ expresses S_{i_1} OR S_{i_2}.

The *RPC*-association rule matching process consists in a syntactic comparison between elements involving the item taxonomy. Thus, each element in the *RPC* should find a correspondent in the association rule. Inversely, each item in the association rule, should be a correspondent of an element from the *RPC*.

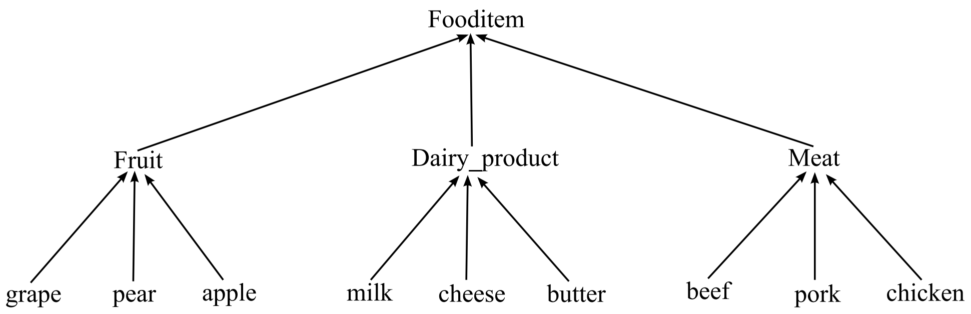


FIGURE 5.2: Supermarket item taxonomy [136].

Example 5.2.1 Let us consider that the user might think that if somebody buys *milk OR cheese* and *beef*, then it is probable that she/he will also buy *Fruit items* (assume the user uses the taxonomy in Figure 5.2). In consequence, using Reasonably Precise Concepts, the user could specify this belief as following:

$$rpc(<\{milk,\ cheese\}*,\ beef \to Fruit+>)$$

The following rules are examples of association rules that are conform to the specification:

$beef \to apple$
$milk,\ beef \to grape,\ pear$ ■

Example 5.2.2 In this second example, let us consider that the user would need to find if customers buying diet products, also buy ecological ones. When using the Liu *et al.* proposition we can make two major observations. On the one hand, we can notice that the user does not have the necessary tools to find which products are proposed in diets or which ones are ecological. On the other hand, no possibility of extension of the existing taxonomy is proposed by the authors so that the user could instantiate the different types of products.

Nevertheless, the user has the possibility to propose a *RPC* by using directly the database items. Let us consider that the *diet products* are *apple* and *chicken* items, and *milk*, *grape* and *beef* items are *ecological products*. Thus, the user could propose the following *RPC* described using the *OR* operator:

$$rpc(<\{apple,\ chicken\} \to \{milk,\ grape,\ beef\}>)$$

In the given example the number of items is not very high, so it is still possible to enumerate an entire subset of items. But, what if we use a database of hundreds of items? It is clear that working directly with database items is not a solution for the user.

If it was possible for the user to extend the taxonomy, it would be able to create an *RPC* such as:

$$rpc(< DietProducts \rightarrow EcologicalProducts >)$$

where *DietProducts* and *EcologicalProducts* represent, respectively, the set of the products integrated in diets, and those products which are produced in an ecological way.

It is important to outline that the two concepts of *diet products* and *ecological products* do not contain information already existing in the database. So, being able to create such a *RPC* consists also of being able to propose additional information comparing to those proposed by the database which will permit to the user to treat the data from another level. Defining such concepts is not possible using taxonomies.

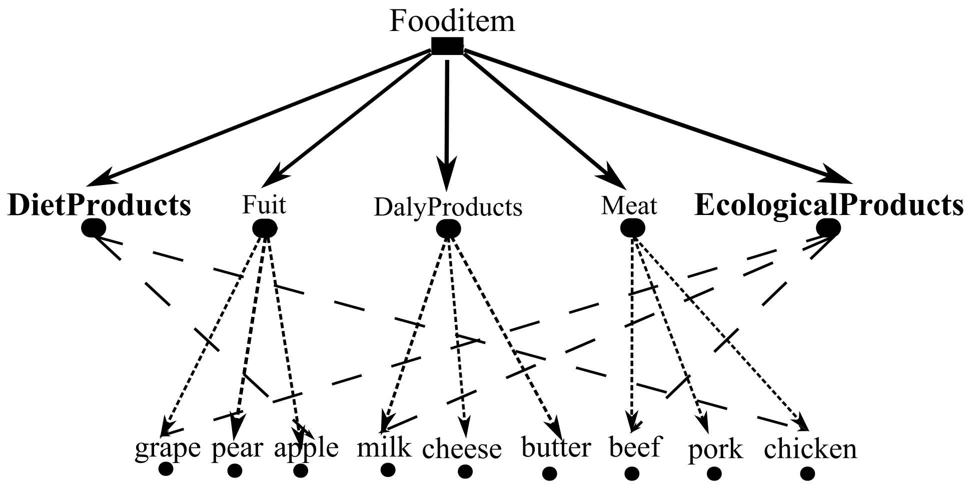


FIGURE 5.3: Visualization of the ontology created based on the supermarket item taxonomy.

This type of information could be integrated in ontologies. Used in designing ontologies, description logic [108] allows concept to be defined using restrictions. In consequence, based on the earlier considerations and starting from the taxonomy presented in Figure 5.2, we developed an ontology presented in Figure 5.3. We propose to integrate two data properties of Boolean type in order to define two types of products:

- those that are useful in diets (*IsDiet*) and
- those that are ecological (*IsEcological*).

And, using these two properties, the concept *DietProducts* is defined as a restriction on *FoodItem* hierarchy using the data property *isDiet*, describing the items useful in a diet. Similarly, the *EcologicalProducts* concept is defined.

In Figure 5.3, we can notice that the definition proposed of the two concepts is followed; so, *apple* and *chicken* are instances of *DietProducts* concept, and *milk*, *grape* and *beef* are instances of *EcologicalProducts* concept. ■

5.2.2 Ontologies – User Domain Knowledge

5.2.2.1 Ontology Structure

Domain knowledge, defined as the user knowledge concerning the database, is described in our framework using ontologies.

Compared to taxonomies used in the specification language proposed in [136], ontologies offer a more complex knowledge representation model by extending the only *is-a* relation presented in a taxonomy with the set R of relations. In addition, the axioms bring important improvements permitting concept definition starting from existing information in the ontology permitting to infer new information by means of reasoning process.

In order to explain the interest of ontologies in our approach, let us present the structure of the ontology and the key concepts that our method is based on. For this purpose, we consider three types of concepts:

- *leaf-concepts*;
- *generalized concepts* created by the use of the subsumption relation ($\leq$) in H of O;
- *restriction concepts* created by using restrictions over the other concepts and properties.

The particularity of these concepts is that leaf-concepts and generalized concepts are common to taxonomies and ontologies, while restriction concepts are particular to ontologies. In order to proceed to the definition of each type of concepts, let us remind that a set of items in a database is defined as $I = \{i_1,\ i_2,\ ...,\ i_n\}$ and that the notation $\leq$ stands for the subsuming relation.

Definition 5.2.3

*A concept is denoted as **leaf-concept** (LC) if it does not subsume other concepts in the ontology. Formally, it is defined as follows:*

$$LC = \{C_0 \in C \mid \nexists C' \in C,\ C' \leq C_0\}.\ ■$$

The Figure 5.4 presents the set of leaf-concepts in the supermarket ontology.

Definition 5.2.4

*A concept is denoted as **generalized concept** (GC) if it subsumes at least one concept in the ontology. Formally, a generalized concept is defined as follows:*

$$GC = \{C_1 \in C \mid \exists C' \in C,\ C' \leq C_1\}.\ ■$$

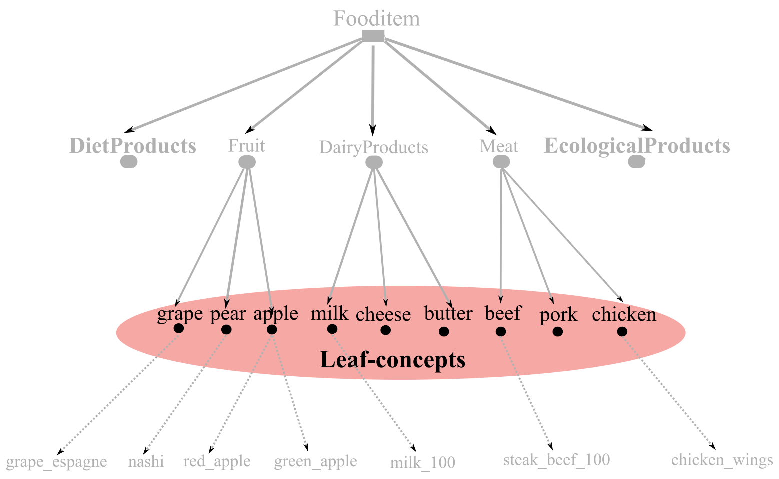


FIGURE 5.4: Example of *leaf-concepts* in supermarket ontology.

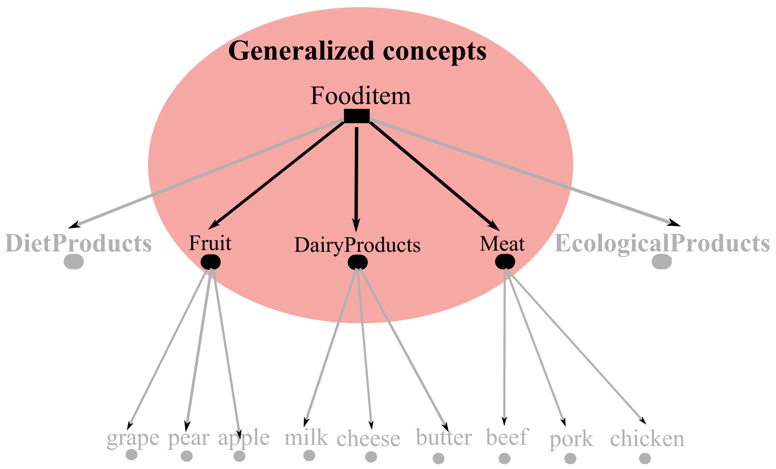


FIGURE 5.5: Example of *generalized concepts* in supermarket ontology.

Example 5.2.5 In the Figure 5.5 we can remark that the concepts *Fooditem*, *Fruit*, *DairyProducts* and *Meat* are generalized concepts. Let us consider the *DairyProducts* concept. It subsumes the following concepts: *milk*, *cheese* and *butter*. Thus, we can formalize that: $milk \leq DairyProducts$, $cheese \leq DairyProducts$ and $butter \leq DairyProducts$. ■

Definition 5.2.6

A concept is denoted as a ***restriction concept*** *(RC) – or defined concept in OWL language – if it is described using a set of restrictions.*

Thus, we can define the set of restriction concepts as those concepts described using a set of restrictions over the properties and over the rest of the concepts as *follows:*

$$RC = \{C_2 \in C \wedge C_2 = \bigcup_{i=1}^{k} R_i \mid R_i = (O_i, P_i, C_i/I_i/D_i)\}$$

where R_i represents a restriction over the set of properties and the set of concepts/ instances/data in the ontology. It can be generally defined as:

$$R = \text{Restriction\_Operator}(O_i)\text{Property}(P_i)\text{Concept/Instance/Data}(C_i/I_i/D_i)$$

where O_i stands for the restriction operator, P_i for property, C_i for the concepts, I_i for the instances and D_i for the data. ■

A restriction describes a constraint on relationships within the individuals participates for a given property. Let us remind that three types of restrictions are available as follows [104] (for more details see Section 3.4.3 on page 47): *Quantifier Restrictions* (*existential quantifier* (∃) and *universal quantifier* (∀)), *hasValue Restrictions (∍)* and *Cardinality Restrictions.*

The main interest in proposing the use of restriction concepts in our ontology is that these concepts do not contain the usual information that we find in the database. An important improvement in information quantity is brought by the use of restriction concepts. They allow the user to create concepts that can be used in the filtering step.

Example 5.2.7 In order to exemplify restriction concepts we will use the *hasValue* restriction to create our restriction concepts. First, let us consider that we integrate 2 data properties in our ontology:

- *Fooditem.isDiet()* is a boolean property which denotes if a product is a diet one or not. It connects the *FoodItem* concept to boolean data ($TRUE$ or $FALSE$);
- *Fooditem.isEcological()* is a boolean property which denotes if a product is produced in ecological ways or not (the same structure as for *isDiet*).

Based on these data properties, we are able now to define two restriction concepts which describes respectively the individuals related to the boolean data $TRUE$ by the *isDiet* property, and by the *isEcological* property:

$$DietProducts \equiv FoodItems \sqcap isDiet\ hasValue\ TRUE$$

$$EcologicalProducts \equiv FoodItems \sqcap isEcological\ hasValue\ TRUE$$ ■

The restriction concepts *DietProducts* and *EcologicalProducts* are reproduced in Figure 5.6.

Last but not least, the instances (I) of the ontology are defined as the individuals of the concepts. For instance, *apple* concept has two individuals: *red_apple* and *green_apple*. They can be viewed in the bottom part of the Figure 5.6.

5.2.2.2 Ontology Reasoning

In the previous section we propose to classify the concepts of the ontology in: leaf, generalized and restriction concepts. The first two are quite natural because they describe the taxonomy structure, while the last one is more difficult to understand and to implement.

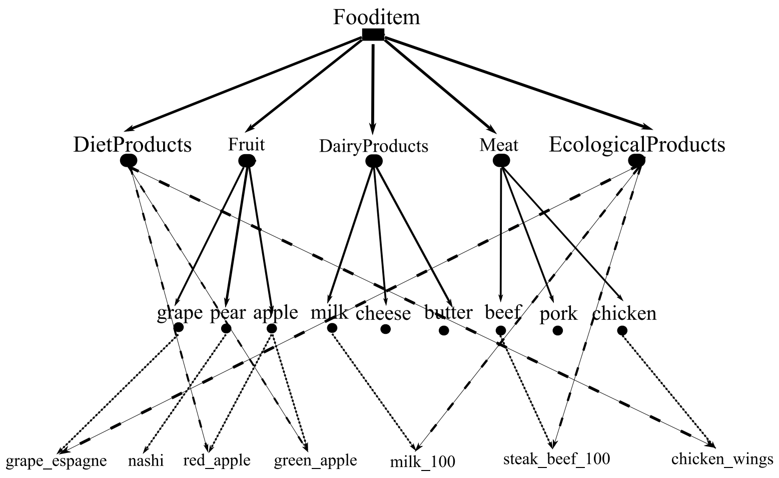


FIGURE 5.6: Example of *restriction concepts* and of the ontolgy structure after inference.

The advantage of restriction concepts comes from the main advantage of Description Logic – the reasoning process. As presented in Section 3.4.4 on page 53, reasoners allow users to find inconsistencies over knowledge bases, and to generate new information. More precisely, in AR*IPSO* approach we are interested in three important actions of reasoners:

- finding inconsistencies;
- reorganizing the concepts;
- creating the corresponding relations between individuals and concepts.

First, finding inconsistencies is a classical task which informs the user concerning important anomalies in the ontology.

Second, reorganizing concepts is a more delicate task.

Example 5.2.8 For instance, let us consider that in the previous example we add a new concept – *ProductTypes* – which subsumes the concepts *EcologicalProducts* and *DietProducts*, but which is in no relation with the *Fooditem* concept. After the reasoning, it is obvious that *EcologicalProducts* and *DietProducts* concepts will be also subsumed by the *Fooditem* concept. ■

Third, creating new relations between the individuals and the concepts is a crucial feature proposed by the reasoners. During this task, the real interest of restriction concepts can be evaluated. In the asserted state (before reasoning) we consider that a set of individuals is asserted in the ontology by the user such as the restriction concepts. After reasoning the ontology is able to answer which individuals are instances of a given restriction concept. In other words, the reasoning can to provide to a given restriction concepts the individuals that satisfy the corresponding restrictions.

Example 5.2.9 For instance, in our example, the Figure 5.6 represents the ontology after reasoning. We will consider that *red_apple*, *green_apple* and *chicken_wings* individuals have the *isDiet* property on the boolean *TRUE*. In consequence, after reasoning, these three individuals will be individuals of the *DietProducts* concept also, because they fulfill its restrictions. ■

5.2.2.3 Ontology-Database Mapping

In this scenario, it is fundamental to connect ontology concepts C to the database, each one of them being connected to one/several items of I. The direct connection is realized by the bottom elements of the ontology. Thus, if the ontology is instantiated, the direct connection is done through instances, other way, through leaf-concepts.

For the case of an instantiated ontology, the instances are connected in the easiest way to the database:

$$\begin{aligned}&f'_0 : I_{nst} \to I,\ \forall i_0 \in I_{nst}\\ &\qquad \exists i \in I,\ i = f'_0(i_0).\end{aligned}$$

And, in this case, leaf-concepts are connected to the database through its instances as follows:

$$\begin{aligned}&f_0 : C_0 \to \mathcal{P}(I),\ \forall c_0 \in C_0\\ &\qquad f_0(c_0) = \bigcup_{i_0 \in I_{nst}} \{i = f'_0(i_0) \mid i_0 \text{ instance of } c_0,\ i \in I\}.\end{aligned}$$

For the cases when the ontology is not instantiated, leaf-concepts are connected in the easiest way to database – each concept from LC is associated to one item in the database through the function f_0:

$$\begin{aligned}&f_0 : C_0 \to I,\ \forall c_0 \in C_0\\ &\qquad \exists i \in I,\ i = f_0(c_0).\end{aligned}$$

A generalized concept is connected to the database through its subsumed concepts. That means that, recursively, only the leaf-concepts subsumed by the generalized concept contribute to its database connection:

$$\begin{aligned}&f : C_1 \to \mathcal{P}(I),\ \forall c_1 \in C_1\\ &\qquad f(c_1) = \bigcup_{c_0 \in C_0 \wedge c_0 \leq c_1} \{i = f_0(c_0)\}.\end{aligned}$$

The connection of restriction concepts (in the case of an instantiated ontology) is quite comparable with the connection of leaf-concepts. Let us consider that after the inference process a restriction concept contains a set of individuals, $I_{nst_{RC}}$, that satisfy the restrictions. The connection to the database is done through this set of individual, as in the case of leaf-concepts.

Example 5.2.10 Let us consider the database presented in Figure 5.7 and described by three transactions. Also, let us consider the ontology presented in the same figure as being the ontology developed over items of database and described as follows.

The ontology is defined by the concepts {*FoodItems, Fruits, DairyProducts, Meat, DietProducts, EcologicProducts,* . . . } which are organized as follows:

Leaf Concepts: {*grape, pear, apple, milk, cheese, butter, beef, chicken, pork*}

Generalized Concepts: {*Fruits, DairyProducts, Meat, FoodItem*}

Restriction Concepts: {*DietProducts, EcologicalProducts*}

and by the following instances:

Instances: {*grape_espagne, nashi, red_apple, green_apple,* . . . }.

Two data properties are also integrated in order to define whether a product is useful for a diet, or is ecological. For example, the *DietProduct* restriction concept is described using description logics language by:

$DietProducts \equiv FoodItems \sqcap \exists isDiet.TRUE$

defining all food items that have the boolean property *isDiet* on *TRUE*. For our example *isDiet* is instantiated as follows:

isDiet: {*(red_apple, TRUE), (green_apple, TRUE), (chicken_wings, TRUE)*}.

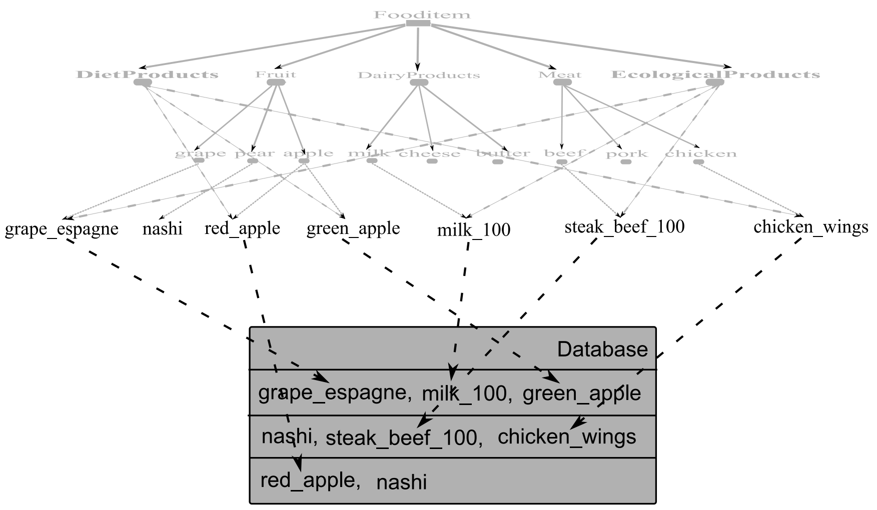


FIGURE 5.7: Ontology-database mapping trough instances.

Now, we are able to connect the ontology and the database. As already presented, instances are connected to items in a very simple way, for example, the instance *green_apple* is connected to the same item f'_0*(green_apple) = green_apple*. Further, the *apple* leaf-concept is connected to the database via its instances such as:

$$\begin{aligned} f_0(apple) &= \{f'_0(red\_apple), f'_0(green\_apple)\} \\ &= \{red\_apple, green\_apple\}. \end{aligned}$$

On the contrary, the generalized-concept $Fruits$ is connected through its three subsumed concepts:

$$\begin{aligned} f(Fruits) &= \{f_0(grape), f_0(pear), f_0(apple)\} \\ &= \{f'_0(grape\_espagne), f'_0(nashi), f'_0(red\_apple), f'_0(green\_apple)\} \\ &= \{grape\_espagne, nashi, red\_apple, green\_apple\}. \end{aligned}$$

Similarly, we can describe the connection for the other concepts.

More interesting, the *DietProducts* restriction concept will be connected through those instances satisfying the restrictions in the definition of the concept. Thus, *DietProducts* is connected through the instances *red_apple*, *green_apple* and *chicken_wings*: *f(DietProducts)* = { *red_apple, green_apple, chicken_wings* }. ■

5.2.3 Rule Schemas – User Expectations

The model of user expectations in AR*IPSO* is based on the proposition made by Liu *et al.* 1999 [135]. In this approach, Liu *et al.* developed general impressions in the view of helping the user to select interesting rules. The general impressions come as a representation language especially for user vague feelings about generated rules.

The model proposed by Liu *et al.* is described by elements from an item taxonomy which develops an *is-a* organization of database attributes. Using taxonomies of items has many advantages: the items are organized in more general concepts and the representation of user expectations is more general, thus, the rules that are filtered are more interesting to the user. However, a taxonomy of items might not be enough to express heterogeneous knowledge as user domain knowledge. The mainly limits of taxonomies can be cited as follows: first, the concepts are connected using only the *is-a* relation, thus, the lack of different types of concept relations does not permit the diversification of types of concepts – only generalized concepts are available. Second, the lack of different relations between concepts does not permit to taxonomies to infer new information.

For example, the user might want to use concepts that are more expressive and accurate than the generalized concepts in taxonomies. Moreover, concepts resulting from relationships more precise than the *is-a* relation (i.e. *IsEcological*, *IsCookedWith*) are more useful. These two points describe the domain knowledge of a human being: humans are able to make a hight series of connections in a short time and to infer the present facts in order to generate new ones.

In the last decade, a lot of research was done in knowledge management field, and a part of it was stated in the Section 3. An important number of representation languages were proposed in order to formalize human knowledge, but, in the last years, ontologies were considered to be a complex language with a very well developed framework permitting the definition of an important type of elements.

Compared to a taxonomy, an ontology includes the features of taxonomies, but adds more representation power. In a taxonomy, the means for subject description consists essentially of one relationship: the subsumption relationship used to build the hierarchy. Thus, the set of items is opened, but the language used to describe them is closed [80]. In conclusion, a taxonomy is simply a hierarchical categorization or classification of items in a domain. On the contrary, an ontology is a specification of several characteristics of a domain, defined using an open vocabulary.

Starting from general impressions and from the ideas presented above, we propose a new representation language for user expectations replacing item taxonomies with ontologies based on database items. We called this new knowledge representation formalism *Rule Schemas (RS)*. Therefore, Rule Schemas bring the expressiveness of ontologies in the post-processing task of association rules combining not only item constraints, but also ontology concept constraints.

Going back to general impressions, it is difficult for a domain expert to know exactly the support and confidence thresholds for each rule schema proposed, because of their statistical definition. That is why we consider that using Precise Knowledge in user expectation representation might be useless. Thus, we propose to improve only two of the three representations introduced in [136]: General Impressions and Reasonably Precise Concepts.

Definition 5.2.11
We describe a Rule Schema as a set of ontology concepts expressing the fact that the user expects these concepts to be present in the extracted association rules. Formally, a Rule Schema is defined as:

$$RS(< X_1,\ ...,\ X_n\ (\rightarrow)\ Y_1,\ ...,\ Y_m >)$$

where $X_i,\ Y_j \in C$ *of* $O = \{C,\ R,\ I,\ H,\ A\}$. *The optional implication "→" outlines that the proposed formalism combines General Impressions and Reasonably Precise Concepts. On the one hand, if we use the formalism as an implication, an implicative Rule Schema is defined extending the RPCs. On the other hand, if we remove the implication, we define non implicative Rule Schemas, generalizing GIs. Moreover, we propose the use of several operators over the elements of the rule schemas as follows:*

- *OR is a logical operator which can be used as* S_1 OR S_2 *– the concept* S_1 *or the concept* S_2*;*
- ** operator is used to express that an element can be present in the rule schema zero or more times;*
- *+ operator is used to express that an element can be present in the rule schema one or more times.* ■

Example 5.2.12 Let us consider the ontology presented in Figure 5.3. Using the concepts of the ontology, we can define the following Rule Schemas:

$RS_1 : RS(< DietProducts,\ EcologicalProducts >)$

$RS_2 : RS(< DietProducts+ \rightarrow EcologicalProducts >)$

In the first Rule Schema, the user knows that there is a connection between the two concepts, meanwhile in the second one the user knows exactly that he/she expects to find in the discovered rules one or more diet products in the antecedent and one ecological product in the consequent. ■

The main goal of a Rule Schema is to filter and to prune discovered association rules. To this end, we propose a set of operators to be applied over the Rule Schemas in order to realize several types of actions. We will detail operators in Section 5.2.4.2.

In conclusion, in a rule-like formalism, a rule schema describes, the user expectations in terms of interesting/obvious rules. As a result, Rule Schemas act as a rule grouping, defining rule families.

5.2.4 User Interactivity – Action

In post-processing data mining frameworks, and more particularly in association rule field, it is difficult for an user to find in only one search all interesting knowledge discovered by the mining process. Thus, an important contribution of our work is to propose an interactive approach with the possibility for the user to come back over his/her decisions/actions to finally find the knowledge researched.

First, the interactivity of our approach comes from the process itself, designed so that the user has the liberty to choose the best action to take after testing an entire set. Second, actions proposed via the rule schema operators increase the degree of user interactivity, giving to the user an great set of actions to take.

5.2.4.1 Interactive Process

The interactive process of the AR*IPSO* framework (presented in Figure 5.8) aims at guiding the user through the post-processing phase. Taking into account his/her feedbacks, the user is able to revise his/her expectations or actions to take in function of intermediate results. Several steps are suggested as follows:

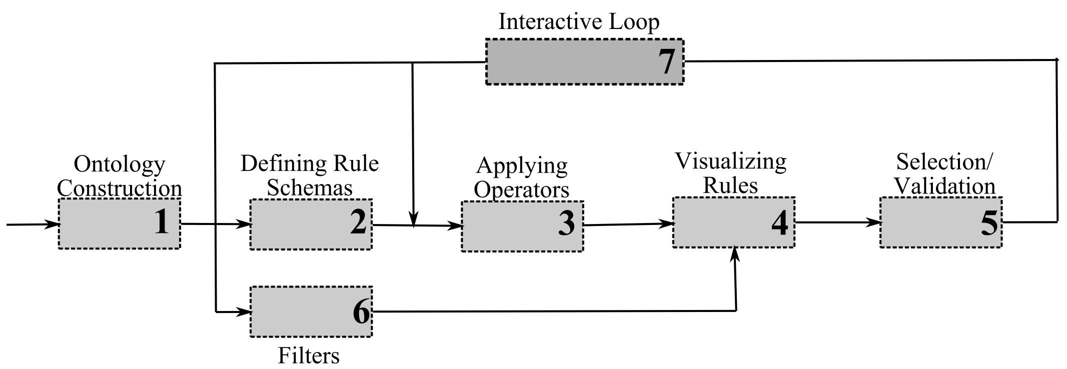


FIGURE 5.8: Interactive process description

1. *Ontology Construction.* This first phase consists in developing the first part of the user knowledge base. Starting from the database, and eventually from existing ontologies, the user develops an ontology of database items. It is important

to encourage the user to propose a wide range of knowledge, and, if possible, to complete with different information comparing to those found in the database;

2. *Rule Schemas Definition.* The second phase consists in creating the second part of the user knowledge base. The user uses ontology concepts in order to express his/her local goals and expectations concerning the discovered association rules;

3. *Operators Definition.* The use of operators increases the level of the framework interactivity. Letting the user to choose the operators to be applied over the Rule Schemas represents an important point in the liberty of the users in this framework, because choosing the operators is choosing the actions to be performed. Once the operators are selected, selecting/pruning filters are generated and applied over the set of rules;

4. *Visualization.* The visualization phase is very important, proposing to the user the result of his/her actions. An important set of interesting measures evaluates the filtered association rules so that, in case of an unanalyzable high volume, the user could at least know which is the more appropriate decision to take;

5. *Selection/Validation.* In this final phase, users evaluate subjectively the set of filtered rules starting from these preliminary results. Two possibilities are proposed: first, the user can validate the result and stop the research or, second, he/she can revise his/her information and restart the research;

6. *Additional Filters.* This phase proposes important methods for the reduction of the rule number: two *filters* already existing in the literature and detailed in Section 5.2.4.3. These two filters can be applied over the set of rules whenever the user needs them;

7. *Interactive Loop.* It allows the user to revise the set of knowledge that he/she provided in the precedent phases. Thus, he/she can return to phase 2 in order to modify the rule schemas, or he/she can return to phase 3 in order to change the operators. Moreover, in the interactive loop the user could decide to apply one of the two predefined filters discussed in phase 6.

5.2.4.2 Interactivity by using Operators over Rule Schemas

Before presenting the operators that we propose in our framework, let us remind the types of rules that Liu *et al.* [136] filter using general impressions. The basic idea of Liu *et al.* to compare association rules with general impressions was to verify syntactically if the rules contain the same items as the general impression. Thus, they proposed four types of rules:

- *Conforming rules* - association rules that are conforming with the specified beliefs;

- *Unexpected antecedent rules* - association rules having the antecedent non conforming with the belief one, but the consequent conforming with the belief one;

- *Unexpected consequent rules* - association rules having the conclusion non conforming with the belief one, but the antecedent conforming with the belief one;
- *Both-side unexpected rules* - association rules having the antecedent and the consequent non conforming with the belief ones.

The operators that we propose in our framework to be applied over rule schemas are slightly inspired from the types of filtered rules proposed by Liu *et al.*, with several differences that we will present in this section.

First, we would like to begin our presentation by explaining the association of 3 important notions: *rule schemas*, *operators* and *filters*. Let us remind you that rule schemas express user expectations in a rule-like formalism. Operators come as an action to be applied over the rules schemas (i.e. we can apply the operator *Conforming* over the rule schema $R_1 : A \rightarrow B$ in order to apply the *conforming action* over R_1, thus finally we select those association rules conforming to R_1). The notion of *filter* comes from the verb to filter and it is defined in our framework by the process of *applying* an operator over a rule schema. In other words, we create a filter which will realize over the association rule set the action of filtering based on conditions given by the operator and the rule schema.

Below, we will define the types of operators that we use in our framework. We propose two important sets of operators: *Pruning* and *Selecting – Conforming*, *Unexpectedness* and *Exception*.

On the one hand, for the first two operators of Selecting we propose to reuse the ideas behind the types of filtered rules proposed by Liu *et al.*: conforming and unexpectedness. On the other hand, we bring two new operators in the post-processing task: *Pruning* and *Exception*.

The filtering technique of the association rules is based on the idea of comparing association rules with the rule schemas. Therefore, we use as comparison technique a modified version of the syntactical method which is defined as follows.

Definition 5.2.13

Let us consider an ontology concept C associated in the database to

$$f(C) = \{y_1,\ ...,\ y_n\},$$

where $y_1,\ ...,\ y_n \in I$ and an itemset

$$X = \{x_1,\ ...,\ x_k\}.$$

We say that the itemset X is conforming to the concept C if $conf(X, C) = TRUE$, where:

$$conf(X, C) = \begin{cases} TRUE & \text{if } \exists y_i, y_i \in X \\ FALSE & \text{otherwise} \end{cases}$$

In other words, an itemset is conforming to an ontology concept if the latter is associated to at least one item of the itemset. ■

In the following we describe the four operators that we integrate in AR*IPSO*.

Pruning. In databases, there exist relations between items that we consider obvious or that we already know. Thus, it is not useful to find these relations among the discovered associations. Applied over a rule schema, the pruning operator, $P(RS)$, removes all association rules matching the rule schema. Thus, it permits to remove families of rules. To extract all the rules matching a rule schema the conforming operator is used.

Conforming. Applied over a rule schema, the conforming operator, $C(RS)$, confirms an implication or finds the implication between several concepts. For an association rule to be selected by the Conforming operator over a rule schema, the following set of conditions should be completed in function of the different types of the rule schema:

- if the rule schema is not an implication, an association rule is conformed to it if the itemset created by the union of the antecedent and the conclusion itemsets of the association rule is conforming to each concept composing the rule schema.

  Let us consider the following association rule

  $$A \to B,$$

  where A and B are itemsets, and a rule schema

  $$RS(< M >)$$

  where

  $$M = \{C_1, \ldots, C_k\}.$$

  We say that the association rule is selected by the Conforming operator, in other words, the association rule is conforming to the rule schema if

  $$\forall C_i \in M,\ conf(A \cup B, C_i) = TRUE.$$

- if the rule schema is defined as an implication, an association rule is conformed to if the antecedent and the consequent itemsets of the association rule are conforming to the each antecedent concept and, respectively, to each consequent concept of the rule schema. A graphical representation of this definition is given in Figure 5.9.

  In order to formalize this definition, let us consider the following association rule

  $$A \to B$$

  and the rule schema

  $$RS(< M_A \to M_B >)$$

  where

  $$M_A = \{C_1, \ldots, C_k\} \text{ and } M_B = \{C'_1, \ldots, C'_{k'}\}.$$

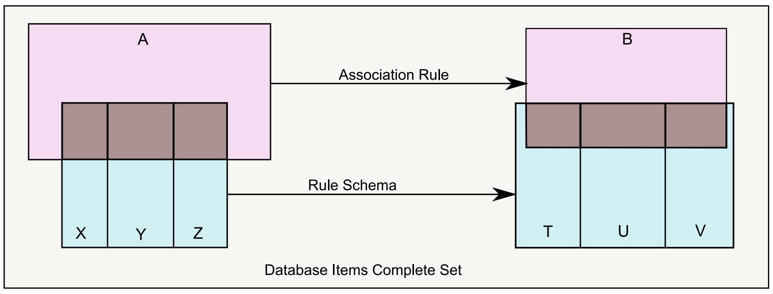


FIGURE 5.9: Selecting $A \to B$ association rule using Conforming operator over $X, Y, Z \to T, U, V$ rule schema

We say that the association rule is selected by the Conforming operator, in other words, the association rule is conforming to the rule schema if

$$\forall C_i \in M_A, conf(A, C_i) = TRUE$$
$$\text{and}$$
$$\forall C'_{i'} \in M_B, conf(B, C'_{i'}) = TRUE.$$

Example 5.2.14 Let us consider the implicative rule schema

$$RS_1 : RS(< Fruits \to EcologicalProducts >)$$

where
$f(Fruits) = \{$ *grape_espagne, red_apple, nashi, green_apple* $\}$,
$f(EcologicalProducts) = \{$ *grape_espagne, milk_100, steack_beef_100* $\}$ and
$I = \{$ *grape_espagne, red_apple, nashi, green_apple, milk_100, steack_beef_100, chicken_wings, chicken_legs* $\}$.

Also, let us consider that the following set of association rules is extracted by traditional techniques starting from the database:

	$C(RS_1)$
$R_1 : grape\_espagne, steack\_beef\_100 \to milk\_100, nashi$	$\oplus$
$R_2 : red\_apple \to steack\_beef\_100$	$\oplus$
$R_3 : green\_apple,\ nashi,\ milk\_100 \to chicken\_wings$	$\oplus$
$R_4 : grape\_espagne,\ nashi \to red\_apple$	$\ominus$
$R_5 : steack\_beef\_100 \to grape\_espagne$	$\ominus$
$R_6 : milk\_100,\ steack\_beef\_100 \to grape\_espagne$	$\ominus$

Let us consider the association rule R_1. Next, we will verify if R_1 is selected by the Conforming operator applied over the rule schema:

- first, we verify if the antecedent of the rule is conforming to the $Fruit$ concept of the rule schema antecedent. The antecedent of the rule contains the item *grape_espagne*, which is a fruit product also ($grape\_espagne \in f(Fruit)$), thus, the antecedent of the rule is conforming to the antecedent of the rule schema;

- second, we verify if the consequent of the rule is conforming to the *Ecological-Products* concept of the rule schema consequent. The consequent of the rule contains the item $milk\_100$, which is an ecological product also ($milk\_100 \in f(EcologicalProducts)$), thus, the consequent of the rule is conforming to the consequent of the rule schema

In conclusion, the association rule R_1 is selected by the Conforming operator applied over the rule schema. Two other association rules are selected by the operator: R_2 and R_3. ■

Unexpectedness. With a higher interest for the user, the unexpectedness operator, $U(RS)$, proposes to filter a set of rules with a surprise effect for the user. This type of rules interests the user more than the conforming one since, generally, a decision maker searches to discover new knowledge with regard to his/her prior knowledge.

Moreover, several types of unexpected operators are proposed:

- antecedent unexpectedness operator, $U_p(RS)$ – a rule is selected by this operator if it is not conformed to the rule schema by its antecedent;
- consequent unexpectedness operator, $U_c(RS)$ – a rule is selected by this operator if it is not conformed to the rule schema by its consequent;
- and both sides unexpectedness operator , $U_b(RS)$ – a rule is selected by this operator if it is not conformed to the rule schema by both its antecedent and consequent.

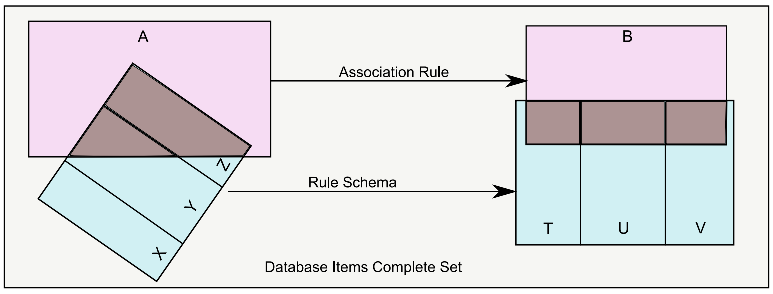


FIGURE 5.10: Selecting $A \rightarrow B$ association rule using antecedent Unexpectedness operator over $X, Y, Z \rightarrow T, U, V$ rule schema

Next, due to space limit and to repetitiveness of the definitions, in the following, we will detail only the antecedent unexpectedness operator applied over implicative rule schemas. Given a rule schema, an association rule is unexpected regarding the antecedent if the antecedent itemset of the association rule is not conforming to each antecedent concept of the rule schema, and if the consequent itemset of the association rule is conforming to each concept in the consequent of the rule schema. A graphical representation of this definition is given in Figure 5.10.

In order to formalize this definition, let us consider the following association rule

$$A \rightarrow B$$

and a rule schema

$$RS(< M_A \to M_B >)$$

where

$$M_A = \{C_1, \ldots, C_k\} \text{ and } M_B = \{C'_1, \ldots, C'_{k'}\}.$$

We say that the association rule is selected by the antecedent unexpectedness operator, in other words, that the association rule is conforming to the rule schema if

$$\forall C_i \in M_A, conf(A, C_i) = FALSE$$
$$\text{and}$$
$$\forall C'_{i'} \in M_B, conf(B, C'_{i'}) = TRUE.$$

Example 5.2.15 For the antecedent unexpectedness operator example, we will use the above example given in the Example 5.2.14 on page 115. Let us consider the rule R_6 and the given rule schema RS_1. We will verify if R_6 is selected by the antecedent unexpectedness operator applied over the rule schema RS_1:

- first, we verify if the antecedent of the rule {*milk_100, steak_beef_100*} is *NOT* conforming to the *Fruit* concept of the rule schema antecedent. None of the antecedent items of the association rule is a fruit product

  $$f(Fruit) \cap \{milk\_100, steak\_beef\_100\} = \emptyset;$$

  thus, the antecedent of the rule is not conforming to the antecedent of the rule schema;

- second, we verify if the consequent of the rule is conforming to the *Ecological-Products* concept of the rule schema consequent. The consequent of the rule contains the item *grape_espagne*, which is an ecological product also (*grape_espagne* ∈ *f(EcologicalProducts)*); thus, the consequent of the rule is conforming to the consequent of the rule schema.

In conclusion, the association rule R_6 is selected by the antecedent unexpectedness operator applied over the rule schema RS_1. Another association rule (R_5) is selected by the operator. ■

Exception. Finally, the exception operator is defined only over implicative rule schemas and extracts exception rules. Exception rules are rare phenomena which contradict more general rules. Thus, they come from a lower population, but they are very strong – having a hight confidence.

Presented by Suzuki 1997 [197], the research process of exception rules uses two important notions:

- the *common rule* which represents the initial rule, $X \to Y$. The goal is to extract the exceptions of this rule;

- the *exception rule* which contradicts the common rule. It can be expressed as $X \wedge z \rightarrow \neg Y$, where z is an item and $\neg Y$ represents the negation of Y – for binary databases this is expresses by the absence of the itemset Y, but in multi modal databases this can be viewed as the Y itemset with different values for its items.

Example 5.2.16 For the exception operator example, let us consider the rule R_3 in the Example 5.2.14 on page 115 and the given rule schema $RS_1 : RS(< Fruits \rightarrow EcologicalProducts >)$. To verify if R_3 is selected by the exception operator applied over RS_1, it should be conforming to the following rule schema:

$$RS_E : RS(< Fruits, z \rightarrow \neg EcologicalProducts >)$$

where z is an item. Thus, we follow the following steps:

- first, we verify if the antecedent of the rule is conforming to the $Fruit$ concept of the rule schema antecedent. The antecedent of the rule contains the items *green_apple* and *nashi*, which are fruit products, thus, the antecedent of the rule is conforming to the $Fruit$ concept in the antecedent of the rule schema;
- second, we verify if, apart the $Fruits$ items in the antecedent, an additional item exists. The *milk_100* item verifies this condition;
- finally, we verify if the conclusion of the rule does not contain ecological product items in order to verify the $\neg EcologicalProducts$ concept of the RS_E consequent. The consequent of the rule contains the item *chicken_wings*, which is an ecological product, thus, the conclusion of the rule is conforming to the conclusion of the RS_E rules.

In conclusion, the association rule R_3 is selected by the exception operator applied over the rule schema. ■

5.2.4.3 Additional Filters

In order to reduce the number of rules, we integrate two additional filters in the framework: minimum improvement constraint filter [20] and item-relatedness filter [148]. In this section we propose to brievly present these two interestingness measures that were already detailed in Chapter 2 and Chapter 4.

5.2.4.3.1 Minimum Improvement Constraint filter (*MICF*), developed by Bayardo *et al.* 1999 [20], selects only those rules whose confidence is greater with *minimp* than the confidence of any of its simplifications.

Definition 5.2.17

Let us consider the $R_{spec} : X \rightarrow Y$ association rule and $R_{gen} : X' \rightarrow Y$ where $X' \subset X$; that is to say that the rule R_{spec} is a specification of the rules R_{gen}, or, inversely, the rule R_{gen} is a generalization of the rule R_{spec}.

The improvement of the rule R_{spec} is computed as the minimum difference between its confidence and the confidence of its more general rules. More formally, it can be defined as:

$$imp(R_{spec}) = min(conf(R_{spec}) - conf(R_{gen}) \mid R_{gen} : X' \to X, \forall X' \subset X). \blacksquare$$

Further, if the improvement of the rule R_{spec} is positive, then we can conclude that this rule is interesting because it brings more information than all its general rules separately. If the improvement is negative we can conclude that the rule may improve the confidence of several general rules, but not of *all* of them. In other words, the rule R_{spec} could be simplified without loosing any information.

The threshold of minimum improvement (*minimp*) is provided by the user (i.e. 5%) once with the support and confidence.

Example 5.2.18 To illustrate the minimum improvement constraint, we will consider the following set of association rules

$$\begin{aligned}
R_1 : milk, pork \to pear &\quad [S = 20\%,\ \ C = 80\%]\\
R_2 : milk, apple \to pear &\quad [S = 27\%,\ \ C = 76\%]\\
R_3 : milk \to pear &\quad [S = 25\%,\ \ C = 70\%]\\
R_4 : pork \to pear &\quad [S = 30\%,\ \ C = 72\%]\\
R_5 : apple \to pear &\quad [S = 40\%,\ \ C = 83\%]
\end{aligned}$$

where R_1 is the specialization of rules R_3 and R_4, and R_2 is the specialization of rules R_3 and R_5. Also, we consider that the threshold of minimum improvement is $minimp = 5\%$.

Thus, we can compute the improvement of R_1 and R_2 as follows:

$$\begin{aligned}
imp(R_1) &= min(conf(R_1) - conf(R_3), conf(R_1) - conf(R_4))\\
&= min(10, 8)\\
&= 8\ (\%)\\
imp(R_2) &= min(conf(R_2) - conf(R_3), conf(R_2) - conf(R_5))\\
&= min(6, -7)\\
&= -7\ (\%)
\end{aligned}$$

and we can conclude that the rule R_1 – with an improvement of 8% which overpasses the minimum improvement threshold of 5% – is an interesting rule because it brings additional information comparing to its generalizations (general rules). On the contrary, the improvement of rule R_2 is negative, then less than the threshold limit of 5%, so, the rule is not interesting. ■

5.2.4.3.2 Item-relatedness filter (*IRF*) was proposed by Shekar and Natarajan 2005 [148]. Starting from the idea that the discovered rules are generally obvious, they introduced the idea of relatedness between items measuring their semantic distance using item taxonomies. This measure computes the relatedness of item couples.

Definition 5.2.19

Let us consider two items – X and Y –, and a fuzzy taxonomy organizing the items. The item-relatedness metric is defined in function of all possible paths from X to Y in the taxonomy by means of three important measures. The first measure, HM (Highest-Level Node Membership), computes the implication of each item in their common parent node; the second one, HR (Highest-Level Relatedness), measures the level difference between the parent of X and Y, and the taxonomy root. And, finally, the last measure, NSR (Node Separation Relatedness), is defined as the length of the simple path connecting the two items.

As a consequence, the item-relatedness measure of X and Y can be defined formally as:

$$\mathrm{IR}(X,Y) = \Sigma_{path} \frac{(1 + HR_{X,Y}(path))HM_{X,Y}(path)}{NSR_{X,Y}(path)}$$

with the variable path being all the possible paths between X and Y. ■

To qualify the interestingness of an association rule, we can compute the item-relatedness metric between each pair of items. Generally, we can distinguished three types of similarity: between the antecedent items, between the consequent items or between the antecedent and the consequent items. In our approach we use the last type of item-relatedness because users are interested to find association between itemsets with different functionalities, coming from different domains. This measure is computed as the minimum distance between the condition items and the consequent items as presented hereafter.

Definition 5.2.20

The distance between each pair of items from the condition and, respectively, from the consequent is computed as the minimum path that connects the two items in the ontology, defined as $d(a,b)$. Thus, in our case, the item-relatedness (IR) for a rule is defined as the minimum of all the distance computed between the items in the condition and in the consequent:

$RA_1 : A \to B$

$IR(A,B) = NSR_{A,B}(path_min) = MIN(d_{ij}(a_i,b_j)), \forall a_i \in A \text{ and } b_j \in B$ ■

Example 5.2.21 Considering the taxonomy in Figure 5.11 and the conditions imposed above, we will compute the item-relatedness of the following association rule:

$$grape,\ pear,\ butter \to milk$$

So, we will compute the metric for the following pairs: $(grape, milk)$, $(pear, milk)$ and $(butter, milk)$ and we can note that one single path is available for each pair:

$d(grape, milk) = 4$

$d(pear, milk) = 4$

$d(butter, milk) = 2$

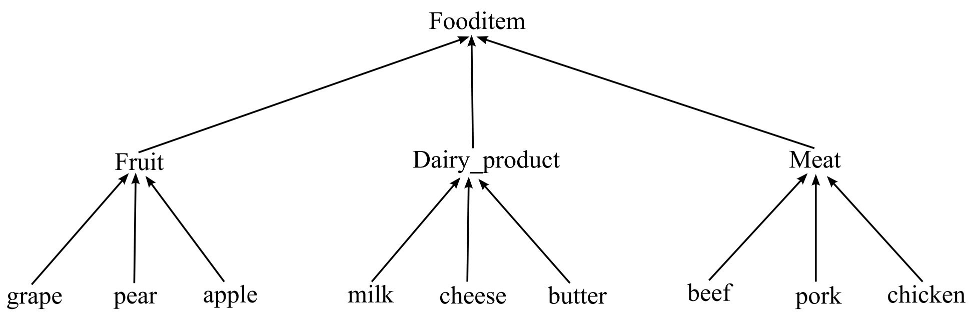


FIGURE 5.11: Supermarket taxonomy [136].

and the interestingness of the rule is computed as it follows:

$$IR = min\{d(grape, milk), d(pear, milk), d(butter, milk)\} = 2. \blacksquare$$

5.3 AR*LIUS* Approach

With this second approach[1] we propose to bring user knowledge and post-mining principles in the mining process. The main interest is that in a local method the user can focus on interesting rules without the necessity of extracting all rules existing in the database. In consequence, a local solution is faster than the post-mining one. The user may explore the rule space incrementally, a small amount at each step, starting from his/her own beliefs and knowledge; thus, the user is able to discover rules related to its knowledge. At each step the user chooses the most relevant rules for further exploration.

The differences with the AR*IPSO* approach are multiple. First, as we mentioned, AR*LIUS* approach (Association Rule Local Interactive mining Using rule Schemas) is based on a local search algorithm, while our first proposition is a post-processing technique. Second, the Rule Schema formalism developed in AR*LIUS* is different from the first one. It does not use ontology concepts, but it is more flexible and more interesting to use in an interactive approach. Next, the mining algorithm is not common to these two approaches, and last, one operator is comparable to conforming and exception in AR*IPSO*, and we propose two new ones. Nevertheless, we present them all because they work differently.

AR*LIUS* approach is composed of two main parts as shown in Figure 5.12. The first one, dedicated to user knowledge integrates new formalisms to represent user expectations and user actions.

On the one hand, we consider that the user has several knowledge concerning the discovered rules – he/she might have a set of elements which can be considered as starting points for the local mining algorithm. On the other hand, we integrate Operators over user beliefs in order to permit several types of actions that will be detailed further.

[1]that we developed during the collaboration with Andrei Olaru

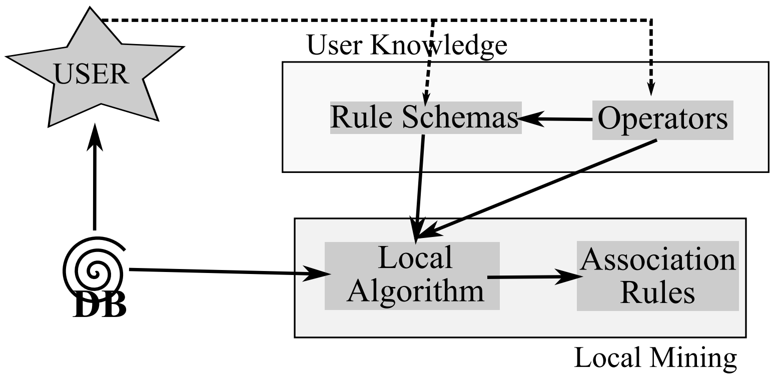


FIGURE 5.12: AR*LIUS* framework description.

The second part of the local approach consists in the mining algorithm itself. Its main goal is to propose to the user a set of association rule highly limited in volume. For this purpose, we suggest to start with the user beliefs and, by applying several operators, to generate directly the candidate rules.

An important characteristic of the AR*LIUS* approach is that it is described by a important interactivity with the user. At each generation of a set of association rules, the user is invited to select the interesting ones and also to propose an Operator (one action) to be applied over.

5.3.1 Rule Schema formalism

This approach is based on a novel, flexible and unitary specification language that we propose in order to represent user interests – the *Rule Schema*. This formalism is based on the specification language proposed by Liu *et al.* [136], but it improves it by completely covering the three levels of specification presented in the General Impressions formalism.

Let us consider that an user wants to find all rules that have the item *milk* in the antecedent and that contain also the following items: *apple*, *pear* and *pork*. For the last three items, the user is not sure of their position in the rule: if they are in the antecedent or in the consequent. We can attempt to create a belief with these elements. First, we will use the Reasonably Precise Concepts because the user has some ideas concerning the position of certain elements in the antecedent and/or consequent. Second, we try to express the existence of some elements in either one of the parts, by using the $*$ operator defining elements that have zero or more occurrences. Thus, the user beliefs can be expressed as follows:

$$rpc(< milk, apple*, pear*, pork* \rightarrow apple*, pear*, pork* >).$$

Nevertheless, this representation could not satisfy entirely the user; first, because it is more difficult to express and not so logic knowing that we cannot have the same item in the antecedent and consequent, and, second, because permitting to have the same item several times in a certain part it is not possible. In a second attempt we can propose to use the ? operator. This representation is also inappropriate for the expectations that the user wanted to express because we can have rules matching this *rpc* that do not contain the *pear* item (for example). Thus, this type of beliefs cannot be clearly expressed using the Liu's General Impressions formalism. As a conclusion, this is our motivation to propose a new language for user expectations, inspired from General Impression.

Definition 5.3.1
A Rule Schema is represented as follows:

$$rs(Antecedent \to Consequent\ \ [General]) \qquad [s\%\ c\%]$$

where the Antecedent and the Consequent contain the items that the user believes to be present in the antecedent and, respectively, in the consequent of the association rules. The General part contains the items that the user is not sure in which part to place.

The three parts – Antecedent, Consequent and General – are all Expressions of the form Expr = $\{Expr\} \mid [Expr] \mid Expr? \mid Item$ *– a disjunction or a conjunction of Expressions or an optional Expression (the items contained might not be present in the rule). The Rule Schema also contains optional constraints of support and confidence.* ■

Example 5.3.2 For instance, let us consider the case presented previously – the user knows that the item *milk* is in the antecedent and that the items *apple*, *pear* and *pork* are implicated also in this relation, but he/she is not sure how, or where to place these items. Then the user may define the following Rule Schema:

$$rs([milk] \to [\,]\ \ [apple, pear, pork]).$$

In this example we can remark that a part constituting the rule schema can be empty – as the Consequent in our case. ■

Example 5.3.3 Let us give a second example. The user may know that buying *pork* implies buying *apple*, and might believe that there is also a relation with *milk* and *cheese* items, but without knowing on what side of the implication they are and not even if *cheese* is really part of any rule. Moreover, the user wants to find only those rules with a support greater than 10% and a confidence greater than 40%. Thus, the user may define the following Rule Schema:

$$rs([pork] \to [apple]\ \ [milk, cheese?]) \qquad [10\%\ 40\%].$$

In this example, we show the use of the ? operator. It permits to put an option on the *cheese* item. ■

The improvement that we bring to General Impressions is a more important flexibility and a higher belief expression because Rule Schema may use all three parts simultaneously, and in some particular cases the Liu languages are simulated. For instance, if the *Antecedent* and *Consequent* are used omitting the General part, the

Rule Schema is more like a Reasonably Precise Concept or Precise Knowledge. If the *General* part is used without the Antecedent and the Consequent, the Rule Schema simulates a General Impression. But, outward these two cases, our formalism brings new expressions to the knowledge language.

5.3.2 User Interactivity

5.3.2.1 Interactive Process

The interactive process of the Local AR*IPSO* framework (presented in Figure 5.13) aims at guiding the user through the rule extraction phase. Taking into account his/her feedbacks, the user is able to revise his/her expectations and actions. Several steps are suggested as follows:

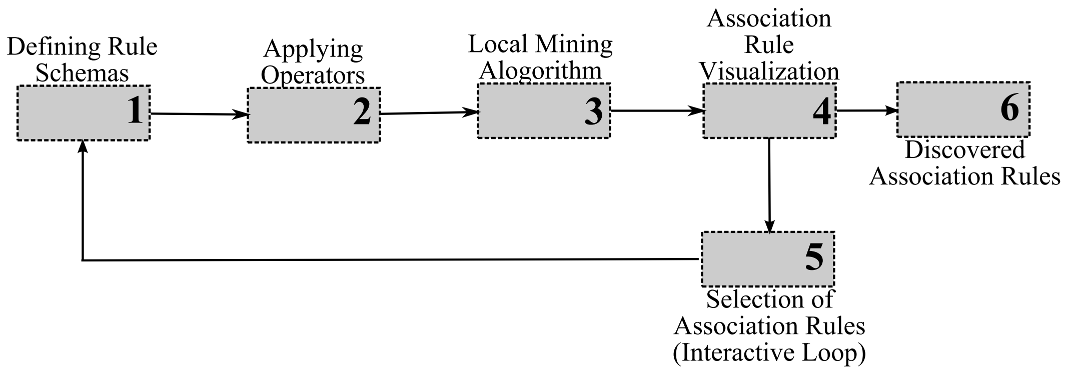

Defining Rule Schemas
1
Applying Operators
2
Local Mining Alogorithm
3
Association Rule Visualization
4
6
Discovered Association Rules
5
Selection of Association Rules (Interactive Loop)

FIGURE 5.13: Interactive process description for the AR*LIUS* Framework

1. *Rule Schemas Definition.*

2. *Operators Definition.*

   These first two phases have the same role as in the AR*IPSO* approach. They help the user to define Rule Schemas and Operators, and to increase the interactivity of the framework;

3. *Local Mining Algorithm.* In this phase, candidate rules are searched locally. The research is based on the rule schemas and operators given in the two last phases;

4. *Association Rules Visualization.* The visualization phase is very important, proposing to the user the result of his/her research. After visualizing the association rules generated by the research, the user has two possibilities: to validate the extracted rules and to stop the mining process (phase 6), or to select several association rules to be used further in the mining process (phase 5);

5. *Selection of Association Rules.* If the user didn't find what he/she was searching for, he/she can continue the mining process by selecting a set of rules in order

to be injected in the Local Mining Process as Rule Schemas; in this case the user goes back to first phase. This interactive loop gives the user a great liberty for choosing the Rule Schemas;

6. *Discovered Association Rules.* This is the last phase of the framework. Once the user considers the rules that he/she searched was found he/she can stop the mining process and validate the discovered rules.

5.3.2.2 Operators over Rule Schemas

Interesting rules are in a certain relation – confirmation or contradiction – with the current beliefs of the user. As a consequence, we extend previous works [136] by proposing 4 Operations that allow the user to explore the rule space starting from his/her beliefs and knowledge.

5.3.2.2.1 Confirmation. The Confirmation Operator is the simplest operation that we propose. The user can insert concept expressions in all three lists, according to what he/she knows or suspects to know. This operation can be used to confirm an implication, to find what is the direction of the implication among several items, or to find how different concepts influence a rule by allowing them to be placed on any side of the implication. It filters all rules that contain the items of the *Antecedent* and of the *Consequent* in the antecedent, and respectively, in the consequent, and the items of the *General* part in any of the two sides of the implication. The items in the *General* part may be split in any possible ways between the antecedent and the consequent.

Definition 5.3.4
Formally, the Confirmation Operator generates rules of the form:

$$Antecedent \cup Subset \to Consequent \cup (General - Subset)$$

$\forall Subset \subseteq General.$ ■

During Confirmation, the Expressions present in the Rule Schema are transformed into lists of items. Disjunctive and optional expressions result into different groups of rules, each corresponding to one element in the disjunction. For example, if the antecedent is an Expression of the form $[\{milk,\ pork\}\ pear?]$, the resulting rules could have the following antecedents:

$[milk, pear]$
$[pork, pear]$
$[milk]$
$[pork]$.

Example 5.3.5 For instance, let us consider the following Rule Schema:

$$rs([pear] \to [grape, milk?]\ \ [pork, apple]).$$

The following eight rules confirm the Rule Schema:

$R_1 : pear, pork, apple \rightarrow grape, milk$

$R_2 : pear, pork, apple \rightarrow grape$

$R_3 : pear, pork \rightarrow grape, milk, apple$

$R_4 : pear, pork \rightarrow grape, apple$

$R_5 : pear, apple \rightarrow grape, milk, pork$

$R_6 : pear, apple \rightarrow grape, pork$

$R_7 : pear \rightarrow grape, milk, pork, apple$

$R_8 : pear \rightarrow grape, pork, apple$

We can remark that we have used all the possibilities to dispatch the two items *pork* and *apple* from the General part, and that we duplicate the rules in order to generate rules with or without the *milk* item. ■

5.3.2.2.2 *k*-Specialization. This second Operator is based on minimum improvement constraint metric proposed by Bayardo *et al.* [19] and it allows the user to find the rules that have a more specific antecedent, the same consequent, and which improve the confidence of the initial rule. That is to say that, this Operator allows to specialize the Rule Schema by specializing the antecedent; thus, a set of items are added to the antecedent. The variable k defines the number of items that are added in the antecedent during the specialization. For example, a specialization of the rule $X \rightarrow Y$ $[s_1\ c_1]$ is the rule $X, Z \rightarrow Y$ $[s_2\ c_2]$, if $c_2 > c_1$; if Z contains only one item, we can say that this is an *1*-Specialization.

The k-Specialization is an operation that can be applied only over a rule containing clearly an antecedent and a consequent. Thus, the k-Specialization is not performed directly on the initial Rule Schemas, but it is preceded by a Confirmation Operator, in order to transform expressions into item lists, to split the General part and to calculate confidence and support. The k-Specialization is performed on rules in the output of the Confirmation operation.

Definition 5.3.6

Let us consider that the following rule is resulted after applying the Confirmation Operator over a typical Rule Schema:

$$Antecedent \rightarrow Consequent\ [s\%,\ c\%].$$

The rules generated by the application of the k-Specialization Operator over this rule are of the form:

$$Antecedent \cup Set \rightarrow Consequent\ [s_s\%,\ c_s\%]$$

$\forall Set \subseteq (I - (Antecedent \cup Consequent))$*, where* $|Set| = k$ *and with* I *the full set of items. Further, the confidence of the specialized rule should improve the confidence of the initial rule, so, the following condition should be satisfied:* $c_s > c$. ■

Example 5.3.7 Let us consider the following Rule Schema

$$rs([milk] \rightarrow [pork][chicken])$$

and the set of items $I = \{tomatoes, pork, pear, grape, chicken\}$. In a first phase, the Confirmation Operator is applied, and two rules are generated with the corresponding support and confidence. The output of the *1*-Specialization operation is:

$milk, chicken \rightarrow pork$ [10% 75%]
  $milk, chicken, pear \rightarrow pork$ [8% 79%]
$milk \rightarrow pork, chicken$ [10% 70%]
  $milk, grape \rightarrow pork, chicken$ [7% 73%]
  $milk, pear \rightarrow pork, chicken$ [8% 80%]

We should note that other specializations can be generated starting from the Rule Schema, but we consider here that they do not improve the confidences of 75% and 70% of the Rule Schemas.

Furthermore, if we apply a *2*-Specialization Operator, not only one, but two items are added in the antecedent, and the confidence improvement constraint remains. Thus, we have the following result:

$milk, chicken \rightarrow pork$ [10% 75%]

$milk \rightarrow pork, chicken$ [10% 70%]
  $milk, grape, pear \rightarrow pork, chicken$ [6% 75%]. ■

5.3.2.2.3 *k*-Generalization. This Operator is the opposite of k-Specialization. It finds the rules that have a more general antecedent implying the same consequent. Support is expected to be higher and confidence slightly lower. As for the k-Specialization, the variable k represents the number of items that are removed. Moreover, as the k-Specialization Operator, the k-Generalization Operator cannot be applied directly over the Rule Schemas, and the same process presented earlier should be done.

Definition 5.3.8
Searched rules by k-Generalization Operator are of the form:

$$Antecedent - Set \rightarrow Consequent$$

$\forall Set \subseteq Antecedent$, *where* $|Set| = k$. ■

Example 5.3.9 Let us consider the following Rule Schema:

$$rs([milk,\ pear,\ apple] \rightarrow [pork]\ [chicken]).$$

In a first phase, the Confirmation Operator is applied, and two rules are generated with the corresponding support and confidence. The *1*-Generalization Operator is applied over these two rules, and the output is:

$milk, pear, apple, chicken \to pork$ [10% 75%]
    $milk, pear, chicken \to pork$ [15% 79%]
    ...
$milk, pear, apple \to pork, chicken$ [10% 70%]
    $milk, pear \to pork, chicken$ [20% 80%]
    ...

Furthermore, if we apply a *2*-Generalization Operator, two items are removed from the antecedent. Thus, we can have the following result:

$milk, pear, apple, chicken \to pork$ [10% 75%]
    $milk, pear \to pork$ [30% 70%]
    ...
$milk, pear, apple \to pork, chicken$ [10% 70%]
    $apple \to pork, chicken$ [60% 80%]
    ... ■

5.3.2.2.4 *k*-Exception. This is an important operation as it finds rules with a unexpected consequent, in the context of a more specialized antecedent. That is, for rules of the form $X \to y$ [$s1$ $c1$] exceptions are of the form $X,\ Z \to \neg y$ [$s2$ $c2$]. In [197] an exception is considered valuable knowledge if, knowing that the confidence of $Z \to \neg y$ is $c3$, then $c2 \geq c1$, and $c3$ must be fairly low, as it must not be Z alone, but the association with X that leads to $\neg y$. k-Exception operation is the k-Specialization of the rules with negated conclusion. For binary attributes, the negation is defined as the negated value (usually it means that the item does not exist in the transaction). For multi-modal attributes, the negation is represented by any other value than in the Schema.

Definition 5.3.10
Let us consider the following Rule Schema:

$$rs(Antecedent \to consequent\ []),$$

where consequent is an item. The rules resulting from the application of the k-Exception Operator over this rule are of the form:

$$Antecedent \cup Set \to \neg consequent$$

$\forall Set \subseteq (I - (Antecedent \cup \{consequent\}))$, *where* $|Set| = k$ *and with* I *the full set of items and* $item \in I$. ■

5.3.3 Local Mining Algorithm

There are many types of algorithms for the extraction of patterns and association rules from transaction databases. One of the most popular is the *Apriori* algorithm. By an incremental approach, *Apriori* finds all frequent itemsets – all itemsets that have a support above a certain threshold. On the basis of the frequent itemsets, the algorithm builds all rules that have a confidence value above a given threshold. Another idea to extract associations was to integrate user knowledge and expectations into the rule mining process. However, modifying the pruning strategy in order to focus on interesting rules, in conjunction with the operations described in the previous section, is not very efficient and generates limited results. Moreover, certain issues arose, like how to remove already known rules in a mining algorithm like *Apriori*, in the context of keeping exception rules and, if necessary, confirmations. As an alternative, post-mining filtering step may be applied, but a better approach could be to integrate the filtering techniques into the mining algorithm itself reversing the direction of previously developed mining algorithms.

5.3.3.1 Presentation

In this approach, the search for interesting rules is done locally, in the neighborhood of rules and associations that the user already knows, or that the user believes to be true, specified by means of the Rule Schemas. Instead of generating all rules (by means of frequent itemsets), and filtering those that are conform to user knowledge, the new approach consists of first generating locally all candidate rules, based on the rule schemas, and then checking their support and confidence against the transaction database. The candidate rules are all possible rules that are conforming to the specified schemas and operations. After generation, a pass through the database is performed in which the support and the confidence of candidate rules are computed. In order to be present in the output, rules must comply with the support and confidence requirements specified globally or for each rule schema.

The pseudocode of the Confirmation Operator algorithm is presented in Table 5.1. First, the lines 2 and 3 are especially dedicated to manipulate the Expressions, more precisely to create the lists of items by evaluating the operators in Expressions. Second, the lines 4 and 5 take each list of items of the Antecedent (*Ant*) and the Consequent (*Cons*), and each subset (*GS*) of each item list (*Gen*) of the General part (all generated in the previous steps) in order to create a new candidate rule: $(Ant \cup GS) \rightarrow Cons \cup (Gen - GS)$. In lines 7-8 the support and the confidence of the candidate rules are tested against the database, and in line 9, all rules that are not satisfying the support threshold are eliminated and the remaining rules are presented to the user.

One of the most complex Operator is the k-Specialization one; the Table 5.2 presents the pseudocode of the algorithm for *1*-Specialization Operator integrated in the AR*LIUS* framework. As already stated, the k-Specialization operation is always done in two important steps: first, a list of candidate rules is generated by applying the Confirmation Operator over the rule schema; second, starting from each rule the k-Specialization rules are generated. In our algorithm, the k-Specialization

Table 5.1: Confirmation Operator Algorithm.

Input: Rule Schema described by Antecedent, Consequent, General
Output: *rList* – the list of rule confirming the Rule Schema

```
1. create empty lists of rule candidates $rList$ and $rListS$

2. for each side $S_d$ in {$Antecedent$, $Consequent$, $General$ }
3.      create set $C[S_d]$ containing all item lists resulting from
             expending the $Expressions$ in $S_d$
4. for each item list $Ant \in C[Antecedent]$, $Cons \in C[Consequent]$,
             $Gen \in C[General]$
5.     for each subset $GS$ of $Gen$
6.          add to $rList$ a new candidate rule
             $CR(Ant \cup GS \rightarrow Cons \cup (Gen - GS))$
7. for each candidate rule $CR \in rList$
8.     check support $s$ and confidence $c$ for $CR$
9.       remove from $rList$ all rules with $s < minsup$
10. return $rList$
```

operation is performed by k *1*-Specialization consecutive operations. Thus, when a *2*-Specialization Operator is applied over a Rule Schema, the algorithm in Table 5.2 is applied 2 times consecutively. The algorithm is quite simple: first, starting from the rules generated by the Confirmation Operator, specific rules of level 1 are generated, in lines 1-3, by adding an item in the antecedent of the candidate rules. Second, for each candidate rule the support and the confidence metric are computed against the database. Last, those rules not over-passing the support and the confidence threshold, and more important, the basis rule confidence, are pruned.

The algorithms for k-Generalization and k-Exception Operators are quite similar and we will not develop them.

Example 5.3.11 Suppose the user wants to find all *1*-Specialization rules of the Rule Schema

$$rs(\ [milk] \rightarrow [\{pear, pork\}]\ [apple])\ \ [10\%, 60\%].$$

That is to say that, *milk* leads to either *pear* or *pork*, and they are associated with *apple*. Support and confidence must be over 10% and, respectively, 60%. The full item set is $I = \{milk, pear, pork, apple, grape, chicken\}$. The algorithm works as follows:

- first, the Confirmation Operator is applied over the given rule schema in order to expend the Expressions and to split the *General* part between the Antecedent

TABLE 5.2: *1*-Specialization Operator Algorithm.

Input: Rule Schema described by Antecedent, Consequent, General
Output: *rListS* – the list of specialization rules of the Rule Schema

```
$rList = Confirmation(Antecedent, Consequent, General)$

1. for each candidate rule $CR(X \rightarrow Y) \in rList$
2.     for each item $i \in I - (X \cup Y)$
3.         add to $rListS$ the $CR(X \cup \{i\} \rightarrow Y)$
4. for each $CR(X \cup \{i\} \rightarrow Y) \in rListS$
5.     check support $s$ and confidence $c_1$ for $CR$ and
               check confidence $c_2$ for $\{i\} \rightarrow B$
6. remove from $rListS$ all rules with $s < minsup$ or
                       $c_1 < minconf$ or $c_1 < c2$
7. return $rListS$
```

and the Consequent. Generated rules are then checked against the database and candidate rules with support lower than 10% are pruned:

[*milk*] → [*pear*, *apple*] [25%, 67%]
[*milk*, *apple*] → [*pear*] [25%, 44%]

[*milk*] → [*pork*, *apple*] [8%, 26%] − −pruned
[*milk*, *apple*] → [*pork*] [8%, 35%] − −pruned

- the second step corresponds to the *1*-Specialization operation itself. Starting from the first two rules provided by the first step, new candidate rules are generated by adding one item from I in the antecedent. Next, candidates rules are checked and pruned if support or confidence are lower than the threshold specified in the Schema, or if there is no improvement in confidence comparing to the more general rule.

  Let us consider the first rule validated by the Confirmation Operator

  $$[milk] \rightarrow [pear,\ apple] \quad [25\%,\ 67\%].$$

  {*pork*, *grape*, *chicken*} are the items in I that do not appear in the rule, and that can be used in order to make the specialization. Thus, one by one, the items are added in the antecedent of the rule, and new candidate rules are generated. These rules are validated against the support and confidence thresholds of $[10\%, 60\%]$ provided with the Rule Schema, and, moreover, they are validated against the confidence improvement constraint – if the candidate rule does not improve the confidence of 67% of the rule, the candidate rule is pruned.

$[milk,\ chicken] \to [pear,\ apple]\quad [20\%,\ 83\%]$

$[milk,\ pork] \to [pear,\ apple]\quad [7\%,\ 20\%]$ – –pruned for support constraint
$[milk,\ grape] \to [pear,\ apple]\quad [17\%,\ 62\%]$ – –pruned for confidence improvement constraint

In the same way, for the second rule $[milk,\ apple] \to [pear]\quad [25\%,\ 44\%]$ validated during the Confirmation phase we have the following result:

$[milk,\ apple,\ chicken] \to [pear]\quad [14\%,\ 66\%]$

$[milk,\ apple,\ pork] \to [pear]\quad [7\%,\ 35\%]$ – –pruned for support
$[milk,\ apple,\ grape] \to [pear]\quad [9\%,\ 48\%]$ – –pruned for support

- results are sorted according to the confidence:
  $[milk\ chicken] \to [pear\ apple]\quad [20\%\ 83\%]$
  $[milk\ apple\ chicken] \to [pear]\quad [14\%\ 66\%]$

■

5.3.3.2 Advantages and Complexity

There are a number of advantages that this approach has compared to the *Apriori* algorithm. They result in great part from the fact that the Rule Schemas are partially instantiated, so the search space is greatly reduced. Moreover, the more the user refines current Rule Schemas, the lower is the number of generated candidate rules.

Compared to *Apriori*, the number of passes through the databases is lower. Once all candidate rules are generated, only one pass through the database is necessary, to check the support of the candidates. For complexity reasons, in the case of multi-level operations, one pass per specificity/generality level is necessary.

One important issue in the presented approach is the number of generated candidate rules. This depends on the Operator and on the properties of particular Rule Schemas, as shown below.

For the operation of Confirmation on a Rule Schema $rs([X] \to [Y]\ [Z])$ (where X, Y, Z are itemsets), the number of generated rules is equal to the number of possibilities of splitting the Z set into two subsets, computed by the Stirling number of the second kind[2], it is comparable to $2^{|Z|}$. For each subset S in Z, S is added to the antecedent and $Z - S$ to the consequent. Usually, the number of items in Z will be fairly low.

[2]http://en.wikipedia.org/wiki/Stirling_number_of_the_second_kind

In Specialization, all the rules of the form $X \cup S \rightarrow Y$ are generated, where $S \not\subset X \cup Y$ and $|S|$ is the specificity level. The number of candidate rules is $C^{|S|}_{|I|-|X\cup Y|}$ where I contains all items in the database. It is important to note that the number of items in S is quite low, so this combinatorial problem is not time consuming.

For the operation of k-Generalization on a rule schema $rs([X] \rightarrow [Y]\ [])$, candidate rules with fewer items in the antecedent are generated. For all levels of generality, $2^{|X|}$ candidate rules are generated, but X is likely to not have more than 5 or 6 items.

For the operation of k-Exception on a schema $rs([X] \rightarrow [y]\ [])$, where X is an itemset, and y an item, the rules of the form $X \cup S \rightarrow \neg y$ are generated (S is a set of items not in X and not containing y, and $|S| = k$). The number of candidate rules is of the same order with the k-Specialization Operator, but, in the case of multi-modal attributes, it is multiplied with the number of modalities of the attribute y, minus 1.

5.4 Conclusion

In this Chapter we presented our two mining/filtering approaches which aim to reduce the number of discovered association rules and to improve their quality. The first method, called AR*IPSO*, performs in the post-processing step of the KDD process. The novelty of this approach resides in supervising the association rule mining process using two different conceptual structures to represent user domain knowledge – one or several *Ontologies* – and user expectations – several *Rule Schemas* generalizing *general impressions*. Further, the approach integrates *Operators* over Rule Schemas that aim to produce an well defined action over the set of association rules. Finally, AR*IPSO* framework provides an interactive and iterative post-mining process, which facilitates the user task.

In a first place, we detailed the structure of the ontology, then we defined the Rule Schema formalism, in order to present the connection between the two ones and to describe the filtering process using Operators.

The second method, called AR*LIUS*, is different from the former because it concentrates on the mining algorithm. It proposes a new algorithm for association rule mining integrating user knowledge by means of *Rule Schemas*. The formalism developed in this method is different from the one proposed in AR*IPSO* – it is more flexible and more expressive. The novelty of this approach, beyond the Rule Schema formalism, stands on a new set of Operators and on a new mining technique proposing a local rule mining. The main interest is that the number of discovered rules during a research is low, and that partial results can be injected in the mining algorithm.

The presentation of AR*LIUS* started with the definition of the Rule Schema formalism and of the Operators. Next, the mining algorithms developed for each Operator were detailed.

6
AR*IPSO* Tool Development

A successful tool is one that was used to do something undreamed of by its author.

Stephen C. Johnson

Contents

6.1 Introduction

In the last decade, using subjective measures in association rule mining techniques has been an important research field with a hight number of publications in the most well coted journals and conferences. While, a great part of these publications conceive different approaches, a real tool implementing the most interesting subjective measures was not yet developed. The main reason comes from the difficulty of putting together all the necessary blocks in such an approach. On the one hand, one of the most difficult problems is to choose the right models for user knowledge representation and the technical solutions that could connect the models with the software tool. On the other hand, the developed tool should be regularly validated by domain experts. As the knowledge acquisition research field pointed out, this task is quite difficult because experts are rarely available for this type of projects.

In this Chapter, we are interested in presenting the AR*IPSO* tool which implements the AR*IPSO* approach. This Chapter is motivated by the important efforts that were deployed in order to choose the right representation languages for user knowledge and the technical tools.

First, we discuss the architecture of the AR*IPSO* tool, and we detail the most important modules. Second, we outline the technical choices that we made. Next, we present the details of the development of each module, and, finally, we study the development process.

6.2 Tool Architecture

The AR*IPSO* Tool proposes to implement the AR*IPSO* approach introduced in the previous chapter. For this purpose, we elaborate a modular and evolving architecture that we designed as it shows Figure 6.1.

The tool is organized in 2 important modules: the *IHM* and the *Core*. In gray nuances we distinguished external modules for data processing: *User Knowledge* and *Classic KDD Process*.

The *Classic KDD Process* consists of two different parts: the first one is a classic iterative process of association rule extraction, while the second one transforms association rules in PMML format. The KDD process starts with the entire database which, in an initial phase, is pre-processed depending on expert needs and transformed in an ARFF file. Next, we use the association rule mining module of *Weka*[1] to extract the association rules, and *Arval* Tool[2] to compute additional metrics of the rules and to represent them in the *PMML*[3] format.

The *User Knowledge* module consists of stocking the user knowledge. We propose to represent ontologies using the OWL language – they will be saved in OWL files –, and the rule schemas in the XML format – XML files.

The *IHM* module is working on the display problems, and the *Core* one deals with the most important tool units: the engines for rule analyzing and ontology treatment,

[1]http://www.cs.waikato.ac.nz/ml/weka/

[2]http://www.polytech.univ-nantes.fr/arval/

[3]http://www.dmg.org/v4-0/GeneralStructure.html

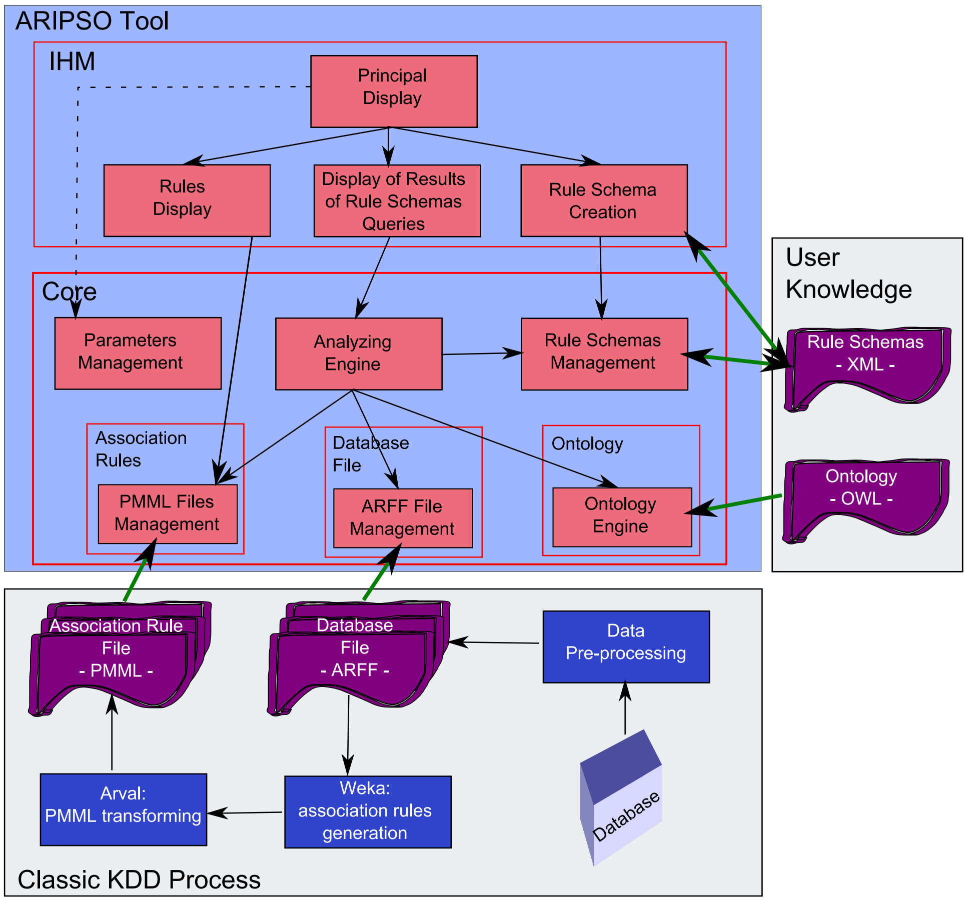


FIGURE 6.1: General architecture of the AR*IPSO* tool and its interactions with the environment.

and the management of all the elements – rule schemas, rules, ontologies, database.

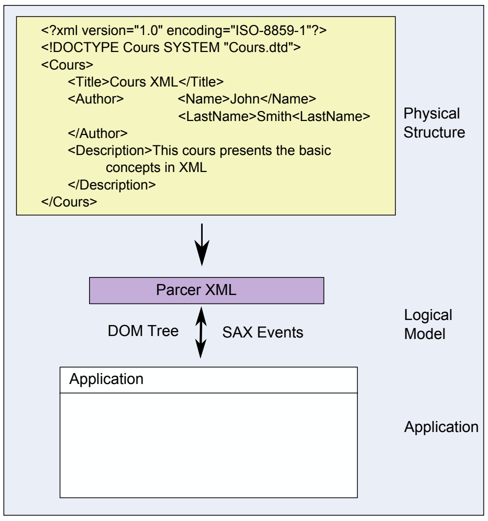
6.3 The Choice of Technical Solutions

The main constraint for our tool was to be developed based on Open Source software. Concerning the programming language, we have chose to use Java[4] language and the development framework, Eclipse[5]. Choosing Java as programming language was decided by the numerous *APIs* available for XML, PMML, or OWL treatment.

The access to XML files (which we have detailed in section 3.4) is done by using an *XML parser* and the technique of parsing XML files is presented in Figure 6.2. Two XML parsers are proposed in the literature: *DOM* and *SAX*.

The *DOM (Document Object Model)* API is built to access and to work on an entire XML stream. The latter is seen as a tree to which we can change, delete or

[4]http://java.sun.com/
[5]http://www.eclipse.org/



```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE Cours SYSTEM "Cours.dtd">
<Cours>
      <Title>Cours XML</Title>
      <Author>          <Name>John</Name>
                        <LastName>Smith<LastName>
      </Author>
      <Description>This cours presents the basic
              concepts in XML
      </Description>
</Cours>
```

FIGURE 6.2: Principle of DOM and SAX API utilization.

retrieve data. It provides all functionalities allowing complex modifications in the file. Its main drawback is the need of important memory resources.

The *SAX (Simple API for XML)* API is a library designed to respond to the content of an XML document. SAX performs a sequential reading of XML files. Events are generated for each element of the XML file (tag, attribute, data). Only current reading data are available. Further, it is not possible to search through the file, but, comparing to DOM API it uses few memory and it can handle large files.

In this case, our choice goes to SAX because, unlike DOM, it has no limit regarding the size of the treated files.

As already discussed, we proposed to use ontologies in OWL-DL for knowledge domain representation and the choice of this language was made naturally. The reason of making this choice is that OWL is a complex and flexible language for knowledge representation. Its provenance from DLs brings it all DL power – reasoners. An important influence in our choice had also the great number of inference engines APIs available for OWL language.

To develop and to view the ontology we proposed to use the Protégé editor. This tool overcomes the syntax OWL-XML and RDF-XML, providing a graphical representation of the ontology. It is easy to visualize relationships, intersection of classes or the restrictions of properties.

To exploit our ontology in the software, we had to choose an ontology framework. Following extensive testing, we chose Jena framework[6], developed by Hewlett Packard. It provides all the functionalities to manipulate the ontology, the hierarchy of classes and their properties. In addition, detailed documentation is available with the installation version. Finally, comparing to OWL API[7], Jena is designed to interface easily with the main inference engines (i.e. Pellet).

[6]http://jena.sourceforge.net/
[7]http://owlapi.sourceforge.net/

Our application is described by an architecture of thick (fat) client type, and it is designed to be autonome. In consequence, this criterion was predominant in the choice of the inference engine. The propositions implementing DIG interface are not a solution because of several know flaws – very limited support for datatypes and problems in fitting exactly DIG relations and OWL properties. Thus, we privileged direct reasoning – Pellet – to DIG-based reasoners (RacerPro, FACT++, . . .). The main advantage of direct reasoning is that it has a direct access to the ontology, while, in the case of DIG-based reasoners, the ontology is transformed in different structures in order to permit reasoner access. As already stated, this process could produce important limitations. Further, Pellet is natively included in Jena distribution.

6.4 Packaging Organization

The diagram in Figure 6.3 presents the organization of different modules in Java packages.

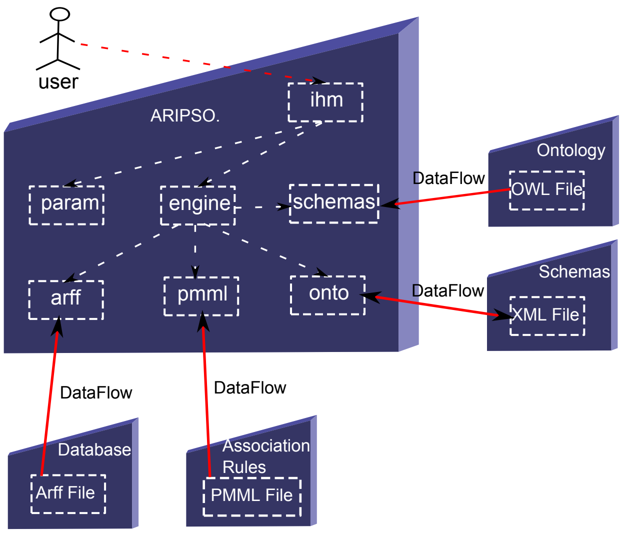


FIGURE 6.3: Diagram Packaging of the AR*IPSO* Tool.

The application is composed of 7 packages and 35 Java classes (detailed in Figure 6.4). The advantage of such a division comes from a weak connexion between different components of the application. The result is time saving when adding new features.

In order to improve clarity, the names of the classes are standardized. The names of all classes specific to the application start by *Aripso*, and the second part describes the role of the class. For instance, in the *Aripso.ihm* package, the class *AripsoDlgXxx* corresponds to dialog boxes, and *AripsoMenuXxx* regroups the menus.

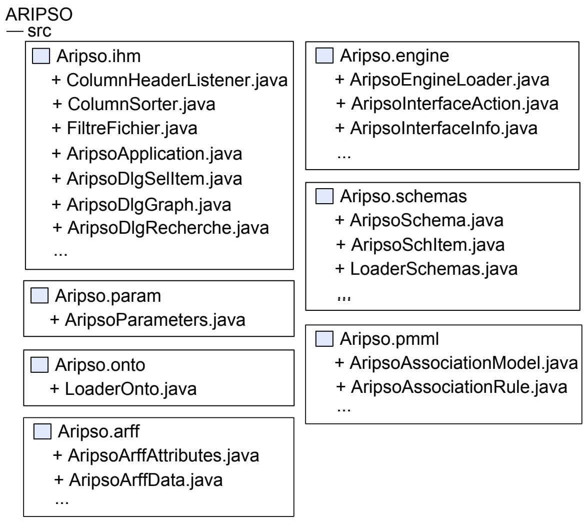


FIGURE 6.4: The application classes organized in packages.

6.5 Modules Development

6.5.1 Association Rule Management Module

Browsing association rules is an essential feature of the application. During various experiments, we determined that the number of rules to handle is from several hundreds of thousands to several millions. Therefore, we must have a reliable, robust and powerful tool which does not penalize the processing time. For this purpose, we chose to represent association rules in PMML language[8]. PMML is an XML-based markup language for association rule representation, and its main advantages is that it is easily exploitable due to XML parser tools (we use SAX API as stated in Section 6.3), and, further, it permits items, itemsets, and different metrics integration.

The functionality of saving association rules in PMML format completes the browsing one. The class diagram in Figure 6.5 allows to stock the informations and to browse them as follows:

- *LoaderPmml* – controls the analyzing phase of a PMML file;
- *AripsoAssociationModel* – contains the entire set of rules, itemsets and items. We used hash trees that ensure consistent response timed which do not depend on the number of processed rules. To improve the response time, the system also maintains a hash tree that acts as indexes by referencing the lists of rules for different number of itemsets;
- *AripsoItemSet* – contains information relative to an itemset;
- *AripsoAssociationRule* – contains information relative to an association rule.

[8]http://www.dmg.org

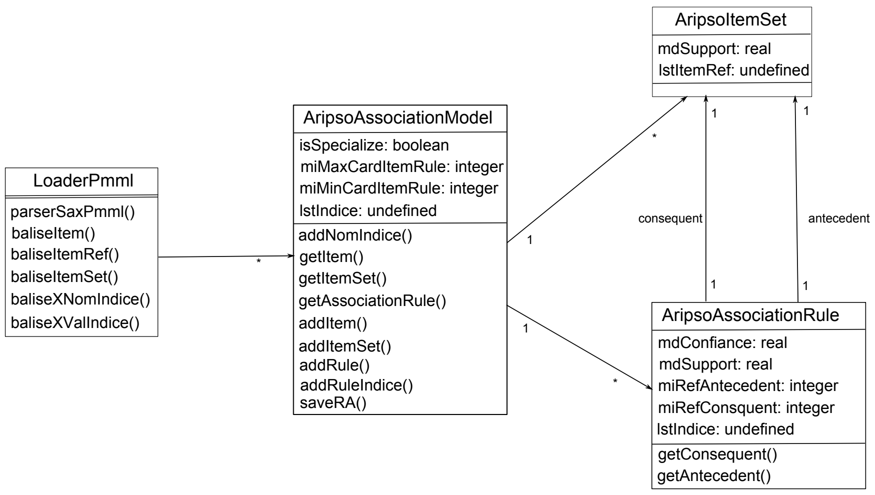


Figure 6.5: Class diagram for association rules management (PMML files).

This module functions as follows: the *LoaderPmml* starts the procedure of reading the Pmml file by the *parserSaxPmml()* method. The latter starts by instantiating an object of the class *AripsoAssociationModel* which will ensure the persistence of information. While reading the file, the discovery of a tag described in Table 6.1 triggers the call of the corresponding method named *baliseXxx*, where *Xxx* denotes the name of the discovered tag. The method activates its counterpart *addXxx* of the object *AripsoAssociationModel*. At the end of treatment, *LoaderPmml* returns the object *AripsoAssociationModel*.

Complementary, the class *AripsoAssociationModel* allows to save a PMML file by calling the *SaveRA* method.

6.5.2 Rule Schema Management Module

In this section, we will present and discuss the components of the rule schema module. Choosing the representation language for this type of knowledge was a crucial step. We proposed an XML-based language in order to ensure the persistence of the schemas. Actually, XML-based languages have a great advantage – they have a low error level comparing to other representation languages. Indeed, the parser APIs takes in charge the entire file structure, so that the user should take in charge only the management of these knowledge in tags and attributes.

The proposed XML-based formalism is defined by easily comprehensible tags and attributes as presented in Table 6.2.

The Management of Rule Schemas module allows two main actions:

- creating Rule Schemas;
- or using Rule Schemas.

The former consists of an interface which helps user in creating the rule schemas. A tree of the ontology concepts is presented, thus the user can choose the concepts that

TABLE 6.1: Description of the tags in PMML Files.

XML Elements	Type	Description
X-QualityIndice	Tag	Defines a quality metric
Id	Attribute	Metric ident
name	Attribute	Metric name
Item	Tag	Defines an item
Id	Attribute	Item ident
name	Attribute	Item name
ItemSet	Tag	Defines an itemset
numberOfItems	Attribute	Number of items
support	Attribute	itemset support
Id	Attribute	Itemset ident
ItemRef	Tag	Ident of the item composing the itemset
itemref	Attribute	Reference item
AssociationRule	Tag	Defines an association rule
confidence	Attribute	Rule Confidence
support	Attribute	Rule support
antecedent	Attribute	Antecedent itemset
consequent	Attribute	Consequent itemset
X-QIV	Tag	Additional metric
id	Attribute	Metric ident
value	Attribute	Metric value

he/she wants to use. Moreover, the latter has the possibility to choose a restriction, general or leaf concept.

The second action allows the user to use already defined rule schemas. Thus, it is possible to manipulate the components of a rule schema – to read and to save (in XML files) –, and to transform these schemas in regular expressions in order to be further used.

The entire process of Rule Schema Management is quite comparable to the one of Association Rule Management. The diagram class in Figure 6.6 presents the management process which works as follows: the *LoaderSchemas* class controls the XML file reading task (containing rule schemas) by calling the SAX API. As in the case of PMML files, the latter allows an event-based reading file process.

Further, *LoaderSchemas* class instantiates an *AripsoSchemas* object containing the entire set of rule schemas. We note also that each class *AripsoSchema*, *AripsoSchItemSet*, *AripsoSchItem AripsoSchItemElem* have a *toString()* method which returns to the user their information in the form of regular expressions. Similarly, they propose the method *saveSchema()* which saves in XML the information.

TABLE 6.2: Description of the Tags of XML Files.

XML Elements	Type	Values	Description
Schemas	Tag		Defines the entire set of Rule Schemas
Schema	Tag		Defines a Rule Schema
type	Attribute	GI	Non-implicative Rule Schema
		RPC	Implicative Rule Schema
action	Attribute	Matching	Confirming Operator
		Pruning	Pruning Operator
		Exception	Exception Operator
ItemSet	Tag		Itemset
member	Attribute	Condition	The itemset is an antecedent
		Consequent	The itemset is a consequent
		Both	The type is unknown – used for GIs
Item	Tag		Defines an item and its cardinality
card	Attribute		Item cardinality
ItemElem	Tag		Describes an item

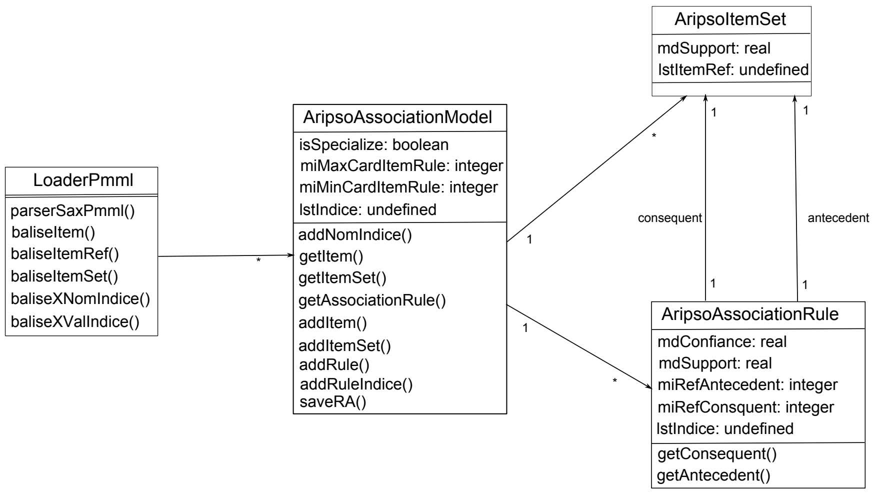


FIGURE 6.6: Class diagram for rule schemas management module.

6.5.3 Ontology Engine Module

The Ontology Engine Module is the core of our tool. In Section 5.2.2 on page 103 we discussed the three types of concepts available in the ontology – leaf, generalized and restriction. Our rule schema management module can access all these three types of concepts, whatever its semantic level. This is possible due to the individuals-based description of ontologies and to reasoning techniques.

In Section 5.2.2.2 on page 106 (and in Example 5.2.10 on page 108) we detailed in Description Logic formalism the interest of reasoners in our tool. In this section, we will show how this is done, and which algorithm the technique follows.

The main reasoners action that we use in AR*IPSO* tool is the recreation of relationships between individuals and concepts. Thus, after reasoning, we could clearly verify which individuals belongs to which class. In other words, for restriction concepts, we will be able to say which are the individuals that are gathered in, and, further, to define the set of items that produce the connection of restriction concepts with the database.

AR*IPSO* Tool uses reasoners in the conforming association rules – rule schemas task, which can be generally reduced to itemsets – concepts conforming (Section 5.2.4.2, Definition 5.2.13). As defined, an itemset is conforming to a concept, if the concept is connected to at least an item of the itemset in the database. The action of testing the conformity between an association rule and a concept is accomplished by the following phases (the algorithm pseudocode is presented in Table 6.3):

- *The initialization phase.* In order to optimize the execution time, before the analysis, for all the association rules we instantiate the ontology with the association rules items;
- *Reasoning phase.* In this phase, an inference engine is called. In our tool we chose Pellet reasoner. After this phase, all (restriction) concepts should contain their individuals;
- *The analyzing phase.* In this phase, each association rule is tested against each rule schema to evaluate the conforming status. First, we define the set of individuals corresponding to association rule items in the ontology. Second, for each concept of rule schema we verify if at least one of its individuals is also an individual of an item.

6.5.4 Additional Filters Modules

In this section we propose the technical solutions for the two additional filters that we integrated in our framework. First, we will discuss about minimum improvement constraint filter, and second, we will detail item-relatedness filter implementation.

6.5.4.1 Minimum Improvement Constraint Filter Module

As already stated, the minimum improvement constraint filter implemented in AR*IPSO* approach is based on the proposition made in [20]. In short, an association rule $X \rightarrow Y$ $[s,\ c]$ – with the support and the confidence $c\%$ – is interesting (i.e. it brougths important information to the rule set) if all its ancestors $X_i \rightarrow Y$ $[s_i,\ c_i]$ with $X_i \subseteq X$ have their confidence lower than the confidence of the more specific rule: $\forall i, c_i \leq c$.

The function of filtering is implemented at the rule management level – in the *AripsoAssociationModel* class.

Table 6.3: Association Rule – Rule Schema conforming algorithm.

```
Input: Rule Schema ($rs$), Association Rules Set ($AR$), Ontology ($O$)
Output: Association Rules conforming to the Rule Schema ($AR_{rs}$)

1.  $AR_{rs} = \emptyset$
2.  forall association rules $r$ in $AR$ do
3.      forall items $i$ in $r$ do
4.          create individual $i_0$ in the ontology $O$
                    as correspondence of the item $i$
5.  call Pellet_inference_engine
6.  forall association rules $r$ in $AR$ do
7.      $bool$ = TRUE
8.      forall concepts $C$ in $rs$ do
9.          $I_C$ = list of individuals belonging to concept $C$ in $O$
10.         $I_r$ = list of individuals in $O$ corresponding
                    to all items $i$ of the rule $r$
11.         if ($I_C \cap I_r = \emptyset$) then
12.             $bool$ = FALSE
13.     if ($bool$ = TRUE) then
14.         $AR_{rs} = AR_{rs} \cup \{r\}$
15. return $AR_{rs}$
```

The algorithm developed in our tool is described in Table 6.4. The process of rule filtering starts with the 2-length rules (with one item in the antecedent and one in the consequent), and continues with longer rules. Further, the main principle is to start with the complete set of association rules, and to remove the rules which do not satisfy the minimum improvement constraint.

For instance, the algorithm starts with the entire set of association rules. For a given k length, in line 2, the set of possible ancestor rules (A) of the k-length rules is initialize by the complete set of association rules with the length from 2 to $k-1$.

The next step is to prune the uninteresting rules from the set $AR(k)$. In this purpose, for each association rule of length k in $AR(k)$ we test that if a rule in A is confirmed to be its ancestor, the confidence of the latter does not overpass the confidence of the former; in a contrary case, the tested association rule is pruned from AR.

In order to reduce execution time, we use in the algorithm special structures as temporary hash trees. The first hash tree is developed to save the association rules of k-length, and the second hash tree is used to define the set of ancestors formed by all the rules of length less than k.

When an association rule has a confidence lower than its ancestor, its reference is

TABLE 6.4: Minimum improvement constraint filter algorithm.

```
Input: Association Rule Set ($AR$)
Output: Filtered Association Rules (filtered $AR$)

1. forall $k$ possible lengths of a rule do
2.     $A = AR(2 \dots k-1)$
3.     forall rules $rc$ in $AR(k)$ do
4.         forall rules $a$ in $A$ do
5.             if ($a$ is an ancestor of $rc$) then
6.                 $a.D = a.D \cup \{rc\}$
7.                 if ($rc.conf \leq a.conf$) then
8.                     delete $rc$ from $AR(k)$
9. return $AR$
```

removed from the list of temporary association rules of k length. At the end of the process, only the references of not pruned rules are kept in memory.

6.5.4.2 Item-relatedness Filter Module

Already discussed in previous sections, Item-Relatedness measure comes to help the user to prune trivial association rules. The triviality is here represented by a semantic proximity between the items of a rule (i.e. *red_grape* fruits are very close to *white_grape*). The general principle of this item-relatedness metric measuring the semantic proximity is discussed in [148].

In Section 5.2.4.3.2 on page 119, we discussed the integration of such a metric in our framework and the advantages that it brings. In this section we present the methodology that we developed in order to implement the item-relatedness metric.

First, we will discuss the technique to compute the item-relatedness metric between two items. In the following we will consider the taxonomic part of the ontology, the distance between the items being determined by the number of arcs that separate the concepts that match the items. The algorithm proposed to compute the item-relatedness measure is describe in Table 6.5.

First, the concepts related to the two items are determined, and they are declared as starting points of the algorithm. The technique is to make a bottom-up partial traversal of the ontology, level by level, till we find a common ancestor. When passing from level l to level $l-1$, the distance between the two items increases by two points. Next, specific methods are proposed for the cases of asymmetric paths – one path is longer than the other.

TABLE 6.5: The algorithm for the computation of the distance between two items.

```
Input: Ontology ($O$), First Item ($i_1$), Second Item ($i_2$)
Output: Distance between the two items ($d$)

1. $C_1$ = the ontology concept containing
          the instance corresponding to $i_1$ item
2. $C_2$ = the ontology concept containing
          the instance corresponding to $i_2$ item
3. $d = 0$
4. while ($C_1 \neq$ ROOT) & ($C_2 \neq$ ROOT) & ($C_1 \neq C_2$)
          & ($C_1$ is not a child of $C_2$) & ($C_2$ is not a child of $C_1$)
5.     $C_1$ = the parent of $C_1$
6.     $C_2$ = the parent of $C_2$
7.     $d = d + 2$
8. $bool = false$
9. if ($C_2$=ROOT & $C_1 \neq$ ROOT) OR & ($C_1$ is not a child of $C_2$)
10.     $child\_concept = C_1$
11.     $parent\_concept = C_2$
12.     $bool = true$
13. if ($C_1$=ROOT & $C_2 \neq$ ROOT) OR & ($C_2$ is not a child of $C_1$)
14.     $child\_concept = C_2$
15.     $parent\_concept = C_1$
16.     $bool = true$
17. if ($bool$)
18.     while ($child\_concept \neq parent\_concept$)
19.         $child\_concept$ = parent of $child\_concept$
20.         $d++$
21. return $d$
```

6.5.5 Exceptions Generating Module

The various features that we have detailed reflect the same need: to find the strongest rules from a statistical point of view, but also the most unexpected for the user. In this context, we could not ignore exception rules [61] which are briefly described in Section 5.2.4.2 on page 117.

A first idea on implementing this types of filtering process was to discover exception rules from the entire set of association rules stocked in PMML file. During different experimentations, we noted that a limited number of rules is extracted due to support limit. Moreover, as a general note, in classical extraction techniques, it is always difficult to decrease the minimum support threshold in order to find rare association rules. As a consequence, a second method of discover exceptions is to find them in the initial database.

First, we will discuss the generation of exceptions using the set of association rules. For the implementation technique, we were motivated by reusing the implementation of the rule schemas management module. In this purpose, we proposed an algorithm for generating a rule schema which describes the researched exceptions. For instance, if we are searching the exceptions of the rule schema $X \to Y$, we will generate a rule schema $rs < X, z \to \neg Y >$ which takes the form of an exception with z being an item. Next, this new rule schema will be used to generate conforming rules. As the implementation of this module was already presented in Section 6.5.3, we will detail below the implementation of the second type of exception generation.

In this second approach, we proposed to develop a research of exceptions starting from the initial database stored in an *Arff* file. *Arff* is a type of file storing databases. The first lines contain the description of each attribute with the possible values that it can take, and the following lines represent the transactions. In order to make an efficient research of exception rules, we developed a module that deals with exploiting Arff files. The class diagram is presented in Figure 6.7.

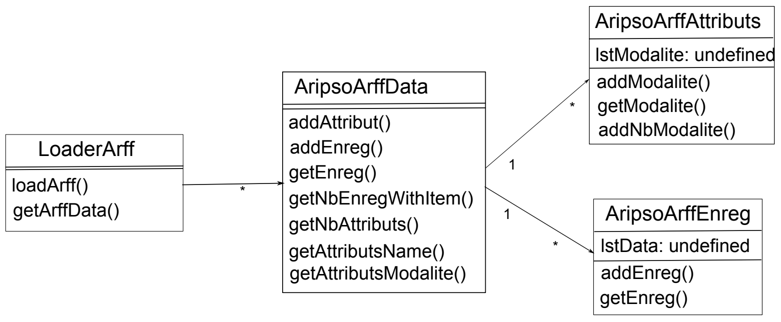


FIGURE 6.7: Class diagram for *Arff* file management module.

The major difference with the exception research in post-processing comes from the step of generation of exceptions candidates. The latter is followed by a validation step which computes the confidence and support of the candidate rule. If the confidence is greater than or equal to the confidence of the general rule, the new exception rule is validated.

From the class diagram we can note that a charging class *LoaderArff* creates an object of the class *AripsoArffData* containing hash trees which permit to accede to the entire set of attributes and transactions. Using *getNbEnregWithItem* method it is possible to determine the number of transactions validating a set of items.

6.6 Development Process

The development process of the AR*IPSO* tool was quite heavy due to the complexity of the proposed approach. In the starting point, the only known need was to post-process rules by schemas and ontologies. During experiments, new functionalities were proposed and they were integrated in the tool. In consequence, the development phase was accomplished as an iterative process which can be described as in Figure 6.8[9].

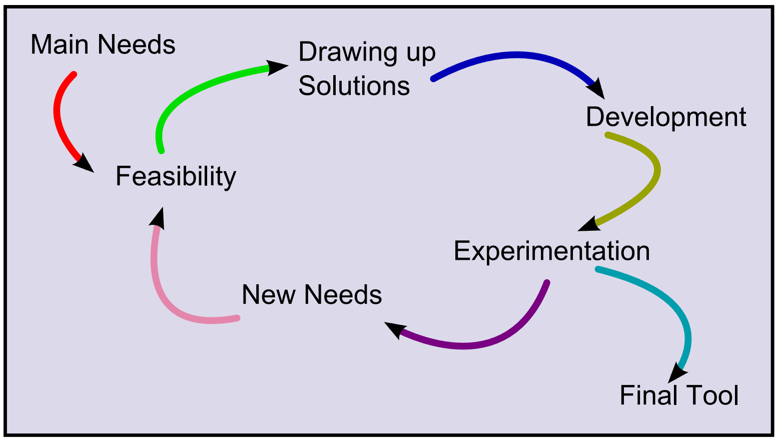


FIGURE 6.8: Iterative cycle of the development process.

We can remark seven different phases:

- *Main Needs* – this phase represents the starting point of the development process and it describes the initial main needs and objectives of the tool;
- *Feasibility* – main (or new – see below) needs are studied in order to verify if solutions can be proposed;
- *Drawing up Solutions* – in this phase technical specifications are conceived and they help to draw up the final solutions;
- *Development* – implementation of the proposed solution;
- *Experimentation* – tool testing phase;
- *New Needs* – after the experimentation phase, new needs can be proposed, and, further, injected in the development process;
- *Final Tool* – delivery of the final tool.

The interest of this type of process is to be able to deliver a valid tool which fulfills the needs. For the AR*IPSO* tool we met the following main cycles:

- Visualization of association rules, visualization under constraints – in function of the number of items;
- Post-processing by using rule schemas and ontologies;
- Implementation of the minimum improvement constraint;

[9]Adaptation after fr.wikipedia.org

- Implementation of item-relatedness measure;
- Rule exception extraction;
- Interface for the creation of rule schemas.

6.7 Conclusion

This Chapter was consecrated to the implementation of the AR*IPSO* approach. AR*IPSO* tool allows the user to interactively filter the whole set of association rules by means of an iterative process.

We have mainly detailed how the elements which compose the approach were implemented. We discussed different problems that we met during the development and the choices that we made, in particular concerning the software technologies. For the most important modules we described the proposed algorithms and we argued our choices.

7 Experimental Results

However beautiful the strategy, you should occasionally look at the results.

Winston Churchill

Contents

7.1 Introduction

This Chapter is dedicated to experimentations that we conducted in order to test our approaches in an application domain. The studies were driven over a real-life database composed of answers to questionnaires.

For AR*IPSO* approach, the experimentation was carried out in complete cooperation with the Nantes Habitat expert. In a first step, we have established the needs and the objectives that our approach should follow. Next, several sessions were necessary to conceive an ontology structure and to start the Rule Schema development. During different partial result deliveries, these knowledge were refined in order to reach the most appropriate description which will give the expected results presented along this Chapter.

For the AR*LIUS* approach, we describe different case studies which insist on the main characteristics of the new algorithm: an important reduction of the number of association rules proposed to the user, a hight reduction of the execution time and the quality of the generated rules.

7.2 Nantes Habitat Database and Objectives

This study is based on a questionnaire database, provided by Nantes Habitat[1], dealing with customers satisfaction concerning accommodation. Monitoring and Study Services of Nantes Habitat conducts an annual satisfaction investigation over a representative panel of customers. In 2003, a process of rationalization of the information system of Nantes Habitat led to the development of a data center to store the results. In order to extract trends emerging from these investigations, dashboards were implemented. So far, no data mining treatment had been done on these data.

Nantes Habitat has the results of investigations between 2003-2007. Each year, 1,500 out of a total of 21,500 customers are chosen to represent the population. The surveys consist of a set of questions with different answers. The questionnaire consists of 67 different questions with 7 possible answers expressing the degree of satisfaction: *very satisfied*, *quite satisfied*, *rather not satisfied* and *dissatisfied* coded as $\{1, 2, 3, 4\}$, *not applicable* cases coded as 95/96, *don't know* answers coded as 99, and *no answer* – 0.

The table 7.1 introduces a sample of questions with the meaning for each one. For instance, the item $q1 = 1$ describes that a customer is very satisfied by the transport in his/her district ($q1 =$ *"Are you satisfied with the transport in your district"*).

In the questionnaire, the 67 questions are organized in several topics which inspired us for the experimentation phase. A sample of the questions organized in topics is presented in Table 7.2.

The main objective of the Nantes Habitat expert was to find explications for some well-defined dissatisfactions of the customers, such as the price, the district, the services, or to find interesting implications that she did not know.

[1] http://www.nantes-habitat.fr/

Table 7.1: Examples of questions & meaning

Question number and description	
q1	Convenient transport
q2	City center access
q3	Shopping facilities
q11	Is it safe to take a walk in the district at night?
q44	Apartment soundproofing
q47	Apartment ventilation
q48	Apartment strands
q70	Clarity of the documents from Nantes Habitat

Table 7.2: Questions organization in topics.

District
↪ Practical District
↪ q1 – Convenient transport
↪ q2 – City center access
↪ q3 – Shopping facilities
↪ Calm District
↪ q11 — Is it safe to take a walk in the district at night?
Apartment
↪ Comfort Apartment
↪ q44 – Apartment soundproofing
↪ q47 – Apartment ventilation
↪ q48 – Apartment strands
Nantes Habitat Service
↪ Communication
↪ q70 – Clarity of the documents from Nantes Habitat

During our study, we verified the knowledge that classical techniques generate starting from the database. For technical issues, we fixed a minimum support of 2%, a maximum support of 30%, and a minimum confidence of 80% for the association rules mining process. Among available algorithms, we used the *Apriori* algorithm in order to extract association rules and $358,072$ rules were discovered.

For instance, the following association rule describes the relationship between the questions $q2$, $q3$, $q47$ and the question $q70$. Thus, if the customers are very satisfied by the access to the city center ($q2$), the shopping facilities ($q3$) and the apartment ventilation ($q47$), then they can be satisfied by the documents received from Nantes Habitat Agency ($q70$) with a confidence of 85.9%.

$$q2 = 1,\ q3 = 1,\ q47 = 1\ \rightarrow\ q70 = 1\ [S = 15.2\%,\ C = 85.9\%]$$

Analyzing manually over 300,000 of association rules is an intractable task for a human. In consequence, the need of efficient and easy filtering techniques becomes an important issue. In the following sections we will study the comportment and the performance of AR*IPSO* and AR*LIUS* approaches over this database.

7.3 AR*IPSO* Approach

In the pre-processing phase, modifications were brought to the database in function of the interest of the expert. First, she decided to remove from the database those answers which do not express satisfaction, i.e. 95, 96, 99 and 0,. This decision was took due to her interest in satisfaction/dissatisfaction answers.

As AR*IPSO* is a post-processing approach, we chose the *Apriori* algorithm to extract the entire set of association rules. As presented above, a set of 358,072 association rules was generated.

7.3.1 Ontology Structure and Ontology-Database Mapping

In the first step of the interactive process described in the Section 5.2.4.1 on page 111, the user develops an ontology on database items. In our case, starting from the database attributes, the ontology was created by the Nantes Habitat expert starting from the topic organization of questions available in the questionnaire. During several sessions, the expert proposed a classification of attributes and items, and, further, she suggested other interesting information by developing her domain knowledge associated to database attributes. In this section, we will present the development of the ontology in our case study.

7.3.1.1 Conceptual Structure of the Ontology

The ontology is one of the most important elements of the AR*IPSO* approach. The reason is that a well-defined user domain knowledge representation brings good results. Inversely, a representation which is not developed enough will limit the quality of generated information and will reduce the efficacy of the method. It is important to note that a ontology perfect structure does not exist, and the most important is to capture all user domain knowledge.

Due to all these reasons, the ontology construction phase took an important time (several sessions). Also, ontology structure was improved during first experimentations, in function of partial results. Thus, the ontology development was an interactive and iterative process.

The final structure of the ontology is composed of two main parts and a sample of it is presented in Figure 7.1. The ontology has the following characteristics: 7 depth levels, a total of 130 concepts among which 113 are primitive concepts and 17 are restriction concepts. Concerning siblings, the concepts have a mean of 6 child concepts, with a maximum of 13 children.

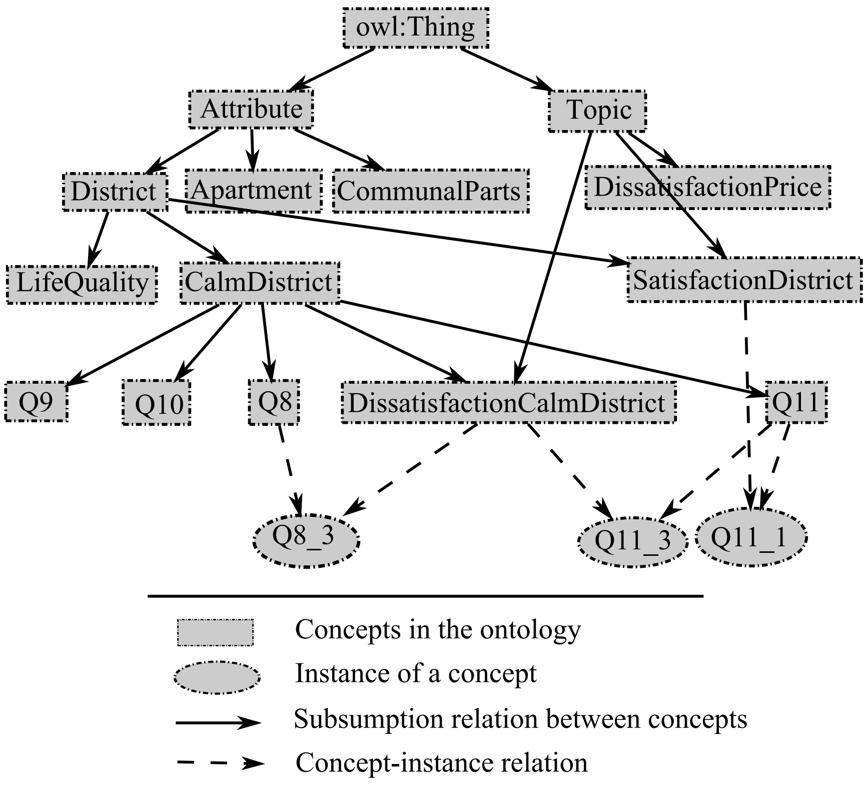


FIGURE 7.1: Ontology structure visualized with *Jambalaya* Protégé plug-in [194] in Protégé software [88] after reasoning.

The first part of the ontology is a database attribute organization with the root defined by the *Attribute* concept; it groups 113 subsumed concepts. The attributes are organized among the question topic in the Nantes Habitat questionnaire. For instance, if we consider the *District* concept, it regroups 14 questions (from $q1$ to $q14$) concerning the facilities and the life quality in a district.

Moreover, the subsumption relation ($\leq$) is completed by the relation $hasAnswer$ associating the *Attribute* concepts to an integer from $\{1, 2, 3, 4\}$, simulating the relation attribute-value in the database. The value 1 represents a high satisfaction, while the value 4 represents a low satisfaction.

The instances in the ontology represent several versions of a question, the difference being made by the value of the answer.

The second hierarchy, $Topics$, regroups all 17 restriction concepts created by the expert using necessary and sufficient conditions.

For instance, let us consider the restriction concept $SatisfactionDistrict$. In natural language, it expresses a satisfaction answer (1 or 2) for the questions concerning the district. The $SatisfactionDistrict$ restriction concept is described in OWL using description logics language by:

$$\begin{aligned} SatisfactionDistrict \equiv District \sqcap hasAnswer \text{ hasValue } 1 \\ \text{OR } hasAnswer \text{ hasValue } 2. \end{aligned}$$

In other words, an individual instantiates the $SatisfactionDistrict$ concept if it represents a question between $q1$ and $q14$ – subsumed by the *District* concept – and if it has an answer denoting satisfaction (the $hasAnswer$ property has the value 1 or 2).

7.3.1.2 Ontology-Database Mapping

The ontology - database connection is made manually by the expert. In our case, with the 67 attributes and 4 values the expert did not meet important problems to realize the connection, but we agree that for large databases, a manual connection could be time consuming. That is why integrating an automatic ontology construction plug-in in our tool is one of our principal perspectives.

As a part of rule schemas, ontology concepts are mapped to database items. Thus, several connections between ontology and database can be designed. Due to implementation requirements, the ontology and the database are mapped through instances.

Thus, using the simplest ontology-database mapping, the expert directly connected one instance of the ontology to an item (semantically, the nearest one). An item is defined by the connection attribute-value (question-value). In this context, an instance (individual) represents the instantiation of a concept (question) and it is connected by the $hasAnswer$ property to the value of the answer. For example, the expert connected the instance $Q11\_1$ to the item $(q11 = 1)$: $f'_0(Q11\_1) = (q11 = 1)$

Then, leaf-concepts (C_0) of the *Attribute* hierarchy were connected by the expert to a set of items (semantically, the nearest one). Considering the concept $Q11$ of the ontology; semantically, it is associated to the attribute $q1$ =*"Are you satisfied with the transport in your district?"*. In Figure 7.1, we consider that the concept $Q11$ has 2 instances describing the question $q11$ with 2 possible answers: 1 and 3. In consequence, the concept $Q11$ was connected by the expert to 2 items, through its two instances, as follows:

$$\begin{aligned} f''_0(Q11) &= \{f'_0(Q11\_1), f'_0(Q11\_3)\} \\ &= \{q11 = 1,\ q11 = 3\}. \end{aligned}$$

The connection of generalized concepts follows the same idea. They are connected through the instances of the concepts that they subsume. For example, the concept *CalmDistrict* is connected to the database as follows:

$$\begin{aligned} f(CalmDistrict) &= \{f''_0(Q8), f''_0(Q9), f''_0(Q10), f''_0(Q11))\} \\ &= \{q8 = 3,\ q11 = 1,\ q11 = 3\}. \end{aligned}$$

The last type of connection implies connecting concepts of the $Topic$ hierarchy to the database. Let us consider the restriction concept *DissastisfactionCalmDistrict*

(Figure 7.2). In natural language, it is defined by all this questions concerning the tranquility of the district (questions $q8$, $q9$, $q10$ and $q11$) having an answer denoting the dissatisfaction of the customer. In our ontology, these questions are represented by 4 concepts subsumed by the *CalmDistrict* concept.

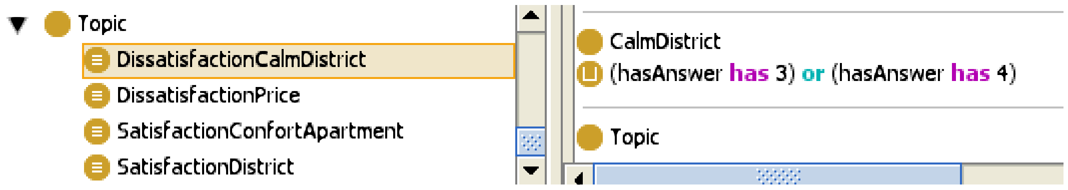

Topic
DissatisfactionCalmDistrict
DissatisfactionPrice
SatisfactionConfortApartment
SatisfactionDistrict
CalmDistrict
(hasAnswer has 3) or (hasAnswer has 4)
Topic

FIGURE 7.2: Restriction concept construction using necessary and sufficient conditions in Protégé.

The *DissastisfactionCalmDistrict* restriction concept is described by the expert using description logics language as follows:

$$DissastisfactionCalmDistrict \equiv CalmDistrict \ \sqcap hasAnswer \text{ hasValue } 3$$
$$\text{OR } hasAnswer \text{ hasValue } 4.$$

Considering that the user has instantiated the concept $Q8$ with the answer 3, and the concept $Q11$ with the answers 1 and 3, instances $Q8\_3$, $Q11\_1$ and $Q11\_3$ are added in the ontology. Then, the concept *DissastisfactionCalmDistrict* is connected in the database as it follows:

$$\begin{aligned} & f(DissastisfactionCalmDistrict) \\ & \quad = \{f_0'(i) \mid i \in Q8 \cup Q9 \cup Q10 \cup Q11 \ \sqcap \ i \ hasAnswer \text{ hasValue } 3 \\ & \qquad\qquad \text{OR } i \ hasAnswer \text{ hasValue } 4\} \\ & \quad = \{f_0'(Q8\_3),\ f_0'(Q11\_3)\} \\ & \quad = \{q8 = 3,\ q11 = 3\}. \end{aligned}$$

7.3.2 Rule Schema Development

Using the ontology proposed in the previous section, the user developed a set of Rule Schema and suggested different Operators to be applied over. First, the expert was interested in finding relations between the dissatisfaction of the customers concerning different elements. For this purpose, she proposed 4 rule schemas presented in Figure 7.3. For example, the RS_3 Rule Schema wants to find the relation that exists between the dissatisfaction of price and the dissatisfaction of common areas.

Further, during different experimentations the expert has noted several implications that she found trivial, but which were not removed by the item-relatedness filter. The expert proposed the set of pruning Rule Schemas in Figure 7.4.

7.3.3 Efficacy Evaluation

This first example proposes to present the efficiency of our new approach concerning the reduction of the number of rules. To this end, we propose to the expert to test the

Table 7.3: Filtering Rule Schemas suggested by the expert.

	Rule Schema	Operator
RS_1	< DissatisfactionPrice >	Conforming
RS_2	< DissatisfactionCalmDistrict >	Conforming
RS_3	< DissasisfactionPrice, DissatisfactionCommonAreas >	Conforming
RS_4	< DissasisfactionPrice → DissatisfactionCommonAreas >	Unexpectedness Exception

Table 7.4: Pruning Rule Schemas proposed by the expert.

	Rule Schema
RS_5	<EntryHall → CloseSurrounding>
RS_6	<Stairwell → EntryHall>
RS_7	<CloseSurrounding → EntryHall>
RS_8	<EntryHall → Stairwell>
RS_9	<CommonAreas → GarbageRoom>
RS_{10}	<TechnicalMaintenance → TechnicalMaintenance>

four filters: on the one hand the pruning filters - *minimum improvement constraint filter* (MICF), *item-relatedness filter* (IRF) and *pruning rule schemas* -, and on the other hand the selection filters - *rule schema filters* (meanings of acronyms in Table 7.5). The expert separately tested each filter and in several combinations in order to compare the results and to validate them.

Table 7.5: Notation meaning.

Nb	The *id* of the filter combination
MICF	Minimum improvement constraint filter
IRF	Item-relatedness filter
PRS	Pruning with Rule Schemas
Rule number	The number of rules remaining after filter application

At the beginning, the expert is faced to the whole set of 358,072 association rules extracted. In a first attempt, we focus on pruning filters. If the *MICF* is applied, all the specialized rules not improving confidence are pruned. In Table 7.6 we can see that the *MICF* prunes *92.3% of rules*, being a very efficient filter for redundancy pruning. In addition, *IRF* prunes *71%* of rules - these rules associating items close semantically. The third pruning filter, *Pruning Rule Schemas*, prunes 43% of rules.

We propose to compare the three pruning filters and the combinations of the pruning filters as presented in Table 7.6. The first line is the reference for our experiments. The rates of number of rules remaining after the three filters are separately used are presented in lines 2, 3 and 4. We can note that the *MICF* filter is the most discriminatory, pruning 92.3% of rules, comparing to other two ones pruning 71% and, respectively, 43% of rules.

TABLE 7.6: Pruning rate for each filter combination.

Nb	MICF	IRF	PRS	Rule number
1				358,072 (100%)
2	X			27,602 (7.7%)
3		X		103,891 (29%)
4			X	207,196 (57%)
5	X	X		16,473 (4.6%)
6	X		X	21,822 (7.7%)
7		X	X	73,091 (20%)
8	X	X	X	13,382 (3.7%)

We can also note, that combining the first two filters, *MICF* and *IRF*, the pruning is more powerful than combining the first one with the third one. Nevertheless, applying the three filters over the set of the association rules implies a rule reduction of 96.3%.

However, applying the most reducing combination, number 8 (Table 7.6), the expert should analyze 13, 382 rules which is impossible manually. Thus, other filters should be applied. The expert was interested in the dissatisfaction phenomena, presented by answers 3 and 4 in the questionnaire. The expert is interested in applying all the rule schemas with the corresponding operator (Table 7.3) for each combination of the first three filters presented in Table 7.6. Table 7.7 presents the number of rules filtered by each rule schema.

In Table 7.7, the first column, *Nb*, represents the identification of each filter combination as denoted in Table 7.6. We can note that the rule schema filters are very efficient. Moreover, studying the dissatisfaction of the clients improves the filtering power of the rule schemas.

Let us consider the *second rule schema*. Applied over the initial set of 358, 072 association rules with the conforming operator, it filters 1, 008 rules representing 0.28% of the complete set. But it is obvious that it is very difficult for an expert to analyze a set of rules of the order of thousands of rules. Thus, we can note the importance of the pruning filters, the set of rules extracted in each case having less

TABLE 7.7: Rates for Rule Schema filters applied after the other three filter combinations.

Nb	$C(RS_1)$	$C(RS_2)$	$C(RS_3)$	$U_p(RS_4)$	$E(RS_4)$
1	185 (100%)	***1,008*** (100%)	96 (100%)	1399 (100%)	98 (100%)
2	92 (49%)	***361*** ***(35%)***	50 (52%)	462 (33%)	48 (48.9%)
3	39 (21%)	***162*** ***(16%)***	39 (40%)	401 (28.7%)	3 (3.1%)
4	107 (57%)	***472*** ***(46%)***	20 (20%)	238 (17%)	96 (6.9%)
5	28 (15%)	***77*** ***(7.6%)***	28 (29%)	187 (13.4%)	3 (3.1%)
6	56 (30%)	***231*** ***(22%)***	14 (14%)	154 (11%)	48 (48.9%)
7	3 (1.6%)	***3*** ***(0.2%)***	3 (3.1%)	24 (1.7%)	3 (3.1%)
8	3 (1.6%)	***3*** ***(0.2%)***	3 (3.1%)	11 (0.8%)	3 (3.1%)

than 500 rules. We can also note that the *IRF* filter is more powerful than the other pruning filters, and that the combination of two filters at the same time gives remarkable results:

- on the fifth line, combining *MICF with IRF* reduces the number of rules to 77 rules;
- combining *IRF with Pruning Rule Schemas* the set of rules is reduced to 3 rules;
- we can also note that in the last two rows the filters have the same results. We can explain this by the fact that we are working on an incomplete set of rules because of the maximum support threshold that we impose in the mining process.

7.3.4 Interactivity and Discovered Rule Quality Evaluation

This second example is proposed in order to outline the quality of the filtered rules, and to confirm the importance of the interactivity in our framework. To this end, we present the sequence of steps (Figure 7.3) performed by the expert during the interactivity process, steps already described in Section 7.3.1. We have already presented the *first step* of the interactive process – ontology construction – in Section 5.2.4.1.

As in the first example, the expert is faced to the whole set of rules. In a first attempt (*step 2 and 3*), she proposed to investigate the quality of rules filtered by two of the rule schemas RS_2 and RS_3 with the conforming operator. The first one deals with *dissatisfaction* concerning the *tranquility in the district*, and the second one searches rules associating *dissatisfaction in price* with *dissatisfaction concerning the common areas of the building*.

Applying these two schemas to the whole rule set, an important selection is made:

- $C(RS_2)$ filters $1,008$ association rules;
- $C(RS_3)$ filters 96 association rules.

The expert is in the visualization and validation steps (*4 and 5*), and she analyses the 96 rules filtered by $C(RS_3)$, because of the reduced number of rules comparing to 1008 filtered by $C(RS_2)$. For example, let us consider the following set of association rules:

$q17$= 4, q26 $= 4, q97 = 4 \rightarrow q28 = 4$ $[S = 2.6\%,\ C = 92.8\%]$

$q16 = 4, q17 = 4, q26 = 4, q97 = 4 \rightarrow q28 = 4$ $[S = 2.5\%,\ C = 92.5\%]$

$q15 = 4, q17 = 4, q97 = 4 \rightarrow q28 = 4$ $[S = 1.9\%,\ C = 80.5\%]$

$q15 = 4, q17 = 4, q97 = 4 \rightarrow q26 = 4, q28 = 4$ $[S = 1.9\%,\ C = 80.5\%]$

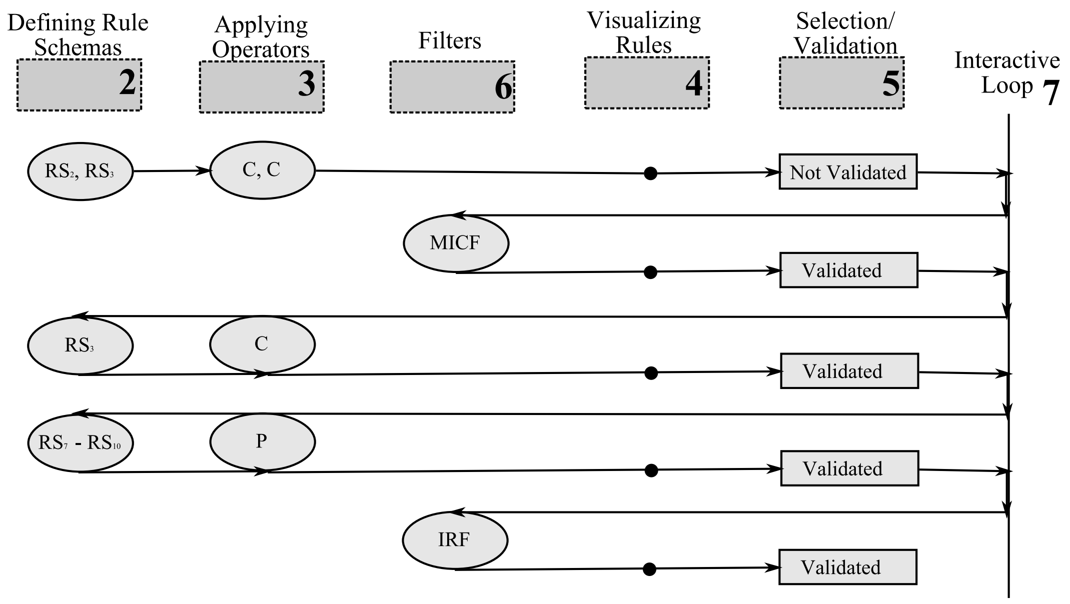


FIGURE 7.3: Description of the interactive process during the experiment.

The expert noted that the second rule is a *specialization* of the first rule - the item $q16 = 4$ is added in the antecedent, and she also noted that *its confidence is* *lower* than the confidence of the more general rule. Thus, the second rule does not bring important information to the whole set of rules, hence, it can be *pruned*. In the same way, the expert noted that the forth rule is the specialization of the third one, and that the confidence is not improved in this case neither. The expert decided to *modify her initial information* (*step 5*) and to *go to the beginning of the process* via the *interactivity loop* (*step 7*), choosing to apply the *MCIF* (*step 6*) which extracts *27,602* rules. The expert decided to keep these results (*step 4 and 5*) and to return in the *interactivity loop*, *going back* to *steps 2 and 3* in order to *redefine rule schemas and operators*.

This time, the expert proposed to use only the rule schema $C(RS_3)$ as a consequence of high volume of rules extracted by the other one. Using $C(RS_3)$, 50 rules are filtered, and the presence of rules 1 and 3, and the absence of rules 2 and 4 (from the set presented above) validate the use of *MICF* (*step 4 and 5*). Moreover, the hight reduction of number of rules validates the application of $C(RS_3)$. In this state, the expert returned to *step 2* in order to modify the rule schema proposing RS_4 and first, she applied the *unexpectedness regarding the antecedent operator*, $U_p(RS_4)$, and then she returned to step 3 in order to *modify the operator*, choosing the *exception* one $E(RS_4)$. This results are briefly presented in Table 7.7, but due to space limit they are not detailed in this Section.

The expert analyzed the 50 rules extracted by $C(RS_3)$ and she found several trivial implications noting that the implication between several items did not interest her. For instance, let us consider the following set of rules:

$$q17 = 4, q97 = 4 \rightarrow q16 = 4\ C = 86.7\%\ S = 3.5\%$$
$$q25 = 4, q28 = 4, q97 = 4 \rightarrow q26 = 4\ C = 100\%\ S = 2.0\%$$

This rules imply items from *EntryHall* and *CloseSurrounding*, thus the expert proposed to apply rule schemas RS_5 to RS_8 with the pruning operator (*step 2 and 3*) in order to prune those not interesting rules. In consequence, 15 rules are extracted, and the absence of the above rules validates the application of pruning rule schemas (*step 4 and 5*).

Let us consider the following two rules:

$$q28 = 4, q97 = 4 \rightarrow q17 = 4\ C = 81.1\%\ S = 2.9\%$$
$$q8 = 4, q16 = 4, q97 = 4 \rightarrow q9 = 4\ C = 88.6\%\ S = 2.1\%$$

The expert noted that a great part of the 15 rules are implications between attributes subsumed by the same concept in the ontology. For instance, the attributes $q28$ and $q17$ of the first rule, described by the $Q28$ and the $Q17$ concepts, are subsumed by the concept *Stairwell*. Similarly, for the second rule, $q8$ and $q9$ are subsumed by *CalmDistrict* concept. Thus, the expert applied the *IRF* filter, and only 3 rules are filtered. One of those rules attracts the interest of the expert:

$$q15 = 4,\ q16 = 4,\ q97 = 4\ \rightarrow\ q9 = 4$$
$$Support = 2.3\%\ Confidence = 79.1\%$$

which can be translated by: if a customer is not satisfied with the cleaning of the close surrounding and of the entry hall, and if he is not satisfied with the service charges, then it is possible with a confidence of 79.1% that he consider that his district has a bad reputation. This rule is very interesting because the expert thought that the building state does not influence the opinion concerning the district, but it is obvious that this is the case.

It is very important to note that the quality of the selected rules was certified by the Nantes Habitat expert.

7.4 AR*LIUS* Approach

For the AR*LIUS* approach, because the algorithm needs binary attributes, items of the form *question=answer* are formed, for example the item $Q35\_1$ represents an answer of *very satisfied* to the question number $Q35$ - about the *lighting of common spaces*. Therefore there are a total of 624 items. For testing, two databases were used, from year 2003 and 2006, containing 1490 transactions each.

7.4.1 Case Study 1

In the first case study, the analyst is interested on the dissatisfaction related to common spaces in the proximity of the accommodation. The search starts with unsatisfied answers of two questions related to the problem: question 7 – *The quantity of green spaces is sufficient* and question 65 – *The cohabitation difficulties are efficiently treated.* The analyst already knows the direction of the implication (which is quite obvious) and creates a Rule Schema of the form:

$$rs([Q7\_4] \to [Q65\_4]\ \ [\,])$$

with no support and confidence constraints.

The application of an *1*-Specialization Operator over this Rule Schemas has the following output:

$62 results:$
$[Q7\_4,\ Q72\_4] \to [Q65\_4]\ \ [S = 1.2\%,\ C = 81.8\%]$
$[Q7\_4,\ Q93\_4] \to [Q65\_4]\ \ [S = 1.2\%,\ C = 78.2\%]$
$[Q7\_4,\ Q37\_4] \to [Q65\_4]\ \ [S = 0.9\%,\ C = 76.4\%]$
$[Q7\_4,\ Q55\_4] \to [Q65\_4]\ \ [S = 0.8\%,\ C = 75.0\%]$
$[Q7\_4,\ Q22\_4] \to [Q65\_4]\ \ [S = 0.5\%,\ C = 70.0\%]$
$\ldots$

It is the rule with the highest confidence that shows the relationship with $question72$ – *In case of works performed in the common spaces, are you satisfied with the information received.* This relation with maintenance works leads the analyst to want to add another item to the search: the unsatisfied answer to $question58$ – *Delays of the interventions to repair degradation of common spaces.* However the analyst does not know how this answer relates to the other elements in the rule, so he just puts it in the *General* part of the Schema. Thus, an operation of Confirmation is performed on the resulted Rule Schema

$$rs([Q7\_4] \rightarrow [Q65\_4]\ \ [Q58\_4])$$

and it shows that item $Q58\_4$ is most likely in the antecedent rather than in the consequent because the confidence of the first rule is greater that the confidence of the second one:

2results :
$[Q7\_4,\ Q58\_4] \rightarrow [Q65\_4]\ \ [S = 1.4\%,\ C = 75.0\%]$
$[Q7\_4] \rightarrow [Q65\_4, Q58\_4]\ \ [S = 1.4\%,\ C = 18.2\%]$

Now, the analyst is interested to find which other elements are related to this rule, thus he/she runs a *1*-Specialization on the same Rule Schema. The results are interesting:

72results :
$[Q7\_4,\ Q39\_4,\ Q58\_4] \rightarrow [Q65\_4]\ \ [S = 0.8\%,\ C = 100.0\%]$
$[Q7\_4,\ Q49\_4,\ Q58\_4] \rightarrow [Q65\_4]\ \ [S = 0.9\%,\ C = 93.3\%]$
$[Q7\_4,\ Q25\_4,\ Q58\_4] \rightarrow [Q65\_4]\ \ [S = 0.9\%,\ C = 92.9\%]$
$[Q7\_4,\ Q93\_4,\ Q58\_4] \rightarrow [Q65\_4]\ \ [S = 0.7\%,\ C = 91.7\%]$
$[Q7\_4,\ Q57\_4,\ Q58\_4] \rightarrow [Q65\_4]\ \ [S = 0.7\%,\ C = 90.9\%]$
$[Q7\_4,\ Q50\_4,\ Q58\_4] \rightarrow [Q65\_4]\ \ [S = 0.5\%,\ C = 88.9\%]$
$[Q7\_4,\ Q6\_1,\ Q58\_4] \rightarrow [Q65\_4]\ \ [S = 0.5\%,\ C = 87.5\%]$
$[Q7\_4,\ Q63\_4,\ Q58\_4] \rightarrow [Q65\_4]\ \ [S = 0.9\%,\ C = 86.7\%]$
$[Q7\_4,\ Q5\_1,\ Q58\_4] \rightarrow [Q65\_4]\ \ [S = 0.8\%,\ C = 85.7\%]$
$[Q7\_4,\ Q2\_1,\ Q58\_4] \rightarrow [Q65\_4]\ \ [S = 1.1\%,\ C = 84.2\%]$
...

A number of relevant conclusions was drawn related to the dissatisfaction about common spaces. The rules extracted have low support showing that there is a less probability for the rules to be trivial. In the same time, the high confidence shows that the rules are important and that they must be considered. The new items in the conditions of the extracted rules show that the problem relates to unsatisfied answers to other questions like:

- $Q39$ – on the building's ambiance,
- $Q25$ – state of building's proximity, or
- $Q57$ – important reparations on common parts.

However, the last rules in the first 10 results are unexpected: the problem relates with very satisfied answers to the following three questions: $Q2$, $Q5$ and $Q6$ – *access* *to the city center*, *administrative services* and *parking spaces.* This can mean that the problems of common spaces and repair works in the common spaces are most likely related to the accommodations closer to the center. That may be considered as valuable information.

7.4.2 Case Study 2

In this second example, the expert knows that there is a relation between questions $Q13$ and $Q2$ – *There are sufficient spaces for children* and *Access to the city center*, more precisely between the satisfaction of the two questions. A Confirmation Operator on the following Rule Schema

$$rs([\,] \rightarrow [\,]\ \ [Q13\_1,\ Q2\_1])$$

shows that the implication is from the first question to the second one, due to the high confidence of the first rule:

2 results :
$[Q13\_1] \rightarrow [Q2\_1]\ \ [S = 15.30\%, C = 75.50\%]$
$[Q2\_1] \rightarrow [Q13\_1]\ \ [S = 15.30\%, C = 23.46\%]$.

The first rule that resulted is strong (confidence of 75.50%), so the analyst wants to know if there are any exceptions to the rule. She performs a *1*-Exception operation and she gets the result below:

1 result :
$[Q1\_4, Q13\_1] \rightarrow [Q2\_2]\ \ [S = 0.20\%, C = 100.00\%]$.

Although the resulting rule has a low support, it has a very strong confidence, which is much higher than the confidence of the initial rule, even if the support is smaller. Moreover, the confidence is higher than the confidence of the rule $Q1\_4 \rightarrow Q2\_2\ \ [S = 0.87\%\ \ C = 56.52\%]$, so it is the *association* of $Q1\_4$ and $Q13\_1$ that leads to the result.

The result shows that, although generally the satisfaction about spaces for children leads to a satisfaction about the access to the center, in the special case where there is dissatisfaction related to transportation (*question1*), people tend to be less satisfied about access to the center.

7.4.3 Case Study 3

In this last case study the expert is interested on what relates to the implication of the dissatisfaction about *price* – *question97* – on the dissatisfaction about the *state of the entry hall* – *question26* – as a part of the common spaces in the building. The search starts with a Rule Schema containing unsatisfied answers to two questions related to the problem: $Q97$ and $Q26$. Support must be reasonable (2%) and confidence must be high (95%). The expert formalized this situation by the following Rule Schema:

$$rs([Q97\_4] \rightarrow [Q26\_4]\ []\ \ [2\%,\ 95\%]).$$

An operation of *2*-Specialization applied over has the following output, finding items that are related to the implication:

17 results :
$[Q29\_4, Q65\_4, Q97\_4] \rightarrow [Q26\_4][S = 2.6\%, C = 97.5\%]$
$[Q29\_4, Q32\_4, Q97\_4] \rightarrow [Q26\_4][S = 2.4\%, C = 97.3\%]$
$[Q16\_4, Q29\_4, Q97\_4] \rightarrow [Q26\_4][S = 4.6\%, C = 97.1\%]$
$[Q29\_4, Q60\_1, Q97\_4] \rightarrow [Q26\_4][S = 2.2\%, C = 97.0\%]$
$[Q29\_4, Q37\_4, Q97\_4] \rightarrow [Q26\_4][S = 2.0\%, C = 96.7\%]$
$[Q18\_4, Q31\_4, Q97\_4] \rightarrow [Q26\_4][S = 2.0\%, C = 96.7\%]$
...

The output shows a number of interesting relations: apart from dissatisfaction about the state and cleanness of common spaces (entry hall, building level, etc) and equipment (interphone), there is also an indication about the efficiency with which the technical requests are addressed ($Q65$). There is also an interesting rule, the forth one:

$$[Q29\_4, Q60\_1, Q97\_4]- > [Q26\_4][S = 2.2\%,\ C = 97.0\%];$$

there is a strong implication between the state of the building (particularly the level – $Q29$), the price ($Q97$) and the *customer expectations* ($Q60$), in that the expectations actually correspond, although the price is considered too high.

Next, the expert might also want to check if there are exceptions to the specified Rule Schema. For this purpose, the expert lowered the support threshold to 1% – exceptions are rare rules –, and she applied the *2*-Exception Operator over the new Rule Schema

$$rs([Q97\_4] \rightarrow [Q26\_4]\ []\ \ [1\%,\ 95\%]).$$

The results are presented below:

15 results :
$[S\_boi, Q18\_1, Q97\_4] \rightarrow [Q26\_1][S = 1.5\%,\ C = 100.0\%]$
$[S\_boi, Q29\_1, Q97\_4] \rightarrow [Q26\_1][S = 1.5\%,\ C = 100.0\%]$
$[S\_boi, Q16\_1, Q97\_4] \rightarrow [Q26\_1][S = 1.3\%,\ C = 100.0\%]$
$[S\_boi, Q47\_1, Q97\_4] \rightarrow [Q26\_1][S = 1.2\%,\ C = 100.0\%]$
$[S\_boi, Q78\_1, Q97\_4] \rightarrow [Q26\_1][S = 1.2\%,\ C = 100.0\%]$
$[S\_boi, Q92\_1, Q97\_4] \rightarrow [Q26\_1][S = 1.0\%,\ C = 100.0\%]$
...

This result is very interesting. It shows that dissatisfaction in price can actually lead to a better opinion on the building, if it is connected, among with some other items, with a certain neighborhood (*Boissiere* – S_boi), which is also very calm ($Q18$). This is important to know, because, although customers are quite happy with the conditions in that neighborhood, they are unhappy with the price.

Considering the rules discovered by the *2*-Exception Operator, the expert might want to look a bit more into the first rule, so she performs a *2*-Generalization over the following Rule Schema:

$$rs([S\_boi, Q18\_1, Q97\_4] \rightarrow [Q26\_1]\ []).$$

The extracted rules

3 results :

$[S\_boi] \rightarrow [Q26\_1][S = 5.0\%, C = 88.1\%]$

$[Q97\_4] \rightarrow [Q26\_1][S = 22.5\%, C = 59.1\%]$

$[Q18\_1] \rightarrow [Q26\_1][S = 63.2\%, C = 76.2\%]$

show that, indeed, the *Boissiere neighborhood* has a great influence on the satisfaction about state of the hall ($Q26$) – first rule; this is outlined by the important confidence of the rule (88.1%).

Last, the expert want to investigate the relation of the first rule in the last result with the satisfaction about adequacy of Nantes Habitat services ($Q92$). In this case, the expert considers that less frequent rules could interest him/her, and less stronger also, thus he/she decreases the support and confidence value. The expert performs a Confirmation operation on the following Rule Schema:

$$rs([S\_boi] \rightarrow [Q26\_1]\ \ [Q92])\quad [1\%\ 60\%].$$

We can note here that in the element $Q92$ the expert gives only the attribute, without proposing a certain value for. That is to say that, in the proposed framework, the expert has two possibilities to denote elements in a general way – all values can be taken bu the attribute, or in a more specific way – values for attributes are specified.

There are 2 results:

$[S\_boi, Q92\_1] \rightarrow [Q26\_1]\ \ [3.7\%, 88.7\%]$

$[S\_boi] \rightarrow [Q26\_1, Q92\_1]\ \ [3.7\%, 65.4\%]$

It appears that the rules relate more to the satisfaction answer to the question92 (Q92_1) and it is usually perceived by the customers in this neighborhood that a good adequacy of the agency's services leads, among others, to a good state of the building.

7.5 Conclusion

This Chapter presented the experimentations that we conducted in order to test our approaches in an application domain. The studies were driven over a real-life database composed of questionnaires answers.

For AR*IPSO* approach, the experimentation was carried out in a complete co-operation with the Nantes Habitat expert. This Chapter outlines the efficiency of AR*IPSO* and the hight quality of discovered rules.

For the AR*LIUS* approach we described different case studies which focused on the main characteristics of the new algorithm: an important reduction of the number of association rules proposed to the user, a hight reduction of the execution time and the quality of generated rules.

8
Conclusion and Perspectives

This thesis is concerned with the merging of two active research domains: Knowledge Discovery in Databases (KDD), more precisely the Association Rule Mining technique, and Knowledge Engineering (KE) with a main interest in knowledge representation languages developed around the Semantic Web.

In Data Mining, association rule mining algorithms generate huge volumes of rules that cannot be directly used. As a consequence, it is necessary to develop a post-processing task with the main goal to find interesting rules among the complete delivered set. As a first solution, the analyzing task was proposed to be done manually by the user; obviously, when the complete set exceeds 100 rules, the analyzing task becomes impossible. In this context, it is important to help the user during this phase with efficient post-processing techniques.

Our work addresses two main issues: the integration of user knowledge in the discovery process of association rules, and the interactivity with the user. The first issue requires defining an adapted formalism to express user knowledge with accuracy and flexibility. For this purpose, we take advantage from the research carried out in the Semantic Web field, and more precisely from the representation languages developed in order to be used as user knowledge representation in the discovery process. Second, the interactivity with the user allows a more iterative mining process where the user can successively test different hypotheses or preferences and focus on interesting rules.

Contributions

Our main contributions are organized around 3 areas: techniques for post-processing and mining association rules, ontologies for knowledge management, and development and validation of tools.

A model to represent user knowledge

We proposed a new model to represent user knowledge in the discovery process of association rules. It is composed of three formalisms:

1. **Ontologies** permit the user to express his/her domain knowledge by means of a high semantic model. Its main advantage is to integrate reasoning techniques which are comparable with the human thinking;

2. **Rule Schema** defines a new rule-like formalism which allows the user to describe his/her expectations regarding the discovered rules. In order to benefit from the flexibility and complexity of the ontologies, Rule Schemas are defined through ontology concepts;

3. **Operators** bring the interactivity in the rule mining process. We designed them to be applied over Rule Schemas and to help the user to take several actions like pruning or selecting over the set of discovered rules. We proposed the user 4 Operators: Pruning, Conforming, Unexpectedness and Exception.

   For instance, if the user chooses the Pruning Operator to be applied over a Rule Schema, all the rules conforming to the Rule Schema are removed from the rule set.

AR*IPSO* – An innovative post-processing approach

The main contribution of this thesis consists in proposing an innovative approach, called AR*IPSO* (Association Rule Interactive Post-processing using rule Schemas and Ontologies), helping the user to reduce the volume of the discovered rules and to improve their quality.

Different elements outline the innovative characteristic of our approach. First, it relies on the knowledge representation model that we proposed, which supports the process of integration of user knowledge and expectations in the mining process. Second, it proposes to combine different interestingness measures that assess the quality of the discovered rules. And third, it consists in an interactive process which allows the user to change the provided information at each step and to reiterate the post-processing phase which produces new results.

Implementation of AR*IPSO* tool

The technical realization of this thesis consists in the implementation of the proposed approach in post-processing. For this purpose, we developed a tool which is complete and operational, and which implements all the functionalities described in the approach. More particularly, it proposes a visualization interface which helps the user in editing his/her knowledge and in validating the discovered rules. From a technical point of view, the implementation makes the connection between different elements like the set of rules and rule schemas stored in PMML/XML files, and the ontologies stored in OWL files and inferred by the Pellet reasoner.

AR*LIUS* – An adapted implementation without post-processing

Working in post-processing has multiple advantages such as having a global view over the discovered rules, but also major drawbacks. The most significant is that extracting all association rules, using classical techniques, is time and resource consuming; this process can quickly became intractable. In this context, we proposed a new implementation of our approach, consisting in an interactive local mining process guided by the user, that we called AR*LIUS* (Association Rule Local Interactive mining Using rule Schemas). It allows the user to focus on interesting rules without the necessity to extract all of them and without minimum support limit. In this way, the user may explore the rule space incrementally, a small amount at each step, starting from his/her own expectations and discovering their related rules.

Experimental studies

A significant part of our work relies on testing and validating the proposed approaches. The experimental study aims to analyze the approach efficiency and the discovered rule quality. For this purpose, we used a real-life and large questionnaire database concerning customer satisfaction. For AR*IPSO*, the experimentation was carried out in complete cooperation with the domain expert. From an input set of nearly 400 thousand association rules, for different scenarios, AR*IPSO* filtered between 3 and 200 rules validated by the expert. Clearly, AR*IPSO* allows the user to significantly and efficiently reduce the input rule set. For AR*LIUS*, we experimented different scenarios over the same questionnaire database and we obtained reduced sets of rules (less than 100) with very low support.

Perspectives

The validation of AR*IPSO* and AR*LIUS* over new data

The validation of our approaches can be carried out according to two axes. First, they can be tested on different data in collaboration with domain experts. These experimentations will validate the approaches over new data, but, more importantly, they will be done by distinct experts having completely different expectations. Thus, we will be able to validate the interactivity of the approaches with the user and the quality of the interestingness measures. Moreover, through their different needs, the new experts could help us to enrich the approaches with new interestingness measures.

Second, it will be possible to compare our approaches with the existing ones. In the case of user-driven approaches, comparing them with other approaches is a difficult task due to the lack of benchmarks. Nevertheless, proposing to domain experts to test different approaches following a well-defined scenario could be a solution. In this context, notions such as quality, rapidity, satisfaction and facility could be assessed during the process.

Towards an Oracle[1]-based implementation of AR*IPSO*

During the experimentation phase, AR*IPSO* generated remarkable results. Nevertheless, we noted several drawbacks. First, the current *Apriori*-based mining tools that we used in order to extract the complete set of association rules have an execution time and use a memory percentage that grow with the increase of database dimensions and/or the decrease of the given support thresholds. In this context, it is important to bear in mind that interesting rules have low support, while trivial ones have high support. When searching interesting rules, decreasing considerably the support threshold causes the explosion of the memory used, the execution time of the mining process, and makes the process intractable. Second, after generating a complete set of rules, the execution of the procedures that we proposed in order to filter interesting rules have an execution time that grows along with the increase of the number of rules.

In this context, we were interested in the possibilities to reduce time and resources. Using database management systems could considerably improve the percentage of memory used by the processes that access database information repeatedly. Moreover, in the past decade, some database management systems have proposed embedded procedures for association rule mining generally based on the *Apriori* algorithm. For instance, in the Oracle Data Mining module, Oracle develops a set of predictive applications, including association rule extraction via *Apriori*. The great advantage of these approaches is that they store information on storage devices and not on the random-access memory. Thus, for the same reasons, we proposed to transform all our filters in stored procedures in Oracle.

[1]www.oracle.com

References

[1] Gediminas Adomavicius and Alexander Tuzhilin. Discovery of actionable patterns in databases: The action hierarchy approach. *IOMS: Information Systems Working Papers*, 1997.

[2] Gediminas Adomavicius and Alexander Tuzhilin. User profiling in personalization applications through rule discovery and validation. In *Proceedings of the fifth ACM SIGKDD International Conference on Knowledge Discovery and Data mining (KDD 1999)*, pages 377–381, 1999.

[3] Gediminas Adomavicius and Alexander Tuzhilin. Expert-driven validation of rule-based user models in personalization applications. *Data Mining and Knowledge Discovery*, 5(1-2):33–58, 2001.

[4] Ramesh C. Agarwal, Charu C. Aggarwal, and V. V. V. Prasad. A tree projection algorithm for generation of frequent item sets. *Journal of Parallel and Distributed Computing, Special issue on high-performance data mining*, 61(3):350–371, 2001.

[5] Rakesh Agrawal, Tomasz Imielinski, and Arun Swami. Mining association rules between sets of items in large databases. *Proceedings of the 12th ACM SIGMOD International Conference on Management of Data*, pages 207–216, 1993.

[6] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules. *Procedings of 20th International Conference Very Large Data Bases, VLDB*, pages 487–499, 1994.

[7] Aijun An, Shakil Khan, and Xiangji Huang. Objective and subjective algorithms for grouping association rules. In *Proceedings of the Third IEEE International Conference on Data Mining (ICDM 2003)*, page 477, Washington, DC, USA, 2003. IEEE Computer Society.

[8] Sarabjot S. Anand, David A. Bell, and John G. Hughes. The role of domain knowledge in data mining. In *Proceedings of the fourth International Conference on Information and Knowledge Management*, pages 37–43. ACM, 1995.

[9] Jürgen Angele, Dieter Fensel, Dieter Landes, Susanne Neubert, and Rudi Studer. Model-based and incremental knowledge engineering: The mike approach. In *Extended Papers from the IFIP TC12 Workshop on Artificial Intelligence from the Information Processing Perspective*, pages 139–168, Amsterdam, The Netherlands, The Netherlands, 1993. North-Holland Publishing Co.

[10] Grigoris Antoniou and Frank Van Harmelen. Web ontology language: Owl. In *Handbook on Ontologies in Information Systems*, pages 67–92, 2003.

[11] Claudia Antunes. Onto4ar: a framework for mining association rules. *Workshop on Constraint-Based Mining and Learning (CMILE - ECML/PKDD 2007)*, pages 37 – 48, 2007.

[12] Claudia Antunes. An ontology-based framework for mining patterns in the presence of background knowledge. In *1st International Conference on Advanced Intelligence*, pages 163–168. Post and Telecom Press, 2008.

[13] Mafruz Zaman Ashrafi, David Taniar, and Kate Smith. Redundant association rules reduction techniques. *AI 2005: Advances in Artificial Intelligence*, pages 254–263, 2005.

[14] Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider. The description logic handbook: Theory, implementation, and applications. In *Description Logic Handbook.* Cambridge University Press, 2003.

[15] B. Baesens, S. Viaene, and J. Vanthienen. Post-processing of association rules. *Workshop on Post-Processing in Machine Learning and Data Mining: Interpretation, visualization, integration, and related topics with in Sixth ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, pages 20–23, 2000.

[16] Jose Balcazar. Objective novelty of association rules: Measuring the confidence boost. In *Conference Extraction et Gestion des Connaissances 2010*, pages 297–302, 2010.

[17] Elena Baralis and Giuseppe Psaila. Designing templates for mining association rules. *Journal of Intelligent Information Systems*, pages 7–32, 1997.

[18] Yves Bastide, Rafik Taouil, Nicolas Pasquier, Gerd Stumme, and Lotfi Lakhal. Mining frequent patterns with counting inference. *ACM SIGKDD Explorations Newsletter*, 2:66 – 75, 2000.

[19] Jr. Bayardo, J. Roberto, and Rakesh Agrawal. Mining the most interesting rules. In *KDD '99: Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 145–154, New York, NY, USA, 1999. ACM.

[20] Roberto J. Bayardo Jr., Rakesh Agrawal, and Dimitrios Gunopulos. Constraint-based rule mining in large, dense databases. *ICDE '99: Proceedings of the 15th International Conference on Data Engineering*, pages 188–197, 1999.

[21] A. Bellandi, B. Furletti, V. Grossi, and A. Romei. Ontology-driven association rule extraction: A case study. *Proceedings of the Workshop "Context & Ontologies: Representation and Reasoning*, pages 1–10, 2007.

[22] Andrea Bellandi, Barbara Furletti, Valerio Grossi, and Andrea Romei. Ontological support for association rule mining. In *Proceedings of the 26th IASTED International Conference on Artificial Intelligence and Applications*, pages 110–115. ACTA Press, 2008.

[23] Zohra Ben-Said, Fabrice Guillet, and Paul Richard. Fouille visuelle de donnees en 3d et realite virtuelle : etat de l'art. In *Proceedings of French-Speaking*

Conference on Knowledge Discovery and Management (EGC2010), pages 151–156, 2010.

[24] Tim Berners-Lee and Mark Fichetti. *Weaving the Web: The Original Design and Ultimate Destiny of the World Wide Web by its Inventor.* Harper San Francisco, 1999.

[25] Tim Berners-Lee, James Hendler, and Ora Lassila. The semantic web - a new form of web content that is meaningful to computers will unleash a revolution of new possibilities. *Scientific American*, 2001.

[26] Abraham Bernstein and Foster Provost. An intelligent assistant for the knowledge discovery process. Technical report, Center for Digital Economy Research, Leonard Stern School of Business, 2001.

[27] Garret Birkhoff. Lattice theory. In *Colloquium publications*, volume 25. American Mathematical Society, 1967.

[28] Julien Blanchard, Fabrice Guillet, and Henri Briand. A user-driven and quality-oriented visualization for mining association rules. *Proceedings of the Third IEEE International Conference on Data Mining*, pages 493–496, 2003.

[29] Julien Blanchard, Fabrice Guillet, Henri Briand, and Regis Gras. Assessing rule interestingness with a probabilistic measure of deviation from equilibrium. In *Proceedings of the 11th international symposium on Applied Stochastic Models and Data Analysis ASMDA-2005*, pages 191–200, 2005.

[30] Julien Blanchard, Fabrice Guillet, and Pascale Kuntz. *Post-Mining of Association Rules: Techniques for Effective Knowledge Extraction*, chapter Semantics-Based Classification of Rule Interestingness Measures, pages 56–79. 2009.

[31] Julien Blanchard, Bruno Pinaud, Pascale Kuntz, and Fabrice Guillet. Visual analytics: A 2d-3d visualization support for human-centered rule mining. *Computers and Graphics*, 31(3):350–360, 2007.

[32] Alex Borgida. On the relative expressiveness of description logics and predicate logics. *Artificial Intelligence*, 82(1-2):353–367, 1996.

[33] J. Bouaud, B. Bachimont, J. Charlet, and P. Zweigenbaum. Methodological principles for structuring an "ontology". *In the Workshop on Basic Ontological Issues in Knowledge Sharing, International Joint Conference on Artificial Intelligence (IJCAI'95)*, 1995.

[34] Ronald Brachman and Hector Levesque. *Knowledge Representation and Reasoning.* Morgan Kaufmann Publishers Inc., 2004.

[35] Dan Brickley and RV Guha. Rdf vocabulary description language 1.0: Rdf schema. Technical report, W3C, 2004.

[36] Sergey Brin, Rajeev Motwani, and Craig Silverstein. Beyond market baskets: Generalizing association rules to correlations. *SIGMOD Record*, 26(1):265–276, 1997.

[37] Sergey Brin, Rajeev Motwani, Jeffrey D. Ullman, and Shalom Tsur. Dynamic itemset counting and implication rules for market basket data. In *SIGMOD '97: Proceedings of the 1997 ACM SIGMOD international conference on Management of data*, pages 255–264. ACM, 1997.

[38] Sergey Brin, Rajeev Motwani, Jeffrey D. Ullman, and Shalom Tsur. Dynamic itemset counting and implication rules for market basket data. In ACM, editor, *SIGMOD '97: Proceedings of the 1997 ACM SIGMOD international conference on Management of data*, pages 255–264, 1997.

[39] Laurent Brisson. Knowledge extraction using a conceptual information system (excis). *Proceedings of Ontologies-Based Databases and Information Systems Workshop in VLDB Conference*, pages 119–134, 2006.

[40] J. Broeskstra and A. Kampman. Serql: A second generation rdf query language. In *SWAD-Europe Workshop on Semantic Web Storage and Retrieval*, pages 13–14, 2003.

[41] Ivan Bruha and A. Famili. Postprocessing in machine learning and data mining. *ACM SIGKDD Explorations Newsletter*, 2:110–114, 2000.

[42] Dario Bruzzese and Cristina Davino. Visual post-analysis of association rules. *Journal of Visual Languages & Computing*, 14(6):621–635, 2003.

[43] Doug Burdick, Manuel Calimlim, Jason Flannick, Johannes Gehrke, and Tomi Yiu. Mafia: A maximal frequent itemset algorithm. *IEEE Transactions on Knowledge and Data Engineering*, 17(11):1490–1504, 2005.

[44] C. H. Cai, A. W. C. Fu, C. H. Cheng, and W. W. Kwong. Mining association rules with weighted items. In *IDEAS '98: Proceedings of the 1998 International Symposium on Database Engineering & Applications*, page 68. IEEE Computer Society, 1998.

[45] Mario Cannataro and Carmela Comito. A data mining ontology for grid programming. In *Proceedings of the First International Workshop on Semantics in Peer-to-Peer and Grid Computing (SemPGrid2003)*, 2003.

[46] Longbing Cao. *Data Mining for Bussiness Applications*, chapter Introduction to Domain Driven Data Mining, pages 3–10. Springer-Verlag, 2009.

[47] Longbing Cao, R. Schurmann, and Chengqi Zhang. Domain-driven in-depth pattern discovery: A practical methodology. In *Proceedings of AusDM*, pages 101–114, 2005.

[48] Longbing Cao and Chengqi Zhang. Domain-driven actionable knowledge discovery in the real world. In *10th Pacific-Asia conference, PAKDD 2006*, 2006.

[49] Longbing Cao, Chengqi Zhang, Qiang Yang, David Bell, Michail Vlachos, Bahar Taneri, Eamonn Keogh, Philip S. Yu, Ning Zhong, Mafruz Zaman Ashrafi, David Taniar, Eugene Dubossarsky, and Warwick Graco. Domain-driven, actionable knowledge discovery. *IEEE Intelligent Systems*, 22(4):78–88, c3, 2007.

[50] Deborah R. Carvalho, Alex A. Freitas, and Nelson Ebecken. Evaluating the correlation between objective rule interestingness measures and real human interest. In *9th European Conference on Principles and Practice of Knowledge Discovery in Databases*, pages 453–461. Springer, 2005.

[51] Aaron Ceglar and John F. Roddick. Association mining. *ACM Computers Surveys*, vol. 38, issue 2, Article 5, 2006.

[52] Hana Cespivova, Jan Rauch, Vojtech Svatek, Martin Kejkula, and Marie Tomeckova. Roles of medical ontology in association mining crisp-dm cycle. *Knowledge Discovery and Ontologies (KDO) at ECML/PKDD*, 2004.

[53] Xiaoming Chen, Xuan Zhou, Richard B. Scherl, and James Geller. Using an interest ontology for improved support in rule mining. In *Data Warehousing and Knowledge Discovery, 5th International Conference, DaWaK 2003, Prague, Czech Republic, September 3-5,2003, Proceedings*, pages 320–329, 2003.

[54] Stefan Decker, Sergey Melnik, Frank van Harmelen, Dieter Fensel, Michel Klein, Jeen Broekstra, Michael Erdmann, and Ian Horrocks. The semantic web: the roles of xml and rdf. *Internet Computing, IEEE*, 4:63 – 73, 2000.

[55] Alexandre Delteil, Catherine Faron-Zucker, and Rose Dieng. Extension of rdfs based on the cgs formalisms. In *ICCS '01: Proceedings of the 9th International Conference on Conceptual Structures*, pages 275–289, 2001.

[56] L. Dice. Measures of the amount of ecologic association between species. *Ecology*, 26:297–302, 1945.

[57] Li Ding, Pranam Kolari, Zhongli Ding, Sasikanth Avancha, Tim Finin, and Anupam Joshi. Using ontologies in the semantic web: A survey. Technical report, UMBC, 2005.

[58] M. A. Domingues and S. A. Rezende. Using taxonomies to facilitate the analysis of the association rules. *The 2nd International Workshop on Knowledge Discovery and Ontologies, held with ECML/PKDD*, pages 59–66, 2005.

[59] Guozhu Dong and Jinyan Li. Interestingness of discovered association rules in terms of neighborhood-based unexpectedness. *PAKDD '98: Proceedings of the Second Pacific-Asia Conference on Research and Development in Knowledge Discovery and Data Mining*, pages 72–86, 1998.

[60] Beatrice Duval, Ansaf Salleb, and Christel Vrain. On the discovery of exception rules: A survey. In *Quality Measures in Data Mining*, pages 77–98. 2007.

[61] Béatrice Duval, Ansaf Salleb, and Christel Vrain. On the discovery of exception rules: A survey. In Fabrice Guillet and Howard J. Hamilton, editors, *Quality Measures in Data Mining*, volume 43 of *Studies in Computational Intelligence*, pages 77–98. Springer, 2007.

[62] Clement Faure, Sylvie Delprat, Alain Mille, and Jean-Francois Boulicaut. Utilisation des reseaux bayesiens dans le cadre de l'extraction de regles d'association. In *Proceedings of the French-speaking Conference on Knowledge Discovery and Management*, pages 569–580, 2006.

[63] Usama Fayyad, Gregory Piatetsky-Shapiro, and Padhraic Smyth. From data mining to knowledge discovery in databases. *AI Magazine*, 17:37 – 54, 1996.

[64] Usama M. Fayyad, Gregory Piatetsky-Shapiro, Padhraic Smyth, and Ramasamy Uthurusamy. *Advances in Knowledge Discovery and Data Mining.* AAAI/MIT Press, 1996.

[65] Christiane Fellbaum. *WordNet: an electronic lexical database.* MIT Press, 1998.

[66] D. Fensel, F. van Harmelen, I. Horrocks, D. L. Mcguinness, and P. F. Patel-Schneider. Oil: an ontology infrastructure for the semantic web. *Intelligent Systems, IEEE [see also IEEE Intelligent Systems and Their Applications]*, 16:38 – 45, 2001.

[67] Dieter Fensel, Ian Horrocks, Frank van Harmelen, Stefan Decker, Michael Erdmann, and Michel C. A. Klein. Oil in a nutshell. In *EKAW '00: Proceedings of the 12th European Workshop on Knowledge Acquisition, Modeling and Management*, pages 1–16, 2000.

[68] Luciano Floridi, editor. *Blackwell Guide to the Philosophy of Computing and Information.* Blackwell Publishers, Inc., 2003.

[69] Philippe Fournier-Viger. *Un Modle de Reprsentation des Connaissances trois Niveaux de Smantique pour les Systmes Tutoriels Intelligents.* PhD thesis, Facult des Sciences, Universit de Sherbrooke, Qubec, Canada, 2005.

[70] William J. Frawley, Gregory Piatetsky-Shapiro, and Christopher J. Matheus. Knowledge discovery in databases: An overview. *AI Magazine*, 13, n 3:57–70, 1992.

[71] Yongjian Fu and Jiawei Hah. Meta-rule-guided mining of association rules in relational databases. In *Proceedings of the 1st International Workshop on Integration of Knowledge Discovery with Deductive and Object-Oriented Databases (KDOOD'95)*, pages 39–46, 1995.

[72] Dragan Gamberger and Nada Lavrac. Generating actionable knowledge by expert-guided subgroup discovery. In *PKDD '02: Proceedings of the 6th European Conference on Principles of Data Mining and Knowledge Discovery*, pages 163–174. Springer-Verlag, 2002.

[73] Jean-Gabriel Ganascia. Charade: a rule system learning system. In *Proceedings of the 10th International Joint Conference on Artificial Intelligence*, pages 345–347, San Francisco, CA, USA, 1987. Morgan Kaufmann Publishers Inc.

[74] Jean-Gabriel Ganascia, Jérôme Thomas, and Philippe Laublet. Integrating models of knowledge and machine learning. In *Proceedings of the European Conference on Machine Learning*, pages 396–401, London, UK, 1993. Springer-Verlag.

[75] Fabien Gandon. Ontologies informatiques, May 2006.

[76] Fabien Gandon. *Graphes RDF et leur Manipulation pour la Gestion de Connaissances*. PhD thesis, INRIA Sophia-Antipolis, 2008.

[77] Bernhard Ganter and Rudolf Wille. *Formal Concept Analysis: Mathematical Foundations*. Springer-Verlag, 1999.

[78] Ana Cristina Bicharra Garcia, Inhauma Ferraz, and Adriana S. Vivacqua. From data to knowledge mining. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, pages 1–15, 2009.

[79] Ana Cristina Bicharra Garcia and Adriana S. Vivacqua. Does ontology help make sense of a complex world or does it create a biased interpretation? *Sensemaking Workshop in CHI 2008 Conference on Human Factors in Computing Systems*, 2008.

[80] Lars Marius Garshol. Metadata? thesauri? taxonomies? topic maps! making sense of it all. *Journal of Information Science*, 30(4):378–391, 2004.

[81] Wolfgang Gassler and Eva Zangerle. *Using Databases for Ontology Processing, Storage and Reasoning in Common and Mobile Environments*. PhD thesis, Institute of Computer Science Databases and Information Systems, Leopold-Franzens-University Innsbruck, 2007.

[82] Michael R. Genesereth and Nils J. Nilsson. *Logical foundations of artificial intelligence*. Morgan Kaufmann Publishers Inc., 1987.

[83] Liqiang Geng and Howard J. Hamilton. Interestingness measures for data mining: A survey. *ACM Computing Surveys*, 38(3), 2006.

[84] IRVING JOHN Good. The estimation of probabilities: An essay on modern bayesian methods. *American Educational Research Journal*, 1967.

[85] Gosta Grahne and Jianfei Zhu. Fast algorithms for frequent itemset mining using fp-trees. *IEEE Transactions on Knowledge and Data Engineering*, 17(10):1347–1362, 2005.

[86] Regis Gras. Limplication statistique, nouvelle mthode exploratoire des donnes. *La Pense Sauvage*, 1996.

[87] Regis Gras and Pascale Kuntz. *Statistical Implicative Analysis: Theory and Applications*, volume 127 of *Studies in Computational Intelligence*, chapter An overview of the statistical implicative analysis developement, pages 21–52. Springer, 2008.

[88] William E. Grosso, Henrik Eriksson, Ray W. Fergerson, John H. Gennari, Samson W. Tu, and Mark A. Musen. Knowledge modeling at the millennium (the design and evolution of protege-2000). *Proceedings of the Twelfth Workshop on Knowledge Acquisition, Modeling and Management (KAW99)*, 1999.

[89] Thomas R. Gruber. Toward principles for the design of ontologies used for knowledge sharing. In Nicola Guarino and Roberto Poli, editors, *Formal Ontology in Conceptual Analysis and Knowledge Representation.* Kluwer Academic Publishers, 1993.

[90] Thomas R. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5:199 – 220, 1993.

[91] Nicola Guarino. Semantic matching: Formal ontological distinctions for information organization, extraction, and integration. In *International Summer School on Information Extraction: A Multidisciplinary Approach to an Emerging Information Technology*, volume 1299, pages 139–170, London, UK, 1997. Springer-Verlag.

[92] Nicola Guarino. Formal ontology in information systems. *Proceedings of the 1st International Conference on Formal Ontology in Information Systems*, pages 3–15, 1998.

[93] Nicola Guarino and Pierdaniele Giaretta. Ontologies and knowledge bases: Towards a terminological clarification. *Towards Very Large Knowledge Bases: Knowledge Building and Knowledge Sharing*, pages 25–32, 1995.

[94] Nicola Guarino and Christopher A. Welty. *A Formal Ontology of Properties*, pages 97–112. Springer-Verlag, London, UK, 2000.

[95] F. Guillet and H. Hamilton. *Quality Measures in Data Mining.* Studies in Computational Intelligence, 2007.

[96] Volker Haarslev and Ralf Möller. Racer system description. In *Proceedings of the First International Joint Conference on Automated Reasoning*, pages 701–706, London, UK, 2001. Springer-Verlag.

[97] P. Hajek, I. Havel, and M. Chytil. he guha method of automatic hypotheses determination. *Computing*, 1:293–308, 1966.

[98] Jiawei Han and Jian Pei. Mining frequent patterns by pattern-growth: methodology and implications. *ACM SIGKDD Explorations Newsletter, Special issue on Scalable data mining algorithms*, 2000(2):14–20, 2.

[99] David J. Hand, Padhraic Smyth, and Heikki Mannila. *Principles of data mining.* MIT Press, Cambridge, MA, USA, 2001.

[100] Frederick Hayes-Roth, Donald A. Waterman, and Douglas B. Lenat. *Building expert systems.* Addison-Wesley Longman Publishing Co., Inc., 1983.

[101] Z. He, X. Xu, and S. Deng. Data mining for actionable knowledge: A survey. *ArXiv Computer Science e-prints*, 2005.

[102] J. Hendler and D. L. McGuinness. The darpa agent markup language. *IEEE Intelligent Systems*, 15:67–73, 2000.

[103] Robert J. Hilderman and Howard J. Hamilton. *Knowledge Discovery and Measures of Interest.* Kluwer Academic Publishers, Norwell, MA, USA, 2001.

[104] Matthew Horridge, Holger Knublauch, Alan Rector, Robert Stevens, and Chris Wroe. A practical guide to building owl ontologies using the protg-owl plugin and coode tools edition 1.0, august 2004.

[105] I. Horrocks, Patel P. Schneider, and F. van Harmelen. From shiq and rdf to owl: The making of a web ontology language. *Journal of Web Semantics*, 1(1):7–26, 2003.

[106] Ian Horrocks. Description logics - basics, applications, and more. Technical report, Information Management Group University of Manchester, UK.

[107] Ian Horrocks. Fact and ifact. In *In Proceedings of the International Workshop on Description Logics (DL99*, pages 133–135, 1999.

[108] Ian Horrocks and Peter F. Patel-Schneider. Reducing owl entailment to description logic satisfiability. In *Journal of Web Semantics*, pages 17–29. Springer, 2003.

[109] Peter Hoschka and Willi Klosgen. A support system for interpreting statistical data. *Knowledge Discovery in Databases*, pages 325–345, 1991.

[110] Maurice A. W. Houtsma and Arun N. Swami. Set-oriented mining for association rules in relational databases. In *Proceedings of the Eleventh International Conference on Data Engineering (ICDE1995)*, pages 25–33, 1995.

[111] Xuan-Hiep Huynh, Fabrice Guillet, Julien Blanchard, Pascale Kuntz, Henri Briand, and Régis Gras. A graph-based clustering approach to evaluate interestingness measures: A tool and a comparative study. In Fabrice Guillet and Howard J. Hamilton, editors, *Quality Measures in Data Mining*, volume 43 of *Studies in Computational Intelligence*, pages 25–50. Springer, 2007.

[112] Xuan-Hiep HUYNH, Fabrice GUILLET, and Henri BRIAND. Evaluating interestingness measures with linear correlation graph. In *Proceedings of the International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems.* Springer, 2006.

[113] Tomasz Imielinski and Heikki Mannila. A database perspective on knowledge discovery. *Communications of the ACM*, 39:58–64, 1996.

[114] Tomasz Imielinski and Aashu Virmani. Association rules... and what's next? towards second generation data mining systems. In *Proceedings of the Second East European Symposium on Advances in Databases and Information Systems (ADBIS1998)*, pages 6–25. Springer-Verlag, 1998.

[115] Tomasz Imielinski, Aashu Virmani, and Amin Abdulghani. Datamine: Application programming interface and query language for database mining. In *Proceedings of the International Conference on Knowledge Discovery and Data mining (KDD1996)*, pages 256–262, 1996.

[116] P. Jaccard. Etude comparative de la distribution florale dans une portion des alpes et du jura. *Bulletin de la Societe Vaudoise des Sciences Naturelles*, 37:547–579, 1901.

[117] Szymon Jaroszewicz and Tobias Scheffer. Fast discovery of unexpected patterns in data, relative to a bayesian network. *Knowledge Discovery and Data Mining (KDD)*, pages 118–127, 2005.

[118] Szymon Jaroszewicz and Dan A. Simovici. Interestingness of frequent itemsets using bayesian networks as background knowledge. *Knowledge Discovery and Data Mining (KDD)*, pages 178–186, 2004.

[119] Yuelong Jiang. *Finding Interesting Rules from Large Data Sets.* PhD thesis, SIMON FRASER UNIVERSITY, Fall 2006.

[120] Micheline Kamber, Jiawei Han, and Jenny Y. Chiang. Metarule-guided mining of multi-dimensional association rules using data cubes. In *Proceedings of the Interantional Conference on Knowledge Discovery and Data Mining*, 1997.

[121] Michel Klein. Xml, rdf, and relatives. *IEEE Intelligent Systems*, 16:26 – 28, 2001.

[122] Mika Klemettinen, Heikki Mannila, Pirjo Ronkainen, Hannu Toivonen, and A. Inkeri Verkamo. Finding interesting rules from large sets of discovered association rules. *International Conference on Information and Knowledge Management (CIKM)*, pages 401–407, 1994.

[123] Graham Klyne and Jeremy J. Carroll. Resource description framework (rdf): Concepts and abstract syntax. Technical report, W3C, 2004.

[124] Ioannis Kopanas. The role of domain knowledge in a large scale data mining project. *Second Hellenic Conference on Artificial Intelligence (SETN)*, 2308:288–299, 2002.

[125] Evangelos E. Kotsifakos, Gerasimos Marketos, and Yannis Theodoridis. *Data Mining with Ontologies: Implementations, Findings and Frameworks*, chapter

A Framework For Integrating Ontologies And Pattern-Bases, pages 237–255. Idea Group Reference, 2007.

[126] Ora Lassila and Deborah McGuinness. The role of frame-based representation on the semantic web. Technical report, Knowledge Systems Laboratory, Stanford University and Software Technology Laboratory, Nokia Research Center, 2001.

[127] Nada Lavrac, Peter A. Flach, and Blaz Zupan. Rule evaluation measures: A unifying view. In *ILP '99: Proceedings of the 9th International Workshop on Inductive Logic Programming*, pages 174–185, 1999.

[128] Philippe Lenca, Patrick Meyer, Benoit Vaillant, and Stephane Lallich. A multicriteria decision aid for interestingness measure selection. Technical report, GET ENST Bretagne, CNRS TAMCIC, France, May 2004.

[129] I.-C. Lerman. *Classification et analyse ordinale des donnees.* Dunod, 1981.

[130] Jiuyong Li. On optimal rule discovery. *IEEE Transactions on Knowledge and Data Engineering*, 18, 2006.

[131] Bing Liu and Wynne Hsu. Post-analysis of learned rules. *National Conference on Artificial Intelligence (AAAI)*, 1:828–834, 1996.

[132] Bing Liu, Wynne Hsu, and Shu Chen. Using general impressions to analyze discovered classification rules. *Knowledge Discovery and Data Mining (KDD)*, pages 31–36, 1997.

[133] Bing Liu, Wynne Hsu, Shu Chen, and Yiming Ma. Analyzing the subjective interestingness of association rules. *IEEE Intelligent Systems*, 15:47–55, 2000.

[134] Bing Liu, Wynne Hsu, and Yiming Ma. Pruning and summarizing the discovered associations. In *KDD '99: Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 125–134. ACM, 1999.

[135] Bing Liu, Wynne Hsu, Lai-Fun Mun, and Hing-Yan Lee. Finding interesting patterns using user expectations. *IEEE Transactions on Knowledge and Data Engineering*, pages 817–832, 1999.

[136] Bing Liu, Wynne Hsu, Ke Wang, and Shu Chen. Visually aided exploration of interesting association rules. *Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*, pages 380–389, 1999.

[137] Jane Loevinger. *A systematic approach to the construction and evaluation of tests of ability*, volume 61 of *Psychological monographs.* American Psychological Assn. (Washington), 1947.

[138] Alexander Maedche and Steffen Staab. Ontology learning for the semantic web. *IEEE Intelligent Systems*, 16:72–79, 2001.

[139] Heikki Mannila, Hannu Toivonen, and A. Inkeri Verkamo. Efficient algorithms for discovering association rules. In *AAAI Workshop on Knowledge Discovery in Databases*, pages 181–192, 1994.

[140] Frank Manola and Eric Miller. Rdf primer: W3c recommendation 10 february 2004. Technical report, W3C, 2004.

[141] Christopher J. Matheus, Gregory Piatetsky-Shapiro, and Dwight McNeill. Selecting and reporting what is interesting. *Advances in knowledge discovery and data mining*, pages 495–515, 1996.

[142] John Mccarthy and Patrick J. Hayes. Some philosophical problems from the standpoint of artificial intelligence. In *Machine Intelligence*, pages 463–502. Edinburgh University Press, 1969.

[143] Ken McGarry. A survey of interestingness measures for knowledge discovery. *The Knowledge Engineering Review*, 20:39 – 61, 2005.

[144] Frederick Mosteller. Association and estimation in contingency tables. *Journal of the American Statistical Association*, 63:1–28, 1968.

[145] Enrico Motta, Tim Rajan, and Marc Eisenstadt. Knowledge acquisition as a process of model refinement. *Knowledge Acquisition*, 2:21–49, 1990.

[146] M.A. Musen, J.H. Gennari, Eriksson H., Tu S.W., and Puerta A.R. Protege-ii: computer support for development of intelligent systems from libraries of components. *Medinfo*, 8, 1995.

[147] Mark A. Musen. An overview of knowledge acquisition. *Second generation expert systems*, pages 405–427, 1993.

[148] Rajesh Natarajan and B. Shekar. A relatedness-based data-driven approach to determination of interestingness of association rules. *Proceedings of the 2005 ACM Symposium on Applied Computing (SAC)*, pages 551–552, 2005.

[149] Zohreh Nazeri and Eric Bloedorn. Exploiting available domain knowledge to improve mining aviation safety and network security data. In *Proceedings of the ECML/PKDD04 Workshop on Knowledge Discovery and Ontologies*, 2004.

[150] Raymond T. Ng, Laks V. S. Lakshmanan, Jiawei Han, and Alex Pang. Exploratory mining and pruning optimizations of constrained associations rules. pages 13–24. ACM Press, 1998.

[151] H.O. Nigro, S.E. Gonzalez Cisaro, and D.H. Xodo. *Data Mining With Ontologies: Implementations, Findings and Frameworks*. Idea Group Inc., 2007.

[152] Natalya F. Noy and Deborah L. McGuinness. Ontology development 101: A guide to creating your first ontology. Online, 2001.

[153] Edward R. Omiecinski. Alternative interest measures for mining associations in databases. *IEEE Transactions on Knowledge and Data Engineering*, 15(1):57–69, 2003.

[154] Balaji Padmanabhan and Alexander Tuzhuilin. A belief-driven method for discovering unexpected patterns. *4th International Conference on Knowledge Discovery and Data Mining*, pages 94 – 100, 1998.

[155] Balaji Padmanabhan and Alexander Tuzhuilin. Unexpectedness as a measure of interestingness in knowledge discovery. *Decision Support Systems*, 27:81–90, 1999.

[156] Balaji Padmanabhan and Alexander Tuzhuilin. Small si beautifull : Discovery the minimal set of unextected patterns. *Knowledge Discovery and Data Mining*, pages 54–63, 2000.

[157] Jong Soo Park, Ming-Syan Chen, and Philip S. Yu. Using a hash-based method with transaction trimming for mining association rules. *IEEE Transactions on Knowledge and Data Engineering*, 9(5):813–825, 1997.

[158] Bijan Parsia and Evren Sirin. Pellet: An owl dl reasoner. In *3rd International Semantic Web Conference (ISWC2004)*, 2004.

[159] Nicolas Pasquier, Yves Bastide, Rafik Taouil, and Lotfi Lakhal. Discovering frequent closed itemsets for association rules. In *ICDT '99: Proceedings of the 7th International Conference on Database Theory*, pages 398–416, London, UK, 1999. Springer-Verlag.

[160] Nicolas Pasquier, Yves Bastide, Rafik Taouil, and Lotfi Lakhal. Efficient mining of association rules using closed itemset lattices. *Information Systems*, 24:25–46, 1999.

[161] Nicolas Pasquier, Yves Bastide, Rafik Taouil, Rak Taouil, and Lotfi Lakhal. Pruning closed itemset lattices for association rules. In *In Actes Bases de Donnes Avances BDA'98, Hammamet, Tunisie*, 1998.

[162] Nicolas Pasquier, Rafik Taouil, Yves Bastide, Gerd Stumme, and Lotfi Lakhal. Generating a condensed representation for association rules. *Journal of Intelligent Information Systems*, 24(1):29–60, 2005.

[163] Jian Pei, Jiawei Han, and Runying Mao. Closet: An efficient algorithm for mining frequent closed itemsets. In *ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, pages 21–30, 2000.

[164] Joseph Phillips and Bruce G. Buchanan. Ontology-guided knowledge discovery in databases. In *Proceedings of the 1st International Conference on Knowledge Capture*, pages 123–130, 2001.

[165] G. Piatetsky-Shapiro. *Knowledge Discovery in Databases*, chapter Discovery, Analysis, and Presentation of Strong Rules, page 229248. AAAI/MIT Press, 1991.

[166] Gregory Piatetsky-Shapiro and Christopher J. Matheus. The interestingness of deviations. In *In Proceedings of the AAAI-94 Workshop on Knowledge Discovery in Databases*, pages 25–36, 1994.

[167] Woody Pidcock. What are the differences between a vocabulary, a taxonomy, a thesaurus, an ontology, and a meta-model?, January 2003.

[168] Carsten Pohle. Integrating and updating domain knowledge with data mining. *Very Large Data Bases (VLDB) Conference PhD Workshop : CEUR Workshop*, 2003.

[169] Carsten Pohle. Integrating domain knowledge for data mining post-processing. *Lernen, Wissensentdeckung und Adaptivitat : Workshop des GI-Arbeitskreises "Knowledge Discovery" (AK KD)*, pages 76–83, 2004.

[170] A. Johannes Pretorius. Ontologies - introduction and overview. 2004.

[171] Eric Prud'hommeaux and Andy Seaborne. Sparql query language for rdf. Technical report, World Wide Web Consortium, January 2008.

[172] Zbigniew W. Ras, Agnieszka Dardzinska, Li-Shiang Tsay, and Hanna Wasyluk. Association action rules. In *Proceedings of the 2008 IEEE International Conference on Data Mining Workshops*, pages 283–290. IEEE Computer Society, 2008.

[173] Zbigniew W. Ras and Alicja Wieczorkowska. Action-rules: How to increase profit of a company. In *Proceedings of the 4th European Conference on Principles of Data Mining and Knowledge Discovery*, pages 587–592. Springer, 2000.

[174] Z.W. Ras, E. Wyrzykowska, and L.-S. Tsay. *Encyclopedia of Data Warehousing and Mining - 2nd Edition*, chapter Action rules mining, pages 1–5. Idea Group Inc., 2008.

[175] Jan Rauch and Milan Simunek. *Foundations of Data Mining and knowledge Discovery*, chapter An Alternative Approach to Mining Association Rules, pages 211–231. Springer Berlin / Heidelberg, 2005.

[176] D. Rogers and T. Tanimoto. A computer program for classifing plants. *Science*, 132:1115–1118, 1960.

[177] Ashoka Savasere, Edward Omiecinski, and Shamkant B. Navathe. An efficient algorithm for mining association rules in large databases. In *Pceedings of the 21th International Conference on Very Large Data Bases*, pages 432–444, 1995.

[178] Guus Schreiber, Hans Akkermans, Anjo Anjewierden, Robert Dehoog, Nigel Shadbolt, Walter Vandevelde, and Bob Wielinga. *Knowledge Engineering and Management: The CommonKADS Methodology.* The MIT Press, December 1999.

[179] Andy Seaborne. RDQL - A Query Language for RDF. Technical report, W3C (proposal), 2004.

[180] M. Sebag and M. Schoenauer. Generation of rules with certainty and confidence factors from incomplete and incoherent learning bases. *Proceedings of European Knowledge Acquisition Workshop*, pages 28.1–28.20, 1988.

[181] Toby Segaran, Jamie Taylor, and Colin Evans. *Programming the Semantic Web.* O'Reilly, Cambridge, MA, 2009.

[182] B. Shekar and Rajesh Natarajan. A framework for evaluating knowledge-based interestingness of association rules. *Fuzzy Optimization and Decision Making*, 3(2):157–185, 2004.

[183] Abraham Silberschatz and Alexander Tuzhilin. On subjective measures of interestingness in knowledge discovery. *Knowledge Discovery and Data Mining (KDD)*, pages 275–281, 1995.

[184] Abraham Silberschatz and Alexander Tuzhilin. What makes patterns interesting in knowledge discovery systems. *IEEE Transactions on Knowledge and Data Engineering*, 8:970–974, 1996.

[185] Avi Silberschatz and Alexander Tuzhilin. User-assisted knowledge discovery: How much should the user be involved. In *SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery*, 1996.

[186] Evren Sirin and Bijan Parsia. Sparql-dl: Sparql query for owl-dl. In *OWLED*, 2007.

[187] P. Smyth and R. M. Goodman. An information theoretic approach to rule induction from databases. *IEEE Transactions on Knowledge and Data Engineering*, 4(4):301–316, 1992.

[188] John F. Sowa. *Principles of Semantic Networks: Explorations in the Representation of Knowledge.* Morgan Kaufmann Publishers, 1991.

[189] Ramakrishnan Srikant and Rakesh Agrawal. Mining generalized association rules. *Proceedings of the 21st International Conference on Very Large Databases*, (2–3):407–419, 1995.

[190] Ramakrishnan Srikant and Rakesh Agrawal. Mining quantitative association rules in large relational tables. In *Proceedings of the 1996 ACM SIGMOD international conference on Management of data*, pages 1–12, 1996.

[191] Ramakrishnan Srikant, Quoc Vu, and Rakesh Agrawal. Mining association rules with item constraints. In *Proceedings of the International Conference on Knowledge Discovery and Data mining*, pages 67–73, 1997.

[192] Steffen Staab and Alexander Maedche. Axioms are objects, too - ontology engineering beyond the modeling of concepts and relations. In *Proceedings of the Workshop on Applications of Ontologies and Problem-solving Methods, 14th European Conference on Artificial Intelligence ECAI 2000*, 2000.

[193] M. Steinbach, P.-N. Tan, H. Xiong, and V. Kumar. Objective measures for association pattern analysis. *Contemporary Mathematics*, pages 205–226, 2007.

[194] Margaret-Anne Storey, Natasha F. Noy, Mark Musen, Casey Best, Ray Fergerson, and Neil Ernst. Jambalaya: an interactive environment for exploring ontologies. *IUI '02: Proceedings of the 7th international conference on Intelligent user interfaces*, pages 239–239, 2002.

[195] Rudi Studer, V. Richard Benjamins, and Dieter Fensel. Knowledge engineering: Principles and methods. *Data & Knowledge Engineering*, 25:161 – 197, 1998.

[196] Einoshin Suzuki. Discovering unexpected exceptions: a stochastic approach. In *Proceedings of the 4th International Workshop on Rough Sets, Fuzzy Sets and Machine Discovery*, pages 225–232, 1996.

[197] Einoshin Suzuki. Autonomous discovery of reliable exception rules. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, pages 259–262, 1997.

[198] Einoshin Suzuki. Scheduled discovery of exception rules. In *DS '99: Proceedings of the Second International Conference on Discovery Science*, pages 184–195, London, UK, 1999. Springer-Verlag.

[199] Einoshin Suzuki and Yves Kodratoff. Discovery of surprising exception rules based on intensity of implication. In *PKDD '98: Proceedings of the Second European Symposium on Principles of Data Mining and Knowledge Discovery*, pages 10–18, London, UK, 1998. Springer-Verlag.

[200] Einoshin Suzuki and Masamichi Shimura. Exceptional knowledge discovery in databases based on information theory. In *KDD*, pages 275–278, 1996.

[201] Vojtech Svatek, Jan Rauch, and Martin Ralbovsky. Ontology-enhanced association mining. *Semantics, Web and Mining, Joint International Workshops, EWMF 2005 and KDO 2005*, pages 163–179, 2005.

[202] P.-N. Tan, V. Kumar, and J. Srivastava. Selecting the right objective measure for association analysis. *Information Systems*, 29:293–313, 2004.

[203] Jérôme Thomas, Philippe Laublet, and Jean-Gabriel Ganascia. A machine learning tool designed for a model-based knowledge acquisition approach.

In *Proceedings of the 7th European Workshop on Knowledge Acquisition for Knowledge-Based Systems*, pages 123–138, London, UK, 1993. Springer-Verlag.

[204] H. Toivonen, M. Klemettinen, P. Ronkainen, K. Hatonen, and H. Mannila. Pruning and grouping of discovered association rules. *ECML-95 Workshop on Statistics, Machine Learning, and Knowledge Discovery in Databases*, pages 47–52, 1995.

[205] Bill Trippe. Taxonomies & topic maps: Categorization steps forward. *EContent Magazine*, 2001.

[206] D. Tsarkov and I. Horrocks. Fact++ description logic reasoner: System description. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 4130:292–297, 2006.

[207] Angelina A. Tzacheva and Zbigniew W. Ras. Action rules mining. *International Journal of Intelligent Systems*, 20:719 – 736, 2005.

[208] G. Udny Yule. On the association of attributes in statistics. In *Philosophical Transactions of the Royal Society of London*, number 194 in A, pages 257–319. The Royal Society, 1900.

[209] Michael Uschold. Where are the semantics in the semantic web? *AI Magazine*, 24:25–36, 2003.

[210] Mike Uschold and Michael Grüninger. Ontologies: principles, methods, and applications. *Knowledge Engineering Review*, 11:93–155, 1996.

[211] Mike Uschold and Martin King. Towards a methodology for building ontologies. In *In Workshop on Basic Ontological Issues in Knowledge Sharing, held in conjunction with IJCAI-95*, 1995.

[212] C. W3. Extensible markup language (xml). Online, February 1998.

[213] Ke Wang, Yuelong Jiang, and Laks V.S. Lakshmanan. Mining unexpected rules by pushing user dynamics. *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Washington*, pages 246–255, 2003.

[214] Xiaoshu Wang, Robert Gorlitsky, and Jonas S Almeida. From xml to rdf: how semantic web technologies will change the design of 'omic' standards. *Nature Biotechnology*, 23:1099 – 1103, 2005.

[215] Rudolf Wille. Restructuring lattice theory: An approach based on hierarchies of concepts. *Ordered Sets, Ivan Rival Ed., NATO Advanced Study Institute*, 83:445–470, 1982.

[216] Dongwoo Won, Bo Mi Song, and Dennis McLeod. An approach to clustering marketing data. In *Proceedings of the 2nd International Advanced Database Conference*, 2006.

[217] Dong Xin, Xuehua Shen, Qiaozhu Mei, and Jiawei Han. Discovering interesting patterns through user's interactive feedback. *Proceedings of the International Conference on Knowledge Discovery and Data Mining (KDD 2006)*, pages 773–778, 2006.

[218] Yue Xu and Yuefeng Li. Generating concise association rules. In *Proceedings of the sixteenth ACM conference on Conference on Information and Knowledge Management*, pages 781–790, New York, NY, USA, 2007. ACM.

[219] M. J. Zaki and M. Ogihara. Theoretical foundations of association rules. In *DMKD'98 workshop on research issues in Data Mining and Knowledge Discovery*, pages 1–8. ACM Press, June 1998.

[220] Mohammed Zaki. Mining non-redundant association rules. *Data Mining and Knowledge Discovery*, 9:223–248, 2004.

[221] Mohammed J. Zaki. Generating non-redundant association rules. *International Conference on Knowledge Discovery and Data Mining*, pages 34 – 43, 2000.

[222] Mohammed J. Zaki and Ching J. Hsiao. Charm: An efficient algorithm for closed itemset mining. *In Proceedings of SIAM'02*, 2002.

[223] Qiankun Zhao and Sourav S. Bhowmick. Association rule mining: A survey. Technical report, CAIS, Nanyang Technological University, Singapore, 2003.

[224] Yanchang Zhao, Chengqi Zhang, and Shichao Zhang. Discovering interesting association rules by clustering. *AI 2004: Advances in Artificial Intelligence*, pages 23–51, 2005.

[225] Xuan Zhou and James Geller. Raising, to enhance rule mining in web marketing with the use of an ontology. *Data Mining with Ontologies: Implementations, Findings and Frameworks*, pages 18–36, 2007.