Tel-Aviv University

The School of Computer Science
Raymond and Beverly Sackler
Faculty of Exact Sciences

What Is Interesting: Studies on Interestingness in Knowledge Discovery

A dissertation submitted for the degree of
"Doctor of Philosophy"

by
Sigal Sahar

The work on this thesis was carried out under the supervision of
Prof. Yishay Mansour

Submitted to the Senate of Tel-Aviv University
March 2003

I dedicate this work with love to my parents,
Nehama and Gavriel.

Your unconditional love and support has given me the strength to be who I am. I now understand and appreciate what you have given me.

Thank you.

Acknowledgments

The author wishes to extend special thanks to her dissertation advisor, Professor Yishay Mansour, for his help, his support and his patience throughout this work. His guidance in making decisions, both big and small, and the freedom to find her own path were instrumental in getting her to where she is today.

Abstract

Knowledge Discovery in Databases (KDD) was defined by [FPSS96a] as *"[...] the non-trivial process of identifying valid, novel, potentially useful and ultimately understandable patterns in data."* As the size of databases increases, the number of patterns mined from them also increases. This number can easily increase to an extent that overwhelms users. To address this problem, patterns need to be processed for **interestingness** in order to distinguish the *"valid, novel, potentially useful and ultimately understandable patterns"* from those that are not. Interestingness has been identified as a central problem in KDD. In this work, we study this problem: the interestingness problem.

We start with an empirical evaluation of objective interestingness ranking criteria (in Chapter 3 based on [SM99]). Interestingness is ultimately subjective, and so we go on to introduce a new approach to subjective interestingness (in Chapter 4, based on [Sah99]). This approach is different from the two prior subjective interestingness approaches in that it uses very simple and very little domain knowledge—that is easy to acquire—to quickly eliminate the majority of the rules that are not interesting. We investigate how this approach can be incorporated into the mining process (in Chapter 5, based on [Sah02b]). To automate much of the tedious manual labor normally involved in interestingness exploration, we introduce a clustering framework for association rules, providing a data-driven grouping and naming scheme for the mined rules (in Chapter 6, based on [Sah02a]). The problem of interestingness is a notoriously difficult one. To reduce the size of the problem we introduce a new *type* of interestingness criteria, the Impartial Interestingness Criteria, as the first step in of the interestingness framework, the Interestingness PreProcessing Step (in Chapter 7, based on [Sah01]). The impartial interestingness criteria can be applied automatically to the output of any association mining algorithm, independently of its domain, task and users, to eliminate a significant portion of not-interesting rules.

Contents

1 Introduction **1**
- 1.1 What Is the Problem? . . . 1
  - 1.1.1 A Partial and Very Brief Historical Overview . . . 2
  - 1.1.2 The Present and the Future . . . 2
  - 1.1.3 The Problem: Too Much Data . . . 4
- 1.2 The Solution: KDD . . . 5
  - 1.2.1 KDD Process . . . 5
  - 1.2.2 Association Rules . . . 9
  - 1.2.3 Drowning In Patterns . . . 10
- 1.3 What Is Interesting . . . 11
- 1.4 Organization of Dissertation . . . 13

2 Preliminaries **15**
- 2.1 Notations and Definitions . . . 15
  - 2.1.1 Attributes . . . 15
  - 2.1.2 Itemsets, Transactions and Databases . . . 16
  - 2.1.3 Association Rules and Large Itemsets . . . 17
- 2.2 Mining Association Rules . . . 18
  - 2.2.1 The Apriori Algorithm . . . 19

3 Objective Interestingness **25**
- 3.1 Objective Interestingness Ranking Criteria . . . 26
  - 3.1.1 Ranking Criteria: Examples . . . 26
  - 3.1.2 Ranking Criteria: Making Choices . . . 28
  - 3.1.3 Ranking Criteria: Making Comparisons . . . 30
  - 3.1.4 Ranking Criteria: Comparison Results . . . 34
  - 3.1.5 Ranking Criteria: Further Analysis . . . 41
- 3.2 Tradeoff: Many Rules Vs. Fewer Rules . . . 43

3.3 Objective Interestingness: Summary and Discussion 48
3.3.1 Other Related Works 49

4 Subjective Interestingness **51**
4.1 Introduction to Subjective Interestingness 52
4.1.1 Related Approaches in the KDD Literature 52
4.1.2 Our Method: Interestingness Via What Is Not Interesting 53
4.1.3 Differentiation of the Three Subjective Interestingness Approaches: an Analogy 56
4.2 Definitions and Notations 58
4.2.1 New Definitions: Ancestor Rule, Coverage List and Family 59
4.2.2 $RSCL_{\Omega}(\rho)$ and Its Properties 59
4.3 User Classifications 60
4.3.1 The True-Not-Interesting Classification 61
4.3.2 The Not-Always-True-Interesting Classification 62
4.3.3 The Not-Always-True-Not-Interesting Classification . . 62
4.3.4 The True-Interesting or Unqualified-Interesting Classifications 63
4.4 The Algorithm: Interestingness Via What Is Not Interesting . 64
4.4.1 Selecting the Best Candidate 65
4.4.2 User Classification of Best Candidate 66
4.4.3 Action Taken According to User Classification of the Best Candidate 67
4.5 Experiments With Real Data 70
4.5.1 Results Over the WWW Proxy Logs Database 70
4.5.2 Results Over the Grocery Database 72
4.5.3 Results Over the Adult Database 73
4.6 Summary 74

5 On Incorporating Subjective Interestingness Into the Mining Process **77**
5.1 Introduction 78
5.1.1 Methods for Using Domain Knowledge and Related Works 78
5.1.2 Our Approach 79
5.2 Definitions and Notations 80
5.3 The Algorithm 81

5.3.1 Ancestor Itemsets: Motivation and Challenge 81
5.3.2 Ancestor Itemsets: Selection and Classification 83
5.3.3 The Mining Process 85
5.4 Experiments With Real Data 91
5.4.1 Reduction in the Number of Rules Mined 91
5.4.2 Resource Consumption Discussion 94
5.5 Conclusions 98

6 Exploring Interestingness Via Clustering: A Framework 101
6.1 Introduction 102
6.1.1 Clustering and Association Rules 102
6.2 Definitions and Preliminaries 104
6.3 Interestingness Exploration Through Clustering 105
6.3.1 Association Rule Profiles 105
6.3.2 Similarity Measures for Association Rules 106
6.3.3 Clustering Algorithms 115
6.4 Discussion of Empirical Analysis 117
6.4.1 Representation and Evaluation of Clustering 117
6.4.2 Outlier Analysis 119
6.4.3 Cluster Evaluation 120
6.4.4 Clustering Implementation Issues 120
6.5 Summary 121

7 Impartial Interestingness: Interestingness PreProcessing 123
7.1 Introduction 124
7.2 Interestingness: Intuitive Problem Statement 127
7.3 Interestingness Processing Techniques 132
7.3.1 Explicit Use of Domain Knowledge 132
7.3.2 Implicit Use of Domain Knowledge 132
7.4 Dependencies of Interestingness Processing Criteria and the Independent Interestingness PreProcessing 135
7.5 Impartial Interestingness Criteria 140
7.5.1 Impartial Interestingness Through Overfitting: Interestingness PreProcessing 1 140
7.5.2 Impartial Interestingness Through Transition: Interestingness PreProcessing 2 142
7.6 Empirical Evaluation With Real Data 144

7.6.1 Results of Application of the Overfitting Impartial Interestingness Criterion . . . 144
7.6.2 Results of Application of the Transition Impartial Interestingness Criterion . . . 148
7.6.3 Results of the Sequential Application of the Impartial Interestingness Criteria . . . 152
7.7 Summary . . . 152

8 Concluding Remarks and Future Work 153

A Data Description 173
A.1 Database Description . . . 173
A.1.1 Grocery Database . . . 173
A.1.2 WWW Proxy Logs Database . . . 174
A.1.3 Four Datasets from the UCI Repository . . . 174
A.2 The Implementations . . . 175

B Acronym Dictionary 177

C Subjective Interestingness: Interestingness Via What Is Not Interesting Example 179

List of Figures

1.1 The steps of the KDD process 6

2.1 Partial view of the itemset search space 19
2.2 Iteration k of the Apriori-TID algorithm 21

3.1 Comparison between different ranking criteria 36
3.2 Criteria distance measure for the Grocery Database measured with $dist_{\text{FAR}_{1.5}}(\pi_j, \pi_k)$ 38
3.3 Behavior of the ranking criteria over the WWW Proxy Logs Database, the Nursery Database and the Mushroom Database, measured using $dist_{\text{FAR}_{1.5}}$ 40
3.4 Distribution of $Pr(A)$ in the different databases 42
3.5 Distribution of the rules mined from the Grocery Database with strict mining thresholds (8%) within the rules mined with lenient ones (3%) 45
3.6 Tradeoff between mining with strict and lenient thresholds depicted for the WWW Proxy Logs Database and the Nursery Database 47

4.1 Overview: the Interestingness Via What Is Not Interesting Algorithm 64
4.2 Results of running the Interestingness Via What Is Not Interesting algorithm on the WWW Proxy Logs Database 71
4.3 Results of running the Interestingness Via What Is Not Interesting algorithm on the Grocery Database 72
4.4 Results of running the Interestingness Via What Is Not Interesting algorithm on the Adult Database 73

5.1 Part I of algorithm for incorporating subjective interestingness into the mining process 85
5.2 Reduction in the number of mined rules for the WWW Proxy Logs Database, the Grocery Database and the Adult Database 92
5.3 Adult Database Apriori-TID results with 15% thresholds . . . 95
5.4 Cumulative iteration runtimes, in seconds, for the Adult Database, the Grocery Database and the WWW Proxy Logs Database 97

6.1 The average intra-cluster distance for the Adult Database measured using d_{sc}, d_{sd} and d_{iset} 114

7.1 Framework for Determining interestingness 139
7.2 Results of the application of the overfitting impartial interestingness criterion to the WWW Proxy Logs Database, Mushroom Database and the Nursery Database 145
7.3 Results of the application of the overfitting impartial interestingness criterion Interestingness to the Adult Database, Chess Database and the Grocery Database 146
7.4 Results of the application of the transition impartial interestingness criterion 149
7.5 Results of the sequential application of the impartial interestingness criteria to the WWW Proxy Logs Database, the Mushroom Database and the Nursery Database 150
7.6 Results of the sequential application of the impartial interestingness criteria to the PreProcessing techniques on the Adult Database, the Chess Database and the Grocery Database . . . 151

C.1 Representation of the first user classification in the Interestingness Via What Is Not Interesting algorithm on the Grocery Database 180
C.2 Representation of results of first user classification in the Interestingness Via What Is Not Interesting algorithm on the Grocery Database 181

List of Tables

2.1 Illustrative database $\mathcal{DB}_{example}$ 23

7.1 Dependency of the Interestingness Methods on User Preferences 138
7.2 Average percent of rules deleted by the application of the overfitting impartial interestingness criterion 148

B.1 Common Storage Sizes 177

Chapter 1

Introduction

Let me explain. No, there is too much. Let me sum up.

—The Princess Bride (Inigo Montoya)

The challenge of extracting knowledge from data has existed from the dawn of humankind. In recent history an inherent characteristic of this challenge has changed. This change has lead to the emergence of a new area of research whose goal is to extract knowledge from masses of data: Knowledge Discovery in Databases (KDD). We therefore start in Section 1.1 with a concise look back to explain the new problems and unique challenges the field of KDD is undertaking. In Section 1.2 we briefly review the KDD process and association rules. Within the KDD solution another problem arises: how to determine which of the myriad of mined patterns are interesting. This problem, described in Section 1.3, is often referred to as the **interestingness problem** and is the problem we study in this work. In this introduction, we keep most definitions informal and provide the formalities in Chapter 2.

1.1 What Is the Problem?

Though science can cause problems, it is not by ignorance that we will solve them.

—Isaac Asimov

In this section we describe the specific knowledge discovery challenge that the field of KDD tackles.

1.1.1 A Partial and Very Brief Historical Overview

History teaches us that men and nations behave wisely once they have exhausted all other alternatives.

—Abba Eban

Long before writing systems were invented, knowledge, or lore, was passed down through generations orally. The lore was committed to memory by the elders of the community, the healers, the religious leaders, and passed to their successors (with varying degrees of success). The amount of knowledge handed down orally through the ages is impressive, yet it pales in comparison to that stored in writing.

Writing lifted the restrictions imposed by human memory on the amount of data and information, fact and fiction, that could be stored. Unfortunately, this bank of knowledge was kept away from the general populace, as literacy initially remained the province of a small minority, and "common" knowledge continued to be mostly oral. Throughout the Middle Ages in Europe Churchmen had *"virtual monopoly on learning and the art of writing [...]"* [Chu58]. Transcription of books (in Latin) was encouraged but was *"a device to preserve and amplify the treasured revolving fund of literary works [...]"* and not as *"a vehicle for new ideas [...]"* [Boo83].

Johann (Gensfleisch) Gutenberg's persistence and ingenuity in the invention of the movable type printing press changed this, and ultimately revolutionized the world. The decision of what books would be published was taken from the governing institutions and given to profit-seeking publishers who published what they believed people would read. Collections of books were no longer the province of the privileged few (wealthy nobility and clergy), but available to the literate public in libraries. Printing was referred to as *"ars aritum omnium consevatrix: the art of preservative arts"* [Ste66]. Over the years, as books became more common, literacy increased in the different local languages and published translations of works from around the world became common, opening new frontiers in philosophy, history, science, art and literature [Boo83]. This progression is thoroughly discussed in [Boo83].

1.1.2 The Present and the Future

The empires of the future are the empires of the mind.

—Winston Churchill

The amount of data, information and knowledge the human race has been collecting is, and has been, increasing dramatically. This increase was

particularly pronounced in the last few centuries. In his address to the 150th Anniversary of the American Association for the Advancement of Science (AAAS) Meeting in 1998, United States President Bill Clinton said that *"today, the store of human knowledge doubles every five years"* [Cli98]. Two years earlier, in 1996, United Kingdom Prime Minister Tony Blair wrote [Bla96] about technology and education *"In a world where human knowledge doubles every nine months [. . .]"*.

There is no consensus as to how the corpus of human knowledge should be measured. *"In the absence of any objective measure of the human knowledge"*, [Han97] takes *"the number of learned books in existence as an indicator of the volume of man's knowledge. The Library of the University of Cambridge is a copyright library in Great Britain, which means that every book published by any publisher with an office in the UK must be deposited there. Consequently the number of books kept in the Library [. . .] might be taken to be a reasonable indication of the growth of human knowledge over the last 500 years."* [Han97] concludes that *"with the exception of the 16th century when all libraries in England were affected by the Reformation, the number of books followed fairly closely an exponential growth pattern with the number of books doubling every 33 years."*

[dSP75] suggests several different methods of evaluating what de Solla Price calls the *"'size of science',"* and reflects on the difficulties in making such measurements. Using the number of founded scientific journals published as one such measurement, [dSP75] shows that the growth of "knowledge" from 1750 to 1975 was exponential (a factor of 10 every half century), so that *"[. . .] it became evident by about 1830 that the process* [of proliferation of scientific journals] *had reached a point of absurdity: no scientist could read all the journals or keep sufficiently conversant with all published work that might be relevant to his interest."* [dSP75] also shows that the growth of abstract journals—that were intended to be a solution to this exponential growth—is also exponential, and at the same rate as that of scientific journals.

The amount of information that is now available in different databases is tremendous. In the summer of 1999 [Pou99] cited the amount of data they had as 60 terabytes on disk and 1.9 petabytes (see Table B, Appendix B) on tape, and eloquently compared it to the distance of the earth from the sun (which, for reference, is "only" 150 million kilometers). [Pou99] also mentioned that more than 80% of the total assets as a percentage of the total market capitalization of twelve corporations, including Shell, BP Amoco, Smith Kline Beecham, Marks and Spencer, BT and British Gas, was intel-

lectual capital, rather than tangible assets. [Les97] calculated in 1997 that large databases contain petabytes of data.

Regardless of the precise size of existing databases, there is little doubt that massive databases are becoming more prevalent, with both the number of features and the number of transactions increasing significantly [FPSS96b]. These databases facilitate the storage of significantly larger amounts of data than was previously possible.

1.1.3 The Problem: Too Much Data

We are drowning in information and starved for knowledge.

—John Naisbitt

We have access to an overwhelming amount of information, with more and more becoming available to us every day, so much that [Les97]'s prediction from 1997 that:

> [...] in only a few years,
>
> 1. we will be able save everything—no information will have to be thrown out, and
> 2. the typical piece of information will never be looked at by a human being,

is becoming a reality. This changes the problem of knowledge discovery; historically the knowledge discovery process was hampered by lack of data and information from which knowledge could be extracted. The datasets available for investigation were relatively small. These datasets could be studied in their entirety and could be processed manually.

This has changed. *"Our capabilities for collecting and storing data of all kinds have far outpaced our abilities to analyze, summarize, and extract 'knowledge' from this data"* [FPSSU96]. We are now inundated by data: there is too much of it for us to process. The problem becomes how to discover within these large masses of data the knowledge that is useful to us. The task of extracting knowledge from elephantine masses of data presents challenges that do not exist when discovering knowledge from smaller datasets. These new challenges have only been recently acknowledged. KDD, Knowledge Discovery in Databases, addresses them.

1.2 The Solution: KDD

There is only one good, that is knowledge; there is only one evil, that is ignorance.

—Socrates

[FPSS96a] provide the most commonly used definition of KDD:

Definition 1 ***Knowledge Discovery in Databases (KDD)*** *is the non-trivial process of identifying valid, novel, potentially useful and ultimately understandable patterns in data.*

1.2.1 KDD Process

All of the books in the world contain no more information than is broadcast as video in a single large American city in a single year. Not all bits have equal value.

—Carl Sagan

The KDD process is summarized in Figure 1.1. The process is comprised of five primary, iterative steps [FPSS96c, GPSP00]. Note that after each step looping back to a previous step (to gather more information or to update the results using new information) may be necessary or advantageous. For example, after preprocessing the collected data, we may conclude that more data is needed and, thus, go back to the "data collection" step. Below we briefly review each of the steps.

KDD Process: Goal/Problem Statement

To measure the success of the KDD process, or of any other process for that matter, measures of success must be identified and agreed upon. To define measures of success, the goal of the KDD process must be stated and understood—*"the data exploration process starts by identifying the right problems to solve"* [Pyl99]. This goal statement drives the entire process: it determines what data will be collected, the preprocessing the collected data will undergo, what type of mining algorithm will be implemented, which of the mined patterns will be selected as interesting, and how the results will be presented.

Note that the goal of the KDD process is not necessarily fixed in time. In fact, it is expected that this goal will be updated over time as the goals of

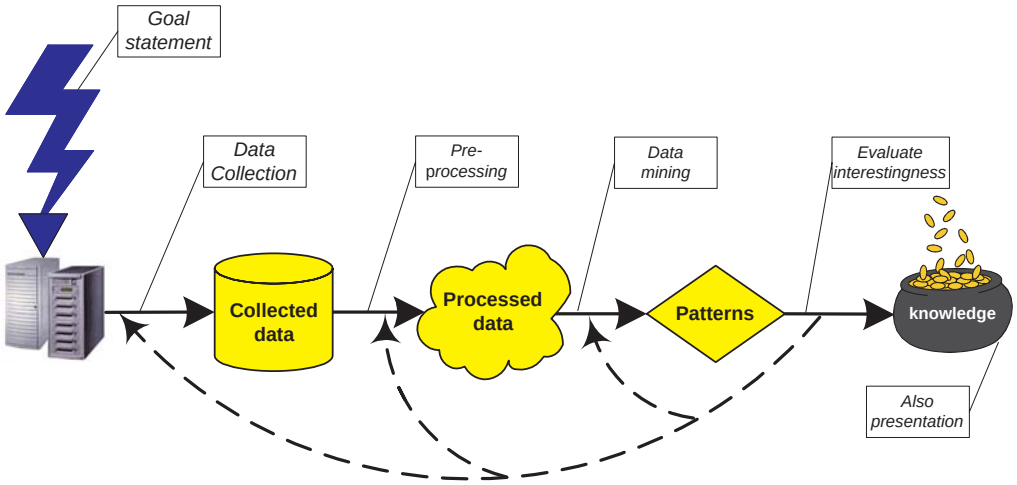

Figure 1.1: **The steps of the KDD process**.

the organization, the people and the projects change. When that happens, the entire, or parts of the KDD process may need to be repeated.

For the KDD process to be successful it is necessary to have a process-sponsor: someone who has a vested interest in the process being successful and in getting its results [GPSP00]. This sponsor ratifies the definition of the KDD process goal statement and its success criteria, and makes the needed resources available. One of the necessary resources is, of course, the data. Another is the relevant domain knowledge.

KDD Process: Data Collection

Once the goal of the KDD process is settled, it is necessary to identify what data is required and then to assemble it. Note that the data used for mining is frequently collected for other purposes. The required data may be stored in several different data warehouses of varying formats. For example, if the goal of the KDD process is to find purchase patterns among different demographic groups, the following data may be needed (1) Shopping basket information: the list of items that comprise each of the shopping baskets in the store over a predefined period of time, (2) Prices of the items in the store, and, (3) Demographic data on the customers. In most cases this

information is stored in three separate databases. Sometimes the data is not even collected. In those cases, the actual collection of the data, as well as the design, construction and maintenance of the databases may be required.

For the KDD process to be successful it is essential to have access to "right" data. We characterize the "right" using the following four characteristics:

- Relevance
- Quality
- Detail, and
- Quantity, that is, sufficiency of data.

In order to ready the collected data for the mining process, a data preprocessing step is required.

KDD Process: Data Preprocessing

The preprocessing step receives as input the data collected in the previous step and outputs the data in a format that can be used by the mining method selected for the KDD process. This preprocessing often includes the integration of the data into a single dataset, as well as dealing with noisy and missing data, reducing dimensionality when needed, etc. [HK01a, GPSP00, FPSS96c]. Note that this step combines the preprocessing and transformation steps of [FPSS96c].

Many challenges originate in the integration of the data. For example, redundant data needs to be reconciled. If the integrated datasets contain the same attribute and show the same values for all entries, using one copy of the data makes for an easy reconciliation. However, there are often inconsistencies that need to be resolved. For example, the age group of a patient in one database may be different from that listed in another database, possibly because the records do not represent the same person. It is also not always straightforward to recognize the redundancies. Attributes may have different names and may be measured using different units, for example, a 30 minute "duration of consultation" versus 1/2 hour "length of consultation." Different formats may add to the complexity of the problem, for example, March 2nd may be written as 2/3 or as 3/2, depending on the local standards. Different attributes may also be directly derived from each other, for example,

"date of birth" and "age." Discretization and approximation of attributes is sometimes needed as well. Depending on the current goal of the KDD process, feature selection and dimensionality reduction may also be required. Some data mining algorithm need a uniquely identifying number for each record a transaction identifier (TID).

Preprocessing the data has other aspects, such as dealing with noisy data. The noise can be introduced due to many reasons, for example, human errors in entering the data, and faulty data collection agents (for example, a faulty barcode scanner). Some of the data may also be missing.

The first two steps of the KDD process, collecting and preprocessing the data are estimated to take the majority of the time consumed by the KDD process [Koh01, NSKF00, AHKV98].

KDD Process: Data Mining

The preprocessed data is the input for the mining step. This step includes *choosing* the mining algorithm to be used and then applying it. This choice is heavily influenced by the goal of the KDD process: nugget discovery, classification, clustering, etc., surveyed in [FPSS96a]. The KDD community concentrates its research efforts and dedicates much of its attention to the task of mining patterns from large databases. Throughout this dissertation, we will concentrate exclusively on one type of patterns: association rules defined in Section 1.2.2. In Section 1.2.3 we list various approaches to mining association rules.

Note that although data mining is only one part of the entire KDD process, the term has become very popular and is often used to refer to the entire KDD process.

KDD Process: Evaluate Interestingness

The output of the mining process is a list of patterns. As the size of the preprocessed data increases, the number of rules mined from it also increases. This number can easily increase to an extent that will overwhelm users. And just like drowning users in masses of data is of little use, overwhelming them with a huge number of rules is also counter-productive. This is especially true since most of these rules are not even interesting to users. Out of this multitude of rules, users only want to be presented with the few rules they find interesting.

The goal of this step is to select the few patterns that *are* interesting out of the exhaustive list of patterns outputted from the data mining algorithm. As we will see in Section 1.3, determining what is interesting a challenging and intriguing problem, and a central problem in KDD. The difficulty stems from the same source that makes defining "interestingness" so complicated.

KDD Process: Knowledge and Its Presentation

The patterns that have been determined to be interesting are the output of the KDD process. These patterns need to be presented to the users of the KDD process in an understandable fashion. The presentation can be textual (for example, "from 5pm to 7pm, on weekdays, people who buy diapers are likely to also purchase beer"), graphical, or any other mutually agreeable form. Note that to be useful, the knowledge must be acted upon. This, however, is outside the scope of the KDD process.

1.2.2 Association Rules

The golden rule is that there are no golden rules.

—George Bernard Shaw

An extremely popular type of mined patterns is association rules. An association rule is a rule of the form "if SOMETHING then SOMETHING ELSE". This type of rule is also called a production rule in expert systems, an if-statement in programming, and an implication in logic. The parameters associated with association rules, support and confidence (defined in Section 2.1.3), are perhaps integral to distinguishing an association rule from these other types of if-then constructs.

For convenience, the notation in Equation 1.1 is often used to depict association rules.

$$\langle \texttt{SOMETHING} \rangle \rightarrow \langle \texttt{SOMETHING ELSE} \rangle \tag{1.1}$$

This form of patterns is readily comprehended by users of the KDD process in many different domains (this is not the case with many other representations, for example, with neural networks). That makes taking actions that rely on information provided by association rules easier and more likely.

Association rules were introduced in [AIS93]. There are many practical applications for association rules, which is one part of their considerable

appeal. Such applications include market basket analysis, which was the first application of association rules [AIS93, AHS+96, Toi96, SON95, PCY95, KMR+94], telecommunication network alarm correlation [Kle99], detecting redundant medical tests [KMS97], intrusion detection [LSM98], fraud detection [DB98], sport statistics [RP01], personalization and recommendations on the web [MDLN01], etc.

An example of an association rule is the well-known "between the hours of 5:00PM and 7:00PM, on weekdays, people that buy diapers are extremely likely to also buy beer":

$$\langle \texttt{(5–7pm)} \wedge \texttt{(Monday–Friday)} \wedge \texttt{diapers} \rangle \rightarrow \langle \texttt{beer} \rangle.$$

An association rule is accompanied by two parameters: the support and the confidence of that rule. The support of an association rule refers to the percent of transactions, or entries, in the database on which the association rule holds true. The confidence of an association rule describes the predictability of the rule. In Example 1.2.1 we describe a real association rule mined from the Grocery Database (Section A.1.1). We provide the formal definition of association rules in Definition 2 in Chapter 2, along with other relevant definitions and notations.

Example 1.2.1 *When mining the Grocery Database (Section A.1.1), we mined the rule* $\langle \texttt{cucumbers} \rangle \rightarrow \langle \texttt{tomatoes} \rangle$ *with support 41.6% and confidence 85.5%. The 41.6% support of the rule indicates that 41.6% of the customers of that grocery store purchased both cucumbers and tomatoes. The 85.5% confidence level of the rule indicates that over 85% of the customers who purchased cucumbers, will also purchase tomatoes in the same shopping basket.*

As we will see in Section 2.2, mining all the association rules that satisfy predetermined support and confidence threshold is a simple and elegant process, adding further to the association rules' already significant appeal.

1.2.3 Drowning In Patterns

Every solution breeds new problems.

—13th Basic Principle of Murphy's Law

The KDD community has dedicated a significant amount of attention to association rule mining algorithms, and has produced a variety of diverse

algorithms to tackle the task [AHS+96, Toi96, PCY95, SON95, MTV94, BMUT97, Web00, Hid99, PH01, PH02, HKK00, HGN00]. In Section 2.2 we describe the Apriori association rule mining algorithm from [AHS+96].

Using an association rule mining algorithm, we can mine the exhaustive list of association rules from a given database. The problem is that the sheer size of this list of rules can easily overwhelm users of the KDD process; this list can easily consist of many thousands of patterns if the thresholds are set to relatively low values.

Users ran the KDD process *because* they are overwhelmed by data. Are these users better off being inundated by patterns instead of drowning in data, especially after going through a resource-expensive mining process? To be successful, the KDD process should identify out of the multitude of rules outputted by mining algorithms *"[...] valid, novel, potentially useful, and ultimately understandable patterns in data"* (Definition 1). The challenge facing builders of KDD systems is not to mine the exhaustive list of all patterns, but to extract *interesting* patterns from large masses of data. As depicted in Figure 1.1 the output of the mining algorithm, consisting of a multitude of patterns, is not the final output of the KDD process; it needs to be processed further to determine **interestingness**, that is, to distinguish the *"valid, novel, potentially useful, and ultimately understandable"* patterns from those that are not.

In this dissertation we study the challenge of determining association rule interestingness.

1.3 What Is Interesting

> *The universe is run by the complex interweaving of three elements: energy, matter, and enlightened self-interest.*
>
> —Babylon 5 (G'Kar in Survivors)

Characterizing what is interesting is a difficult problem. The very definition of the word "interest" is elusive; according to [Dic92], the relevant definition of the noun INTEREST are:

1. (a) A state of curiosity or concern about or attention to something: *an interest in sports.*
   (b) Something, such as a quality, a subject or an activity, that evokes this mental state: *counts the theater among his interests.*

2. Often interests. Regard for one's own benefit or advantage; self-interest. *It is in your best interest to cooperate. She kept her own interests in mind.* [...]

4. Involvement with or participation in something: *She has an interest in the quality of her education.* [...]

Given such a vague definition, it is difficult to formulate what qualities define "interesting". Many attempts have been made to characterize these qualities. [Klo96] lists six facets of interestingness: evidence, redundancy, usefulness, novelty, simplicity and generality. These facets capture what interestingness is, and at the same time highlight the inherent difficulty in determining interestingness; what is interesting to a user depends on that specific user's prior knowledge and existing needs, making interestingness, ultimately, a subjective problem. [PJ98] speak of those characteristics and others, and classify them into two main categories: characteristics (or metrics) that can be determined in isolation, and ones that are determined in context. Piatetsky-Shapiro was the first to address the issue of interestingness in knowledge discovery in [PS91], and Klemettinen et al. [KMR+94] were the first to introduce subjective interestingness criteria in the form of pattern templates that describe the structure of subjectively interesting rules. Since then, many different approaches to the problem have been suggested such as [PS91, KMR+94, MPSM96, AT97, BJAG99, PT98, PT00, PSM99, SVA97, SLRS99, SM99, Sah99, SR00, Sah01, Fre98, LHH00]. A thorough discussion of interestingness approaches appears in the following chapters, when we describe relevant related works.

[ST95, ST96] initiated a structured approach to the problem of determining what is interesting. They differentiate between two general classes of measures of interestingness: objective measures and subjective measures. **Objective measures of interestingness** or **objective interestingness criteria** are measures of interest that depend only on the structure of the data and the patterns extracted from it. **Subjective measures of interestingness** or **subjective interestingness criteria** are measures that also depend on the specific needs and prior knowledge of the user. We will discuss both objective and subjective criteria in this work. Subjective interestingness will be a recurring topic since what is interesting to a user is ultimately subjective. For example, what is new and terribly exciting to one user may be old news, and therefore definitely not interesting to another user.

1.4 Organization of Dissertation

Be amusing: never tell unkind stories; above all, never tell long ones.

—Benjamin Disraeli

In this work we study the problem of determining which patterns are interesting. Interestingness has been posed, and is widely recognized as a critical and central component of KDD as well as a very difficult problem [KMR+94, FPSS96a, ST96, LH96, LHC97, AT97, DL98, SM99, Sah99, HH00, CTS00, TK00, LHH00, PT00, LMY01, RP01].

In Chapter 2 we formalize the terms, notations and definitions we use throughout the dissertation, and outline the Apriori [AHS+96] association rule mining algorithm.

In Chapter 3 we empirically evaluate objective interestingness ranking criteria. We examine two aspects of the behavior of ranking criteria over several real databases. The work in Chapter 3 is based on [SM99].

In Chapter 4 we introduce a new approach to subjective interestingness, the Interestingness Via What Is Not Interesting approach. This approach is differentiated from the other two subjective interestingness approaches in that it uses only a few, simple classification questions to very quickly eliminate the majority of rules that are not-interesting. The work in Chapter 4 is based on [Sah99].

The Interestingness Via What Is Not Interesting approach, presented in Chapter 4, is a post-processing approach. We show how this approach can be incorporated into the mining process in Chapter 5, and discuss the advantages and disadvantages of doing so. The work in Chapter 5 is based on [Sah02b].

In Chapter 6 we present a framework for exploring interestingness through clustering, to automate much of the laborious manual effort normally involved in this task. The work in Chapter 6 is based on [Sah02a].

In Chapter 7 we introduce a new type of interestingness criteria, the Impartial Interestingness Criteria. The impartial interestingness criteria are domain-, task- and user-independent, and form the Interestingness PreProcessing Step, complementing the subjective and objective interestingness criteria that constitute the Interestingness Processing Step. The work in Chapter 7 is based on [Sah01].

We end with a summary and some concluding remarks in Chapter 8.

Chapter 2

Preliminaries

The beginning of wisdom is the definition of terms.

—Socrates

In this chapter we present the background and formalities that we use throughout our discussion. We introduce notations and definitions and review the Agrawal et al.'s Apriori association rule mining algorithm [AHS+96].

2.1 Notations and Definitions

A beginning is a very delicate time.

—Frank Herbert in Dune (Princess Irulan)

2.1.1 Attributes

Let Λ be a set of attributes over the boolean domain. $a \in \Lambda$ can have two possible values: 1 (denoted as TRUE) and 0 (denoted as FALSE). For nonboolean domains, see [BH03].

An **attribute set** or **set of attributes** is a set, A, such that $A \subseteq \Lambda$. We use lower case letters to denote attributes and upper case letters to denote sets of attributes. We will use the notation $\|A\|$ to denote the size of an attribute set A, that is, the number of attributes in A.

We will use the notation $\oplus$ to denote the `xor` operation, that is, $A \oplus B = (A \setminus B) \cup (B \setminus A)$.

2.1.2 Itemsets, Transactions and Databases

Let $I = \{i_1, \ldots, i_m\}$ be the set of attributes in the database. For example, if the database is the set of shopping baskets of a grocery store, I may include:

$i_1 =$ `diapers purchased in shopping basket`,

$i_2 =$ `beer purchased in shopping basket`,

$i_3 =$ `tomatoes purchased in shopping basket`,

$i_4 =$ `basket purchased on weekend`,

$i_5 =$ `basket purchased between 5pm and 7pm`,

etc. i_1–i_5 are examples of boolean attributes. An example of a non-boolean attribute is $i_{not\text{-}boolean} =$ `age of customer`.

An **itemset** is a set of attributes. For example, $I' = \{i_5, i_8, i_{329}, i_{397}, i_{826}\}$. The **size of an itemset** is the number of attributes in the itemset. For example, the size of the itemset I', above, is 5. The notation **ℓ-itemsets** refers to an itemset of size ℓ. For example, I' above is a 5-itemset.

A **transaction**, τ, is a subset of attributes of Λ that have the boolean value of TRUE. We denote with $\tau(c) =$ TRUE that attribute c has the value TRUE in τ.

A transaction, τ, is said to **contain** an itemset, I, if and only if $I \subseteq \tau$. Using a different notation, we can express that τ contains I as:

$$\forall a \in I, \quad \tau(a) = \text{TRUE}, \tag{2.1}$$

We denote that transaction τ contains the itemset I with $\tau(I) =$ TRUE. If τ contains I, then the itemset I is also said to be **supported** by the transaction τ.

We will refer to the set of transactions over Λ as the **database** and denote it by $\mathcal{DB}$. The database is a **boolean database** if all its attributes are boolean. Note that if a database can be recorded in a table, the rows of the table represent the transactions and the columns detail the attributes as in Table 2.1.

If exactly $s\%$ of the transactions in the database contain I then we say that I has **support** s and denote it as **support**$(I) = s$. For example, in $\mathcal{DB}_{example}$, presented in Table 2.1 in Section 2.2.1, $\tau_1(\{a\}) =$ TRUE, $\tau_2(\{a\}) =$ TRUE, $\tau_3(\{a\}) =$ FALSE, $\tau_4(\{a\}) =$ TRUE and $\tau_5(\{a\}) =$ TRUE. The first,

second, fourth and fifth transactions support $\{a\}$ signifying that $\{a\}$ has support 4/5: $support(a) = 4/5$.

An **s-large itemset** is an itemset, I, that is supported by *at least* $s\%$ of the transactions in the database, $\mathcal{DB}$, that is, at least $s\%$ of the transactions in the database contain I. I will then be referred to as **s-large** in $\mathcal{DB}$, or, when there is no room for confusion, simply as **large**. We will often refer to a large itemset as a **frequent** itemset, and to an s-large itemset as an **s-frequent** itemset. Note that if $I' = \{i_5, i_8, i_{329}, i_{397}, i_{826}\}$ from the example above has at least support s, then I' is an s-large or s-frequent 5-itemset.

2.1.3 Association Rules and Large Itemsets

[AIS93, AHS+96] introduced the definition of association rules which we present in Definition 2 below. Example 1.2.1 in Section 1.2.2 describes an instance of an association rule.

Definition 2 *Let A and B be sets of attributes such that $A, B \subseteq \Lambda$ and $A \cap B = \emptyset$. Let $\mathcal{DB}$ be a set of transactions over Λ. The* ***association rule*** *$A \rightarrow B$ is defined to have* ***support*** *$s\%$ and* ***confidence*** *$c\%$ if $s\%$ of the transactions contain $A \cup B$, and $c\%$ of the transactions that contain A also contain B. The support of $A \rightarrow B$ is denoted by support$(A \rightarrow B)$. The confidence of $A \rightarrow B$ is denoted by confidence$(A \rightarrow B)$.*

For convenience, in a rule $A \rightarrow B$ we refer to A as the **assumption** of the rule and to B as the **consequent** of the rule. The **support of the assumption** is the percentage of the transactions in $\mathcal{DB}$ that contain A, denoted by $support(A)$, and the **support of the consequent**, is the percentage of the transactions in $\mathcal{DB}$ that contain B: $support(B)$.

We will express the support of a rule $A \rightarrow B$ as $\mathbf{Pr}[(A \cup B)]$ or $\mathbf{Pr}[AB]$ when there is no room for confusion. Note that the term probability is used here rather loosely. In general, we use $\mathbf{Pr}[C]$ to denote the fraction of transactions in $\mathcal{DB}$ that contain C: $\|\{\tau \in \mathcal{DB} | \tau(C) = \text{TRUE}\}\| / \|\mathcal{DB}\|$. The common assumption is that the database, and hence the samples, are large enough to ensure that the frequency counts are accurately indicative of the actual probabilities. Thus,

$$\mathbf{Pr}[(A \cup B)] = \mathbf{Pr}[\tau(A \cup B) = \text{TRUE}], \tag{2.2}$$

and in general,

$$\mathbf{Pr}[X] = \mathbf{Pr}[\tau(X) = \text{TRUE}]. \tag{2.3}$$

Using this notation of probability is useful, and we can now express the confidence of $A \rightarrow B$ as $\mathbf{Pr}[B|A] = \mathbf{Pr}[(A \cup B)]/\mathbf{Pr}[A]$, where we remember that $\mathbf{Pr}[(A \cup B)]$ is defined as in Equation 2.2, and $\mathbf{Pr}[A]$ is shorthand for $\mathbf{Pr}[\tau(A) = \text{TRUE}]$ as in Equation 2.3.

2.2 Mining Association Rules

First rule in government spending: why build one when you can have two at twice the price?

—Carl Sagan in Contact (S. R. Hadden)

[AHS+96] present an elegantly simple algorithm to mine the exhaustive list of association rules that have at least a predefined support threshold s and confidence threshold c from boolean databases. [AHS+96]'s association rule mining algorithm consists of two parts:

1. Find all the s-frequent (s-large) itemsets in the database, and,
2. From the frequent (large) itemsets, generate all the association rules with support and confidence levels of at least s and c respectively.

Let L be the set of all s-large itemsets in $\mathcal{DB}$. L is generated in the first step of the algorithm. This step is called the **Apriori Algorithm**, and it comes in many flavors. The general idea was introduced in [AIS93], and it was enhanced in [AHS+96]. We describe the Apriori-TID algorithm from [AHS+96] in Section 2.2.1, and use throughout this work. Given support and confidence thresholds s and c, the Apriori-TID algorithm only requires a single pass over the database to generate the exhaustive list of association rules that have at least the support s and confidence c.

Assume for a moment that we know how to generate L. Since every itemset in L is an s-large itemset, if we generate all the association rules with at least confidence c from L, we will have mined all the association rules that have at least support s and confidence greater or equal to c. Now, $A \rightarrow B$ has at least confidence c if:

$$\mathit{confidence}(A \rightarrow B) = \mathit{support}(A \cup B)/\mathit{support}(A) \geq c. \tag{2.4}$$

Thus for each $I \in L$, we need to check for $A \subset I$ if $\mathit{support}(I)/\mathit{support}(A) \geq c$. If $\mathit{support}(I)/\mathit{support}(A) \geq c$ holds, then $A \rightarrow I \setminus A$ is an association rule

with the required support and confidence thresholds s and c, and otherwise it is not. This process can be made more efficient (since if we know that *confidence*$(A \rightarrow I \setminus A) < c$, then for any $A' \subseteq A$, *confidence*$(A' \rightarrow I \setminus A') < c$), with the implementation details in [AHS+96]. All that remains is to show how L, the set of s-large itemsets, is generated in the Apriori algorithm which we proceed to do in Section 2.2.1.

2.2.1 The Apriori Algorithm

The Apriori algorithm capitalizes on a very simple observation delineated below:

Observation 1 *Each* subset *of a large itemset is necessarily a large itemset, too.*

Observation 1 holds true since the number of transactions that contain an itemset I_1 is larger than the percentage of transactions that contain $I_2 \supseteq I_1$, or, using our probability notations: $\mathbf{Pr}[I_1] \geq \mathbf{Pr}[I_2]$ if $I_1 \subseteq I_2$. We show instances of Observation 1 in Example 2.2.1.

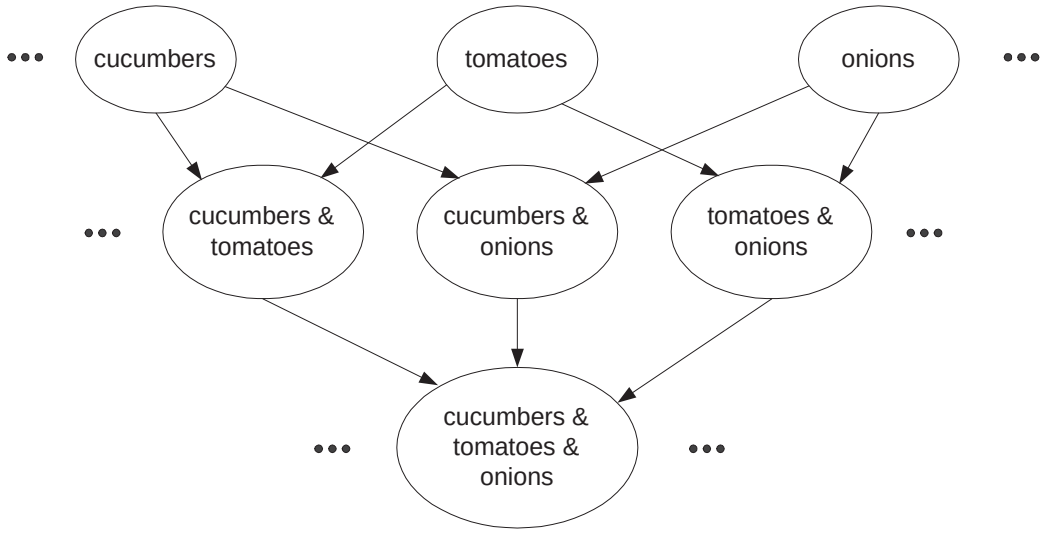

Figure 2.1: Partial view of the itemset search space

Example 2.2.1 *If* $I_{ct} = \{\texttt{cucumbers,tomatoes}\}$ *is a large itemset, then its two subsets,* $I_c = \{\texttt{cucumbers}\}$ *and* $I_t = \{\texttt{tomatoes}\}$ *are necessarily large itemsets, too, as in Figure 2.1. If* $I_{cto} = \{\texttt{cucumbers,tomatoes,onions}\}$ *is a large itemset, then, as in Figure 2.1, its three 2-itemset subsets,* I_{ct}, $I_{co} = \{\texttt{cucumbers,onions}\}$ *and* $I_{ot} = \{\texttt{onions,tomatoes}\}$ *are all large itemsets, too. We will follow this example and refer back to it throughout this work.*

Observation 1 allows us to build the set of all large itemsets of size k from the set of all large itemsets of size $k-1$ alone. [BMS97] called Observation 1 the **downward closure property**. Observation 1 is also referred to as the **anti-monotone property**, defined by [NLHP98] as the property that *"whenever a set violates the frequency constraint, so does any of its supersets"* where the frequency constraint is *"[...] that the support of an itemset must exceed some given threshold."* That is, if I_2 is not large, then $I_1 \supseteq I_2$ is *not* going to be a large itemset.

The Apriori algorithm exploits on the downward closure property by exploring only s-large $(k-1)$-itemsets to discover large k-itemsets. Let L_k be the set of all s-large k-itemsets in the database, $\mathcal{DB}$, that is, $L_k = \{I | support(I) \geq s, \|I\| = k\}$. We then have that L, the set of all s-large itemsets in $\mathcal{DB}$, is $L = \bigcup_{k=1}^{k_{max}} L_k$ where k_{max} is the maximum number of attributes in an s-large itemset over $\mathcal{DB}$. All that remains to be shown is how to generate L_k for any given $k \leq k_{max}$.

We start the Apriori-TID algorithm by constructing the set of all large itemsets of size 1, L_1. This task is completed in a single pass over the database, $\mathcal{DB}$. During this pass $\widehat{C}_1$ is also initialized to contain a representation of the database, $\mathcal{DB}$:

$$\widehat{C}_1 = \{\langle \tau_i, \{X | X = \{x\}, x \in \Lambda, \tau_i(x) = \text{TRUE}\}\rangle\}_i.$$

That is, $\widehat{C}_1$ includes, for each transaction τ_i in $\mathcal{DB}$, a list of the attributes supported by the transaction. Note that the Apriori-TID algorithm only requires this single pass over the database to construct $\widehat{C}_1$. A representation of the relevant portions of the database, $\widehat{C}_k$, is used in subsequent iterations of the algorithm to replace further passes over the database.

For each iteration $k \geq 2$, the Apriori-TID algorithm follows the steps outlined in Figure 2.2. The stop condition for the iterations is $L_k = \emptyset$. L_k is the set of all large k-itemsets: $L_k = \{I_k | support(I_k) \geq s, \|I_k\| = k\}$. L_k

is generated in `support-prune` from C_k as delineated in the Support-Prune Algorithm subsection below.

C_k is the set of all potentially large k-itemsets, generated by `apriori-gen` as detailed in the Apriori-Gen Algorithm subsection below. These potentially large k-itemsets are called **candidate large itemsets** or **candidate itemsets**. The set of candidate large k-itemsets is the set of k-itemsets whose every $(k-1)$ subset is a large $(k-1)$-itemset.

$\widehat{C}_k$ is the representation of the transactions that support each of the candidate itemsets of size k: $\widehat{C}_k = \{\langle \tau_i, \{X^i_{k,j}\}_j \rangle\}_i$, where $\{X^i_{k,j}\}_j$ is the list of the candidate s-large itemsets of size k supported by transaction τ_i. For $k = 1$, $\widehat{C}_k$ is a representation of the entire database, $\mathcal{DB}$, since every attribute in $\mathcal{DB}$ is a potentially s-large 1-itemset. We describe the steps of `candidate-support` in the Candidate-Support Algorithm subsection below.

`iteration` k: INPUT($\widehat{C}_{k-1}$, L_{k-1}); OUTPUT($\widehat{C}_k$, L_k)
(1) C_k = `apriori-gen`(L_{k-1})
(2) $\widehat{C}_k$ = `candidate-support`($\widehat{C}_{k-1}, C_k$)
(3) L_k = `support-prune`(C_k)

Figure 2.2: **Iteration k of the Apriori-TID algorithm**

Note that for consistency we use the same notations, L_k, C_k and $\widehat{C}_k$ that were used in the introduction of the Apriori-TID algorithm in [AHS+96], for the set of large k-itemsets, the set of candidate large k-itemsets, and the k-th representation of the database, respectively.

Apriori-Gen Algorithm

We start the `apriori-gen` algorithm for $k = 2$ with L_1 and $\widehat{C}_1$ initialized. For $k \geq 2$, given the list of large $(k-1)$-itemsets, L_{k-1}, the `apriori-gen` algorithm generates the list of candidate large k-itemsets, C_k. This is accomplished using Observation 1 in two steps:

1. **Join Step.** Applying Observation 1 as the downward closure property we have that the subset of every large k-itemset is a large itemset, too. That means that C_k can be constructed exclusively from L_{k-1}.

[AHS+96] does so by joining L_{k-1} with L_{k-1} into ${C_k}^{join}$ as follows:

$$ {C_k}^{join} = \{I_k = \{i_1, \ldots, i_k\} | \{i_1, \ldots, i_{k-3}, i_{k-2}, i_{k-1}\} \in L_{k-1}, \{i_1, \ldots, i_{k-3}, i_{k-2}, i_k\} \in L_{k-1}.\} \quad (2.5) $$

2. **Prune Step.** Using Observation 1, this time as the anti-monotonicity property, we have that if a subset of a k-itemset is not large, then the itemset is not large either. We use this property to prune from ${C_k}^{join}$ any itemset that has a subset that is not large, that is, we set:

$$ {C_k}^{prune} = \{I_k \in {C_k}^{join} | \exists I_{k-1} \subset I_k, I_{k-1} \notin L_{k-1}\}. \quad (2.6) $$

This step ensures that the subset of size $k-1$ of any itemset in C_k is a large $(k-1)$-itemsets, that is, in L_{k-1}.

Combining steps 1 and 2 we have $C_k = {C_k}^{join} \setminus {C_k}^{prune}$.

Example 2.2.2 *Let* $I_1 = \{$`oranges`, `tomatoes`$\}$, $I_2 = \{$`onions`,`tomatoes`$\}$ *and* $I_3 = \{$`onions`, `oranges`$\}$, *If both* I_1 *and* I_2 *are large itemsets, in the join step for* $k = 3$, $I_4 = \{$`oranges`,`tomatoes`,`onions`$\}$ *will be added into* ${C_3}^{join}$. *A necessary condition for* I_4 *to be a large 3-itemset is that its three 2-itemset subsets,* I_1, I_2 *and* I_3 *will all be large. If* $I_3 \notin L_2$, then I_{cto} *will be added to* ${C_3}^{prune}$ *and will not be included in* $C_3 = {C_3}^{join} \setminus {C_3}^{prune}$.

Candidate-Support Algorithm

In order to circumvent the need for more than one pass over the database, the support of a candidate large k-itemset is calculated using $\widehat{C}_k$. $\widehat{C}_1$ is initialized in the one and only pass over $\mathcal{DB}$ (during the initialization step). For $k \geq 2$, $\widehat{C}_k$ is constructed using $\widehat{C}_{k-1}$ and C_k, without requiring further passes over $\mathcal{DB}$. $\widehat{C}_k = \{\langle \tau_i, \{I^i_{k,j}\}_j \rangle\}_i$, where $I^i_{k,j} \in C_k$. If a transaction τ_h does not support any candidate itemset in C_k, τ_h will not be included in $\widehat{C}_k$.

Example 2.2.3 *We continue to follow Example 2.2.1 depicted in Figure 2.1. For the purpose of this illustration, we assume that* I_{ct}, I_{co} *and* I_{ot} *are all s-large, and that in the transaction* τ_{ex}, *tomatoes, cucumbers and onions were purchased. That means that* $\langle \tau_{ex}, \{\{$`tomatoes`$\}, \{$`cucumbers`$\}, \{$`onions`$\}\}\rangle \in \widehat{C}_1$. *Since according to our assumption* $I_{ct}, I_{co}, I_{ot} \in C_2$, *we have* $\langle \tau_{ex}, \{ \{$`cucumbers`,`tomatoes`$\}, \{$`cucumbers`,`onions`$\}, \{$`onions`,`tomatoes`$\}\}\rangle \in \widehat{C}_2$.

TID	$\mathcal{DB}_{example}$ attributes			
	a	b	c	d
1	TRUE	FALSE	FALSE	TRUE
2	TRUE	TRUE	TRUE	TRUE
3	FALSE	FALSE	TRUE	FALSE
4	TRUE	TRUE	FALSE	TRUE
5	TRUE	TRUE	FALSE	FALSE

Table 2.1: Illustrative database $\mathcal{DB}_{example}$

Support-Prune Algorithm

Once we have $\widehat{C}_k$ and C_k, we can construct L_k. To determine that a candidate itemset $I_k \in C_k$ is large we need to make sure that it has sufficient support, that is, $support(I_k) \geq s$. We do this by referring to the transaction list in $\widehat{C}_k$, which eliminates the need to refer back to the original database, $\mathcal{DB}$. The original Apriori algorithm used a pass over the database to check for the support of the candidate k-itemsets for every k.

Example 2.2.4 *We follow the Apriori-TID algorithm for $DB_{example}$ described in Table 2.1. Let the minimum support required for mining this database be that of two transactions, that is, $s = 2/5$.*

For $k = 1$ (initialization), $\widehat{C}_1 = \{\langle 1, \{\{a\}, \{d\}\}\rangle, \langle 2, \{\{a\}, \{b\}, \{c\}, \{d\}\}\rangle, \langle 3, \{\{c\}\}\rangle, \langle 4, \{\{a\}, \{b\}, \{d\}\}\rangle, \langle 5, \{\{a\}, \{b\}\}\rangle\}$, and $L_1 = \{\{a\}, \{b\}, \{c\}, \{d\}\}$ where the support of these itemsets is 4/5, 3/5, 2/5 and 3/5 respectively.

For $k = 2$, `apriori-gen` will output the list of candidate large 2-itemsets: $C_2 = \{\{a, b\}, \{a, c\}, \{a, d\}, \{b, c\}, \{b, d\}, \{c, d\}\}$. `candidate-support` will generate $\widehat{C}_2 = \{\langle 1, \{\{a, d\}\}\rangle, \langle 2, \{\{a, b\}, \{a, c\}, \{a, d\}, \{b, c\}, \{b, d\}, \{c, d\}\}\rangle, \langle 4, \{\{a, b\}, \{a, d\}, \{b, d\}\}\rangle, \langle 5, \{\{a, b\}\}\rangle\}$ where we note that transaction 3 is not represented in $\widehat{C}_2$ since it does not support an candidate large itemset in C_2, and `support-prune` will yield: $L_2 = \{\{a, b\}, \{a, d\}, \{b, d\}\}$ with support 3/5, 3/5 and 2/5 respectively. Note that $\{a, c\}$,$\{b, c\}$, and $\{c, d\}$ with support $1/5 < s$ were pruned by `support-prune`.

For $k = 3$, $C_3 = \{\{a, b, d\}, \{a, c, d\}\}$, with support 2/5 and 1/5 respectively, $\widehat{C}_3 = \{\langle 2, \{\{a, b, d\}, \{a, c, d\}\}\rangle, \langle 4, \{\{a, b, d\}\}\rangle\}$, and $L_3 = \{\{a, b, d\}\}$, since $\{a, c, d\}$ has support $1/5 < 2/5 = s$.

Chapter 3

Objective Interestingness*

It is a riddle wrapped in a mystery inside an enigma.

—Sir Winston Churchill, Oct. 1st, 1939

Objective interestingness criteria are measures of interest whose application relies on the structure of the data and the patterns that can be extracted from it, rather than explicitly on domain knowledge. We review this concept in depth in Chapter 7. There are three general objective interestingness approaches. The first uses a mapping of patterns to numbers indicating levels of interest. The second approach concentrates on pruning and the application of constraints, and summarization is the third approach. In this chapter we concentrate on the first approach, that of defining a mapping to determine interestingness, and discuss the other two approaches in Chapter 7. We conduct an empirical evaluation of these objective interestingness criteria using six real databases. We evaluate two aspects: first, we empirically examine several such criteria, new criteria as well as previously known ones, to discover whether the interestingness ranking performed by the various criteria is similar or easily distinguishable. The second aspect we evaluate is the dilemma of how to determine how many association rules to mine (in accordance with the selected support and confidence thresholds). The tradeoff is between using strict thresholds and therefore mining few rules, and applying low thresholds and thus mining a multitude of rules. In many cases, our empirical evaluation revealed that most of the rules found by the comparably strict thresholds ranked highly according to the interestingness criteria when using lenient thresholds (producing significantly more association rules).

*A preliminary version of this chapter was published in [SM99].

3.1 Objective Interestingness Ranking Criteria

As a general rule, the most successful man in life is the man who has the best information.

—Benjamin Disraeli

When confronted by the enormous volume of mined patterns, a natural solution is to sort the patterns by how "interesting" they are. To do that, we need to define a mapping, or a ranking criterion, f, from the set of mined patterns, Ω, to the realm of real numbers, $\mathbb{R}$, that is, $f : \Omega \rightarrow \mathbb{R}$. To generate this ranking, we assume that the criteria have access to a variety of statistical parameters, and that based on these parameters each criterion associates a numerical score with every rule.

The number a pattern is mapped to serves as an indication of how interesting the pattern is; the larger the number, the more interesting the pattern is presumed to be. The entire ranking is then achieved by sorting the rules according to the scores the rules are associated to by the mapping function, f. Alternatively, some approaches use the mapped number as a filter: any rule that is mapped to a number lower than a pre-determined threshold is marked as not-interesting. Rules that are mapped to numbers greater or equal to the predetermined threshold are pronounced as interesting.

3.1.1 Ranking Criteria: Examples

There is only one difference between a madman and me. The madman thinks he is sane. I know I am mad.

—Salvador Dalí

The only difference between me and a madman is that I'm not mad.

—Salvador Dalí

We have already come across a function that can be used as an objective interestingness ranking criterion: the confidence of an association rule. An order can be imposed on the list of mined association rules by sorting them according to their associated confidence level; the first pattern in the new order, r_1, would be the one with the highest confidence, that is, r_1 is such that:

$$\forall r \in \Omega,\ \mathit{confidence}(r_1) \geq \mathit{confidence}(r).$$

The second pattern in the confidence-interest ordered list is r_2 whose associated confidence level, $c_2 = \mathit{confidence}(r_2)$, is greater than or equal to the confidence levels of all other mined patterns with exception of r_1, i.e., $\forall r \in \Omega \backslash \{r_1\},\ c_2 = \mathit{confidence}(r_2) \geq \mathit{confidence}(r)$. And so on.

The problem with using *confidence* as an interestingness criterion is that *confidence* does not take into account the a priori probability of the consequent of a rule. The downfall of this measure is portrayed in Example 3.1.1.

Example 3.1.1 *Assume that the rule* $r_{mb} = \langle \texttt{milk} \rangle \rightarrow \langle \texttt{bread} \rangle$ *has a* 0.9 *confidence level, indicating that the 90% of the customers that buy milk also buy bread.* r_{mb} *will be ranked as highly interesting according to the confidence criterion, since it has a relatively high confidence level. However, if we are also told that* $\boldsymbol{Pr}[\texttt{bread}] = 0.9$*, that is, that 90% of the customers that shop in that store buy bread regardless of whether or not they purchased milk, our interest in the association* $\langle \texttt{milk} \rangle \rightarrow \langle \texttt{bread} \rangle$ *will decline significantly. Note that this is an illustrative example.*

Example 3.1.1 shows us that the a priori probability of the consequent should not be ignored. We resolve this issue by defining our first objective interestingness criterion, the *AddedValue* interestingness criterion in Definition 3. The *AddedValue* criterion incorporates the a priori probability of the consequent into the interestingness criterion.

Definition 3 *The* ***Added Value interestingness criterion****, denoted by* ***AddedValue*** *or* ***AV****, of an association rule* $r = A \rightarrow B$ *is defined as:*

$$\boldsymbol{AddedValue} \stackrel{\mathbf{Def}}{=} \boldsymbol{Pr}[B|A] - \boldsymbol{Pr}[B]. \tag{3.1}$$

Without any prior knowledge, we have $\mathbf{Pr}[B]$ of finding the consequent, B, in the database. The *AddedValue* of a pattern, the conditional probability of the consequent of the rule given the assumption, minus the a priori probability of the consequent, can have a positive, negative or zero value. An interesting rule should therefore have a high positive *AddedValue* value. Note that the *AddedValue* criterion has a very interesting property: a rule with a negative *AddedValue* and a large absolute value is indicative of the existence of another rule with high *AddedValue*.

The problem of ranking rules according to objective interestingness measure has been addressed in the literature as early as 1991 in [PS91]. [PS91] suggested the first three principles of interestingness evaluation criteria, as

well as a the simplest function that could satisfy them. Since then, many different ranking criteria have been proposed as measures of interestingness (lists of such criteria appear in [SM99, BJA99, HH00, HH01b, HH01a, TKS02] where [HH01a] provides one of the more comprehensive reviews of such criteria).

Ranking criteria can be "borrowed" from other areas of research, for example, mutual information from information theory. The properties of mutual information make it a natural objective interestingness ranking criterion. We can write the mutual information as $\mathit{MutualInformation}(A{\rightarrow}B) = I(B;A) = H(B) - H(B|A)$, where $H(B)$ is the entropy of B, defined as $H(B) = -\big(\mathbf{Pr}[B]\log\mathbf{Pr}[B] + \mathbf{Pr}[\overline{B}]\log\mathbf{Pr}[\overline{B}]\big)$, where $\mathbf{Pr}[\overline{X}] = 1 - \mathbf{Pr}[X]$. Intuitively, the mutual information is the information in A about B. I(B;A) is high if knowing that A occurs reduces the uncertainty of B occurring. This is precisely what we require of an interesting rule $A{\rightarrow}B$. An in-depth review of mutual information is presented in [CT91].

Other than the *AddedValue* interestingness ranking criterion, we have defined several other ranking criteria in the following section.

3.1.2 Ranking Criteria: Making Choices

> *Cleanliness and order are not matters of instinct; they are matters of education, and like most great things, you must cultivate a taste for them.*
>
> —Benjamin Disraeli

The many sources of potential objective interestingness ranking criteria lead to a wealth of possibilities according to which the interestingness order over the patterns can be set. The question that now arises is which of those different ranking criteria we should use. If the different ranking criteria impose a vastly different order on the mined patterns, the question becomes crucial: how will we know which criterion to use for the ranking? The choice of which criterion to use could make an enormous difference on the results. We therefore need to compare the criteria, see how different they are from each other and determine what course of action those differences will dictate.

All the criteria we will discuss can be implemented using the following three statistics to score an association rule, $A{\rightarrow}B$: $\mathbf{Pr}[A]$, $\mathbf{Pr}[B]$, and $\mathbf{Pr}[(A \cup B)]$. Remember our use of the probability notations defined in Equations 2.3 and 2.2, so that $\mathbf{Pr}[A]$ denotes $\mathbf{Pr}[\tau(A) = \textsc{true}]$, $\mathbf{Pr}[B]$ denotes $\mathbf{Pr}[\tau(B) = \textsc{true}]$ and $\mathbf{Pr}[(A \cup B)]$ denotes $\mathbf{Pr}[\tau(A \cup B) = \textsc{true}]$.

List of Criteria

Many interestingness criteria appear in different contexts in the literature. We list a several below.

1. **confidence**$(A \rightarrow B) = \mathbf{Pr}[B|A]$, the *confidence* level associated with an association rule (as in Definition 2).

2. **AddedValue**$(A \rightarrow B) = \mathbf{Pr}[B|A] - \mathbf{Pr}[B]$. We introduced this ranking criterion in Definition 3.

3. **MutualInformation**$(A \rightarrow B) = I(B;A) = H(B) - H(B|A)$, as detailed in [CT91].

4. **P-S**$(A \rightarrow B) = \mathbf{Pr}[(A \cup B)] - \mathbf{Pr}[B] \cdot \mathbf{Pr}[A]$. This was the first objective criteria to be suggested [PS91]. We can re-write it as $P\text{-}S(A \rightarrow B) = \mathbf{Pr}[A] \cdot AddedValue(A \rightarrow B)$.

5. **BFOS**$(A \rightarrow B) = \frac{\mathbf{Pr}[A]}{1-\mathbf{Pr}[A]} \cdot (AddedValue(A \rightarrow B))^2$, was suggested in [BFOS84]. This criterion is more readily understood if we rewrite it as:

$$\begin{aligned} BFOS(A \rightarrow B) &= \frac{\mathbf{Pr}[A]}{\mathbf{Pr}[\overline{A}]} \cdot \left(\frac{\mathbf{Pr}[(A \cup B)]}{\mathbf{Pr}[A]} - \mathbf{Pr}[B] \right) \cdot \\ &\quad \left(\mathbf{Pr}[B|A] - \mathbf{Pr}[B|A]\mathbf{Pr}[A] - \mathbf{Pr}[B|\overline{A}]\mathbf{Pr}[\overline{A}] \right) \\ &= \left(\mathbf{Pr}[(A \cup B)] - \mathbf{Pr}[A]\mathbf{Pr}[B] \right) \left(\mathbf{Pr}[B|A] - \mathbf{Pr}[B|\overline{A}] \right), \end{aligned}$$

   which can be rewritten as:

$$BFOS(A \rightarrow B) = P\text{-}S(A \rightarrow B) \cdot \left[confidence(A \rightarrow B) - confidence(\overline{A} \rightarrow B) \right].$$

6. $\mathbf{K}(A \rightarrow B) = \sqrt{\mathbf{Pr}[A]} \cdot AddedValue(A \rightarrow B)$. This criterion was suggested in [Klo92, Klo96].

In addition to the known criteria we listed above, we introduce the following two criteria:

7. $\mathbf{combined}_1(A \rightarrow B) \stackrel{\mathbf{Def}}{=} AddedValue(A \rightarrow B) \cdot I(B;A) \cdot \mathbf{Pr}[(A \cup B)]$. This criterion requires the rule to have high support, high *Mutual-Information* and high *AddedValue*.

We also propose the following variation on it is:

8. $\mathbf{combined}_2(A \rightarrow B) \stackrel{\mathbf{Def}}{=} AddedValue(A \rightarrow B) \cdot I(B; A) \cdot \mathbf{Pr}[A]$.

It is interesting to note that although the last two criteria we introduce, 7. and 8., seem very similar, differentiated as they are only by their last terms, they end up manifesting significantly different behaviors as we will see in Section 3.1.3.

3.1.3 Ranking Criteria: Making Comparisons

> *Yes, yes, Zathras is used to being beast of burden to other people's needs. Very sad life. Probably have very sad death, but at least there is symmetry.*
>
> —Babylon 5 (Zathras in War Without End Part 1)

We number the rules in the order they were mined (or in any arbitrary order), $r_1, \ldots, r_n$. Throughout Section 3.1 we will refer to the rules by their indexes.

Each objective interestingness ranking criterion, c_k, imposes an order on the n rules: $\{\pi_k(i)\}_{i=1}^n$, a permutation of $1, \ldots, n$. $\pi_k(i)$ is the interestingness rank (out of n) of the rule r_i imposed by the ranking criterion c_k, as illustrated in Example 3.1.2.

Example 3.1.2 *Let $n = 4$, that is, we have four rules in this example: r_1, r_2, r_3, r_4. Let the ranking criterion, $c_{example}$, impose the following interestingness values: $c_{example}(r_1) = 32$, $c_{example}(r_2) = 0.4$, $c_{example}(r_3) = 31$, and $c_{example}(r_4) = 4$. The permutation defined by $c_{example}$ is then $\pi = \{2, 4, 3, 1\}$ since $c_{example}(r_2) \geq c_{example}(r_4) \geq c_{example}(r_3) \geq c_{example}(r_4)$.*

We have investigated several distance measures between any two given permutations, π_j and π_k over $1, \ldots, n$, and summarize several of them below. Note that we use the term "distance measure" loosely. A distance metric, d, is defined [KF70, Lan83, Fik65] to be a real function that obeys the following four properties:

1. **Identity**: $\forall p,\ d(p, p) = 0$,
2. **Positivity**: $\forall p \neq q,\ d(p, q) > 0$,
3. **Symmetry**: $\forall p, q,\ d(p, q) = d(q, p)$, and,
4. **Triangle inequality**: $\forall p, q, r,\ d(p, q) + d(q, r) \geq d(p, r)$.

Most of the measures we used to gauge the differences between two permutations do not satisfy the triangle property. All the distance measures we use do share an important characteristic defined in Property 2.

Property 2 *Differences between the order imposed on two rules by two ranking criteria (permutations) weigh less as the discrepancy occurs on rules that are ranked as less interesting by these criteria. In other words, the "further along" in the ranking order the discrepancy happens, the less weight it receives.*

Example 3.1.3 *We examine two permutations, π_j and π_k, and two rules, r and r', such that $\pi_j(r) = 1$, $\pi_k(r) = 2,001$, $\pi_j(r') = 1,001$, and $\pi_k(r') = 3,001$. Although $\pi_k(r) - \pi_j(r) = \pi_k(r') - \pi_j(r') = 2,000$, the difference between the order imposed on r by π_j and by π_k is considered more significant, or more important, than the difference between the order imposed on r' by the two permutations, since r' is ranked as much less interesting by both permutations than r is.*

In our case, where the permutations are used to impose an "interest" order on the rules, Property 2 is an important characteristic for the desired measure of differences between ranking-permutations; the order is used to determine which patterns are more (or less) interesting, where rules with higher rankings are interpreted to be more interesting than rules with a lower rankings[†]. Therefore, discrepancies amongst the higher ranking patterns should be considered as more significant. In Example 3.1.3 this means that the different rankings of r according to π_j and π_k are considered more significant than the rankings of r' according to π_j and π_k even though in both cases the *absolute* difference in the rankings is only of 2000 ($2,001 - 1$ for r and $3,001 - 1,001$ for r'). In other words, the difference between having a rule be ranked as the most important or as the two-thousand-first most important is much more significant than the difference between having a rule ranked as the thousand-first most significant or the three-thousand-first most significant. In other words, the "further along" the permutation we are, the less significant conflicts in permutation values are. This concept is defined formally below.

[†]Note that the highest ranking possible is 1, the smallest number.

The Farness Distance Measure

Let π_j and π_k be two different permutations that map a rule r_i to two different scores, $\alpha = \pi_j(i)$ and $\beta = \pi_k(i)$, where r_i is the i-th rule. If the two scores are close enough to each other, we may disregard the difference in the scores as insignificant. On the other hand, if the two scores are "too" far apart from each other, we would like to count that as a meaningful or important difference. We now need to formalize what we mean by "too far," our notion of farness.

For two numbers, α and β, given a number $\mathcal{L} > 1$, α and β are defined to be $\mathcal{L}$-FAR if $\alpha \notin [\frac{\beta}{\mathcal{L}}, \beta \cdot \mathcal{L}]$. Now we define in Equation 3.2 an indicator, $\text{FAR}_{\mathcal{L}}(\alpha, \beta)$, that will be set to 1 if and only if α and β are more than $\mathcal{L}$ multiple apart from each other. For $\alpha = \pi_j(i)$ and $\beta = \pi_k(i)$, the $\text{FAR}_{\mathcal{L}}$ indicator is set to 1 if the values that the the permutations π_j and π_k give the rule mapped to i by the ranking criterion are $\mathcal{L}$-FAR from each other.

$$\text{FAR}_{\mathcal{L}}(\alpha, \beta) \stackrel{\textbf{Def}}{=} \begin{cases} 1 & \text{if } \alpha > \mathcal{L} \cdot \beta \text{ or } \beta > \mathcal{L} \cdot \alpha \\ 0 & \text{otherwise} \end{cases} \tag{3.2}$$

Note that the $\text{FAR}_{\mathcal{L}}$ indicator is symmetric: $\text{FAR}_{\mathcal{L}}(\alpha, \beta) = \text{FAR}_{\mathcal{L}}(\beta, \alpha)$. Additionally, we restrict the values of $\mathcal{L}$ to $\mathcal{L} > 1$. This ensures that $\text{FAR}_{\mathcal{L}}(\alpha, \alpha) = 0$, and precludes an inequality indicator.

For a permutation π, as $\pi(i)$ increases there are less values that are $\mathcal{L}$-FAR of $\pi(i)$, permitting a larger acceptable discrepancy in the orders imposed by the two permutations, naturally satisfying Property 2. Remember that the permutations are used to impose an interestingness ranking order on the rules, and rules that are ranked amongst the first ones are judged to be more interesting than latter ones. Therefore, the "further along" the permutation we are, the less significant conflicts in the interest ranking are, as in Example 3.1.3.

A logical distance measure of two permutations is the number of rules on which the permutations are $\mathcal{L}$-FAR. Normalizing this value we get the $dist_{\text{FAR}_{\mathcal{L}}}$ measure defined in Equation 3.3.

$$dist_{\text{FAR}_{\mathcal{L}}}(\pi_j, \pi_k) \stackrel{\textbf{Def}}{=} \frac{1}{n} \sum_{i=1}^{n} \text{FAR}_{\mathcal{L}}(\pi_j(i), \pi_k(i)). \tag{3.3}$$

Intuitively, $dist_{\text{FAR}_{\mathcal{L}}}$ measures the fraction of rules that are $\mathcal{L}$-FAR over two permutations π_j and π_k. Note that it is possible that for three permutations,

π_1, π_2 and π_3, the values of $dist_{\mathrm{FAR}_{\mathcal{L}}}(\pi_1, \pi_2)$ and $dist_{\mathrm{FAR}_{\mathcal{L}}}(\pi_2, \pi_3)$ will be very small (indicating that, respectively, both couples, π_1 and π_2, and π_2 and π_3, rank rules similarly), but at the same time, π_1 and π_3 will be very different, that is, $dist_{\mathrm{FAR}_{\mathcal{L}}}(\pi_1, \pi_3)$ is large. So although the $dist_{\mathrm{FAR}_{\mathcal{L}}}$ measure obeys the properties of identity, positivity and symmetry, it is not a distance metric.

Using the $dist_{\mathrm{FAR}_{\mathcal{L}}}$ distance measure, we can calculate the distance between a permutation π and all other permutations for any value of $\mathcal{L} > 1$. Note that the $dist_{\mathrm{FAR}_{\mathcal{L}}}$ distance measure[‡] has several useful characteristics. First, it is normalized, and the actual distance measure provides a useful indicator to the difference between two permutations. Furthermore, it defines a continuous distance measure[‡] in the sense that as we can increase $\mathcal{L}$ to allow for a more lenient distance measure.

Ranking Criteria: Other Distance Measures

We used other distance measures, and list representatives below:

- **The Ratio Measure**. Another way to estimate the distance between the values of $\pi_j(i)$ and $\pi_k(i)$ is to look at their ratio. In order to achieve a symmetric measure we look at both the ratio and the inverse ratio, and define $dist_{\mathrm{RATIO}}$ in Equation 3.4 below:

$$dist_{\mathrm{RATIO}}(\pi_k, \pi_j) \stackrel{\mathbf{Def}}{=} \left(\frac{1}{2n} \sum_{i=1}^{n} \left[\frac{\pi_j(i)}{\pi_k(i)} + \frac{\pi_k(i)}{\pi_j(i)} \right] \right) - 1. \tag{3.4}$$

  We subtract one from the sum of the ratio and inverse ratio to make sure that $dist_{\mathrm{RATIO}}(\pi, \pi) = 0$.

- **The Weighted Length Measure**. Given the i-th rule, r_i, and any two permutations, π_j and π_k, we define the **length of the permutations over** r_i as:

$$\ell_{\pi_j, \pi_k}(i) \stackrel{\mathbf{Def}}{=} |\pi_j(i) - \pi_k(i)|. \tag{3.5}$$

  We use ℓ_{π_j, π_k} from Equation 3.5 to define two distance measures below. Remember that we want all the permutation distance measures to obey

[‡]Remember that we use the term distance measure rather loosely in this context.

Property 2: that differences in the earlier ranking (higher rankings indicated by smaller values of the ranking criterion or permutation) contribute more significantly to the distance; dissimilarities in the ranking of two rules by two permutations that occur at very low ranking (high values of the permutation) should weigh less, as in Example 3.1.3. To support this property, we weigh the length, ℓ_{π_j,π_k}. We can choose the weights to be the actual elements: $\pi_j(i)$ and $\pi_k(i)$. This yields the $dist_{\text{LENGTH}_{\text{ELEMENT}}}$ distance measure defined as follows:

$$dist_{\text{LENGTH}_{\text{ELEMENT}}}(\pi_j, \pi_k) \stackrel{\text{Def}}{=} \frac{1}{2n} \sum_{i=1}^{n} \left[\frac{\ell_{\pi_j,\pi_k}(i)}{\pi_j(i)} + \frac{\ell_{\pi_j,\pi_k}(i)}{\pi_k(i)} \right]. \tag{3.6}$$

If we to use the minimum value of the permutation, $\min\{\pi_j(i), \pi_k(i)\}$, as the weight, we get the $dist_{\text{LENGTH}_{\text{MIN}}}$ distance measure defined in Equation 3.7:

$$dist_{\text{LENGTH}_{\text{MIN}}}(\pi_j, \pi_k) \stackrel{\text{Def}}{=} \sum_{i=1}^{n} \frac{\ell_{\pi_j,\pi_k}(i)}{\min\{\pi_j(i), \pi_k(i)\}}. \tag{3.7}$$

3.1.4 Ranking Criteria: Comparison Results

For the empirical evaluation, we used the six databases described in Appendix A. We mined the databases using the following support and confidence thresholds:

- 3.5% thresholds for the Grocery Database, yielding 3,046 associations.
- 6% thresholds for the WWW Proxy Logs Database, yielding 9,206 rules.
- 3% thresholds for the Nursery Database, yielding 8,314 associations.
- 10% thresholds for the Chess Database, yielding 8,292 associations.
- 20% thresholds for the Adult Database, yielding 13,906 associations.
- 35% thresholds for the Mushroom Database, yielding 6,356 rules.

As we will see, the very high thresholds used for some of the databases make the rules more difficult to distinguish and rank according to the objective interestingness ranking criteria. Note that following the mining process

(described in Section 2.2), the patterns undergo a filtering process prior to being ranked. We remove from the output of the data mining process all the rules that have an *AddedValue* of 5% or under (detailed explanation in Section 3.1.5).

Using each of the distance measures, we calculated the distances between each permutation and all eight permutations (which are defined by the interest ranking criteria) and derived eight distance values. Note that one of these values will always be zero, the distance from the permutation to itself, as all the distance measures we used obey the identity property: $dist(\pi_i, \pi_i) = 0$. For each permutation we plotted these points as a line on a graph where on the y-axis we have the distance measures and the x-axis denotes to which permutation we measure the distance. The permutations are numbered $1, \ldots, 8$, according to the order they were listed in Section 3.1.2. Each graph in Figure 3.1 has a legend in the upper right hand side indicating the ranking criterion that defines the permutation.

In each of the four graphs of Figure 3.1 we plotted eight lines depicting the distances from each permutation to all the permutations. Thus, each line describes the interestingness ranking behavior of a criterion with respect to all the objective ranking criteria. The pair of graphs at the top portrays the behavior of the eight criteria over the Grocery Database (the database is described in Section A.1.1). The pair of graphs at the bottom depicts the behavior of the eight criteria over the Mushroom Database(Section A.1.3). This behavior was measured using two distance measures, $dist_{\text{RATIO}}$ in the top-right and bottom-right graphs of Figure 3.1 and using $dist_{\text{LENGTH}_{element}}$ in the top left and bottom left graphs of the figure.

We noticed that regardless of the distance measure used, the ranking criteria tend to cluster in three groups:

1. The *confidence* criterion and the *AddedValue* criterion (criteria 1 and 2) are two outliers.

2. The *MutualInformation* criterion, the *P-S* criterion and the $combined_2$ criterion (criteria 3, 4 and 8) fall in one group, and finally,

3. The *BFOS*, *K* and $combined_1$ criteria (criteria 5, 6 and 7) occur in a tight cluster.

The clusters are differentiated by the behavior of the distances to each other. In other words, the distances from the members (the interestingness

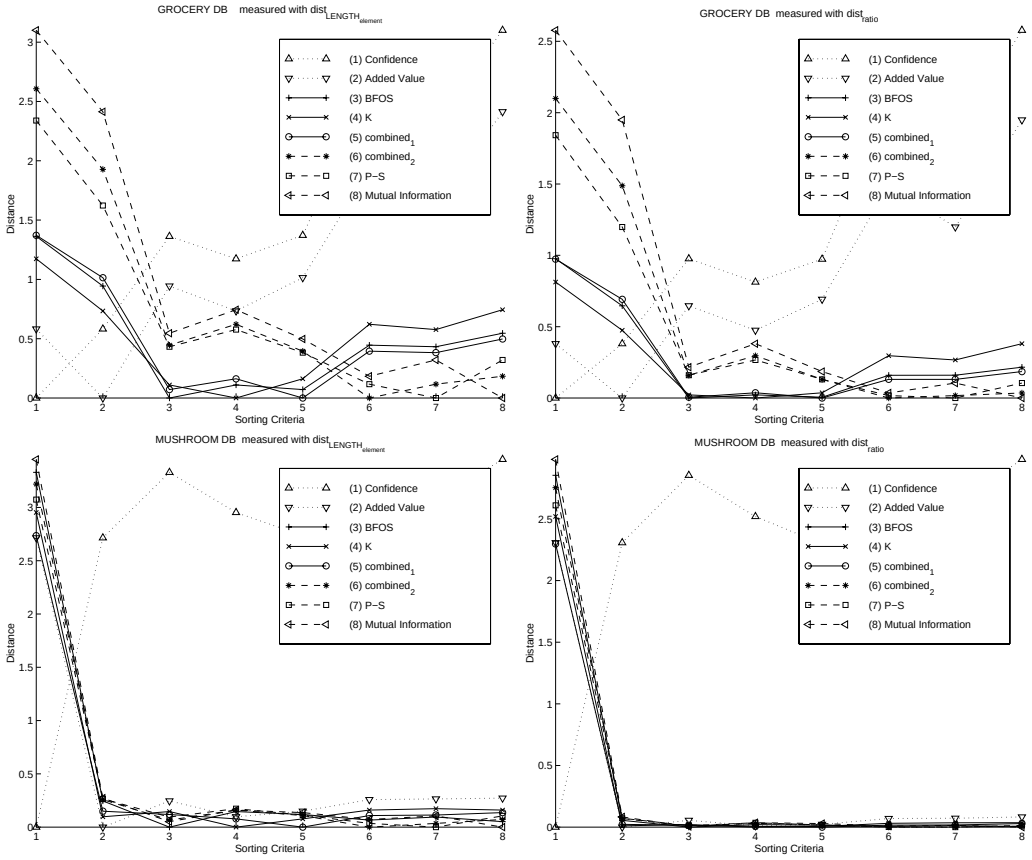

Figure 3.1: **Comparison between different ranking criteria.** *The top and bottom pairs of graphs in this figure depict the behavior of the ranking criteria over the Grocery Database and the Mushroom Database respectively. Each line in the graphs details the distance from the order imposed by one ranking criterion to all others. For example, the lines detailing the behavior of criteria 3, 4 and 5 behave in a similar manner: they are* ***all*** *relatively close to each other,* ***all*** *relatively far from criteria 1 and 2, and* ***all*** *at a medium distance to the other criteria, 6, 7 and 8. Thus, they form one cluster. The distance behavior is measured using* $\mathrm{dist}_{\mathrm{RATIO}}$ *in the top right and bottom right graphs, and using* $\mathrm{dist}_{\mathrm{LENGTH}_{\mathrm{ELEMENT}}}$ *in the top left and bottom left graphs. Although different distance measures are used, the same clusters are manifested. Note that there is no notable difference in the behavior of criteria 2–8 in the Mushroom Database.*

ranking orders imposed by ranking criteria) of one cluster to other permutations (ranking criteria) behave similarly. What is more, this behavior is invariant under the distance measure used to measure the distance; in all the different distance measures we used, the distances maintain similar behavior pattern. Note that the two members of the first cluster are outliers. While the *AddedValue* criterion often behaves similarly to the criteria in the last cluster, its distance to those criteria just as often remains significant. The confidence criterion always remains conspicuously far from criteria 3–8, and rarely resembles the behavior of the *AddedValue* criterion.

To illustrate these different behavior patterns, we present the distance measured by the $dist_{\text{FAR}_{1.5}}$[§] measure in Figure 3.2. Figure 3.2 contains two different graphs: in the top graph we have the distance behavior between all the ranking criteria. As in Figure 3.1, each line in the graph denotes the distances from a single criterion to all criteria, where on the y-axis we have the numeric measure of the distances, and on the x-axis the criterion index. Since the $dist_{\text{FAR}_{1.5}}$ measure is normalized, the scale is from zero to one. To make the clusters more discernible, in the bottom graph of Figure 3.2 we plotted each of the three clusters in a separate subgraph. As $\mathcal{L}$ increases, the distances between permutations converge to zero, as do the lines that depict these distance behaviors. The convergence rate of the first cluster, the outliers, is significantly slower than that of the other two clusters. Even for extremely large values of $\mathcal{L}$, the difference between members of the first cluster and members of the second and third cluster remains non-zero.

Since $dist_{\text{FAR}_{\mathcal{L}}}$ is normalized to output values between 0 and 1, the resulting distance measure values can be considered as an indication of the relative, or percentage, difference between the criteria being measured. Even for low values of $\mathcal{L}$, such as $\mathcal{L} = 1.5$, the differences between members of the same cluster are small. The members of the second cluster are less than 30% different from each other (with under 9% difference between $combined_2$ and other members of the cluster), and there is less than 6% difference between the members of the third cluster (the difference falls to under 0.5% between *BFOS* and $combined_1$). For $\mathcal{L} = 2$ the differences are under 8% for members of the second cluster (with approximately 2% difference between $combined_2$ and other members of the cluster), and approximately 2% for the members of the third cluster. For $\mathcal{L} = 3$ the intra-cluster differences within mem-

[§]Recall that $\mathcal{L} = 1.5$ means, for example, that not to be considered $\mathcal{L}$-FAR, a rule ranked 30 by one criterion is expected to rank between 20 and 45 by another criterion.

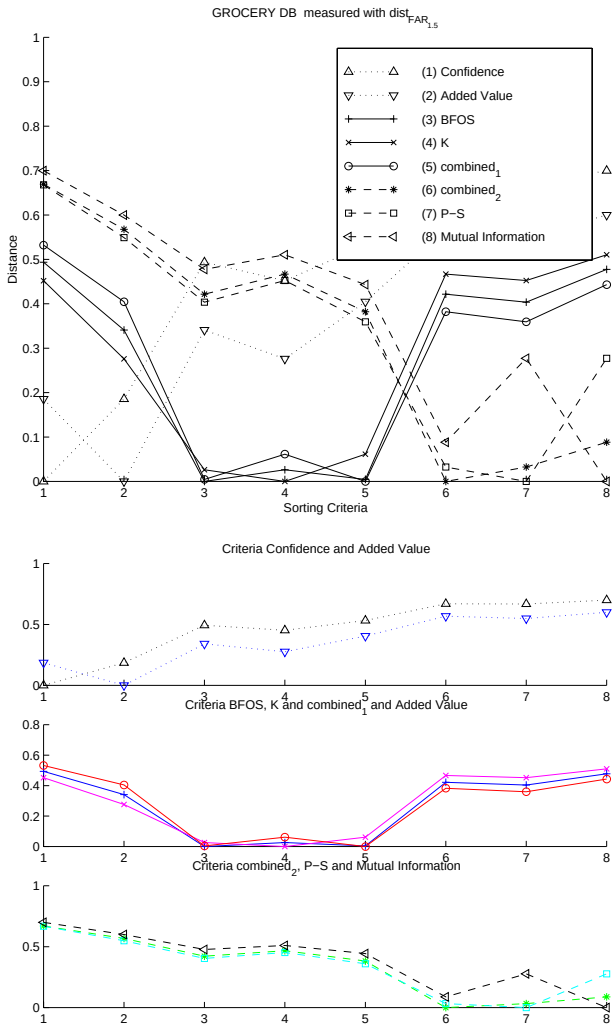

Figure 3.2: **Criteria distance measures of the Grocery Database measured with $dist_{far_{1.5}}(\pi_j, \pi_k)$.** *Each line in the graph details the distance measures from a particular criterion to all the other criteria. The lines therefore relate the behavior of the distances from one permutation to another. We can see three clusters amongst the permutations (and the respective criteria that define them) in the graph on the top. To see them more clearly, each of the three clusters is plotted in a separate subgraph on the bottom graph. Note that the distances between members of the same clusters are close to zero, while the distances to members of other clusters increase in accordance to how different or similar the clusters are.*

bers of the second cluster fall under 3% (with under 0.5% difference between $combined_2$ and the other members of the cluster), and between members of the second cluster they decrease to approximately 1%. For larger values of $\mathcal{L}$ the differences are insignificant. This implies that the criteria in each cluster (with the qualification on the first cluster that is comprised of 2 outliers) are very similar to one another.

The members of the first cluster exhibit the most drastically different behavior from all other criteria: even their top ten rules, the ones picked as the most interesting, are very different from the ones picked by all other criteria. This is to be expected, as the *confidence* criterion and the *AddedValue* criterion, defined in Section 3.1.2, measure different aspects. While the *confidence* criterion measures the confidence of a rule alone, the *AddedValue* criterion measures the confidence of the rule with respect to the probability of the conclusion (as in Example 3.1.1 with more on this in Section 3.1.5).

The members of the second cluster, the *MutualInformation*, *P-S* and $combined_2$, often manifest similar behavior patterns. Not only are their top picks for the most interesting rules similar, but the distance behavior of cluster members is similar to each other.

The tightest cluster is the third one, containing the criteria *BFOS*, *K* and $combined_1$. The top rules these members picked are practically identical, and the interestingness ranking order they impose on the rules is very similar to that imposed by other criteria in the cluster.

Criteria from the same cluster are thus interchangeable (even for small values of $\mathcal{L}$). This permits a significant and important reduction in the quantity of criteria we are trying to evaluate. Furthermore, for reasonable values of $\mathcal{L}$, members of the second and third cluster are close enough to each other to be interchangeable. The members of the first cluster are, on the other hand, very different from the members of the second and third cluster, thus, deciding to use them requires a subjective selection.

Figure 3.3 depicts the behavior of the 8 criteria, measured with the $dist_{\text{FAR}_{1.5}}$ measure, over three other databases. From top to bottom those databases are the WWW Proxy Logs Database, the Nursery Database and the Mushroom Database.

The bottom pair of graphs in Figure 3.1 and the bottom graph in Figure 3.3, depict the distance behavior over the Mushroom Database. In all three graphs, the distances between the criteria are practically nonexistent for all but the *confidence* and *AddedValue* criteria. When the distance is measured using the $dist_{\text{FAR}_{1.5}}$ measure, criteria 3–8 are less than 7% different,

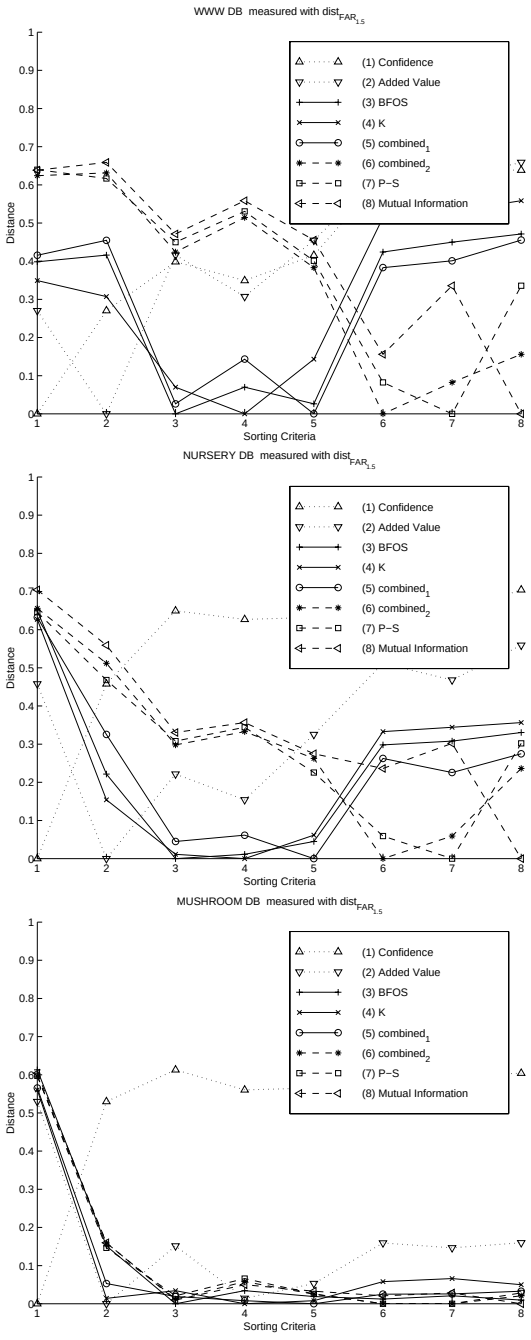

Figure 3.3: **Behavior of the ranking criteria over the WWW Proxy Logs Database, the Nursery Database and the Mushroom Database, measured using $dist_{far_{1.5}}$.**

whereas the same measure yields more than 50% difference for the same criteria over the Grocery Database, over 55% difference for the same criteria over the WWW Proxy Logs Database and over 35% over the Nursery Database. The differences between criteria 3–8 fall to about 28% for the Chess Database and to under 14% for the Adult Database. Our results indicate that the objective interestingness ranking criteria cannot readily distinguish between rules mined with high thresholds (35% for the Mushroom Database). The same effects can be seen, to a slightly lesser effect, for the Adult Database mined with 20% and to an even lesser extent for the Chess Database mined with 10% thresholds. The clusters seem apparent for rules mined with low thresholds.

3.1.5 Ranking Criteria: Further Analysis

The Confidence and the AddedValue Criteria

A significant confidence value for a rule indicates that in a significant number of transactions where the assumption is present, the consequent will also be present. It is a reasonable necessary requirement of an interesting association rule. However, it is not a sufficient one, as in Example 3.1.1. The confidence criterion is the conditional probability of the consequent given the assumption. The exclusion of any other measure means that the assumption and the consequent of a rule can be independent and the conditional probability will still be high. But any rule whose assumption and consequent are independent *cannot* be considered objectively (that is, according to objective criteria) interesting.

A and B are defined to be independent if

$$\mathbf{Pr}[AB] = \mathbf{Pr}[A] \cdot \mathbf{Pr}[B]. \tag{3.8}$$

Remember the use of the probability notations was defined in Equations 2.3 and 2.2, so that $\mathbf{Pr}[A] = \mathbf{Pr}[\tau(A) = \text{TRUE}]$, $\mathbf{Pr}[B] = \mathbf{Pr}[\tau(B) = \text{TRUE}]$ and $\mathbf{Pr}[AB] = \mathbf{Pr}[(A \cup B)] = \mathbf{Pr}[\tau(A \cup B) = \text{TRUE}]$.

Our loose use of the term probability interprets Equation 3.8 as the more different $\mathbf{Pr}[(A \cup B)]$ and $\mathbf{Pr}[A]\cdot\mathbf{Pr}[B]$ are, that is, the larger $|\mathbf{Pr}[(A \cup B)] - \mathbf{Pr}[A] \cdot \mathbf{Pr}[B]|$ is, the further from independence the consequent and the assumption are, and therefore, the more interesting the corresponding association rule $A \rightarrow B$ may be. This measure is easily converted to the *AddedValue* measure.

The confidence criterion is used *within* the association rule mining algorithm to derive rules from frequent itemsets (Section 2.2). In that process only association rules that have a *confidence* value greater or equal to the predefined confidence threshold will be mined. Following our observation that rules with independent assumption and consequents *can* be included in the "interesting" rule-list according to the confidence criterion, we used a filtering mechanism by applying the *AddedValue* criterion for the rules used in our experiments (Section 3.1.4), Only rules that have non-negligible (absolute value of) *AddedValue* are included in the list¶.

Effects of Pr[*A*]

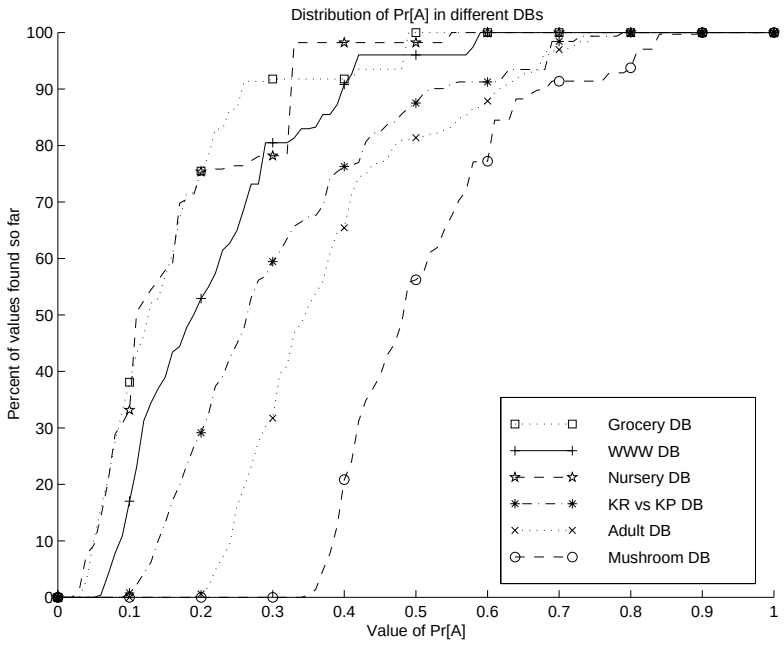

Figure 3.4: **Distribution of Pr[*A*] in the different databases**

The formation of, and membership in, the clusters is not immediately

¶For small values of *AddedValue*, only the *confidence* criterion remains unchanged. All the other criteria have a dependency on it (linear, quadratic or otherwise. We remember that the mutual information of two independent variables is zero—no information is gained on one by knowing the other).

self-evident; it would seem that some criteria that according to the empirical results belong in the same cluster, should not. This confusion can be resolved with some further inspection. For example, at first, it would seem that the criteria $BFOS(A \rightarrow B)$ and $K(A \rightarrow B)$ should not belong in the same cluster; $BFOS(A \rightarrow B)$ greatly favors rules with "large assumptions" (that is, ones with large $\mathbf{Pr}[A]$), as it directly dependent on $\mathbf{Pr}[A]/(1 - \mathbf{Pr}[A])$. The criterion $K(A \rightarrow B)$ has a significantly weaker preference to $\mathbf{Pr}[A]$ (it is dependent $\sqrt{\mathbf{Pr}[A]}$). Further examination of the two criteria allows us to express the $BFOS(A \rightarrow B)$ criterion as $[K(A \rightarrow B) \cdot K(A \rightarrow B)]/(1 - \mathbf{Pr}[A])$. Since we are interested only in the ranking of the rules, $K(A \rightarrow B)$ and $K(A \rightarrow B) \cdot K(A \rightarrow B)$ are identical for our purposes. Therefore, the difference between the two criteria is minor for all but large values of $\mathbf{Pr}[A]$. As we can see in Figure 3.4, those values are extremely rare in the three databases that were mined using thresholds under 10%, and uncommon in the others. Thus, the difference between the criteria $BFOS(A \rightarrow B)$ and $K(A \rightarrow B)$ criteria is inconsiderable.

3.2 Tradeoff: Many Rules Vs. Fewer Rules

> *The inherent vice of capitalism is the unequal sharing of blessings; the inherent virtue of socialism is the equal sharing of miseries.*
>
> —Sir Winston Churchill

A user runs the KDD process to discover the m most interesting rules that can be mined from a database, where m is determined by the user's preferences. To achieve this objective, a user can select one of two possible courses of action:

1. **Strict Thresholds Approach.** Run the mining algorithm with strict support and confidence thresholds so that the output will only include a few (approximately m) rules. The problem with this approach is that many rules that *should* be included in the top m most interesting rules might not be mined due to the strict thresholds used in the mining process.

2. **Lenient Thresholds Approach.** Run the mining algorithm with lenient thresholds and select the top m rules using a ranking criterion. The potential problem with this approach is that the output may include too many rules, possibly orders of magnitude more than m.

The second solution, that of mining with lenient thresholds, facilitates the mining of potentially interesting patterns that the strict thresholds approach might preclude. Therefore, this second approach seems the superior one.

The important question is, will the rules the algorithm with strict thresholds mined necessarily appear within the top rules the more lenient algorithm mined? Rephrased, the question is, how certain can we be that the rating criterion will rank amongst its top rules the rules with strict thresholds?

In order to answer this question, we ran the Apriori-TID mining algorithm (Section 2.2) on the six different databases (Section A.1) with both strict and lenient support and confidence thresholds. For example, we mined the Grocery Database using strict thresholds of 8% for both support and confidence thresholds. We also mined the Grocery Database using lenient support and confidence thresholds of 3.5%. After applying the 5% *AV* filter in both cases, this resulted in the mining of 205 association rules using the strict thresholds and an output of 2,279 association rules when mining with the lenient thresholds.

We used each ranking criterion listed in Section 3.1.2 to sort both the smaller and the larger sets of mined rules. Remember that the smaller sets are the output of mining with strict thresholds and the larger sets are the output of mining with the more lenient thresholds.

Then, for each ranking criterion, we calculated what percent of the rules from the smaller set of rules was discovered in an accumulated percent of the rules from the larger set of rules (mined with the more lenient thresholds). We plotted these results for the Grocery Database in the top graph of Figure 3.5. On the y-axis we have the percentage of rules from the small set (strict thresholds) discovered so far, and on the x-axis we have the percentage of rules from the larger set of rules (mined with the more lenient thresholds). For example, the top 10% of the rules mined from the Grocery Database with the lenient thresholds and sorted according to the *confidence* criterion (that is, the 227 rules with the highest confidence levels) contain only 14% of the top rules mined with the strict thresholds (that is, the top 29 rules from the list rules mined with the strict parameters sorted according to the *confidence* criterion). When the ranking is performed according to *P-S* criterion, the top 10% rules mined with the lenient thresholds (the 227 rules with the highest values according to the the *P-S* criterion from the list of rules mined with the lenient parameters) contain the top 76% of the rules mined with the strict thresholds (the 154 rules with the highest values according to the *P-S* criterion from the list of rules mined with the strict parameters).

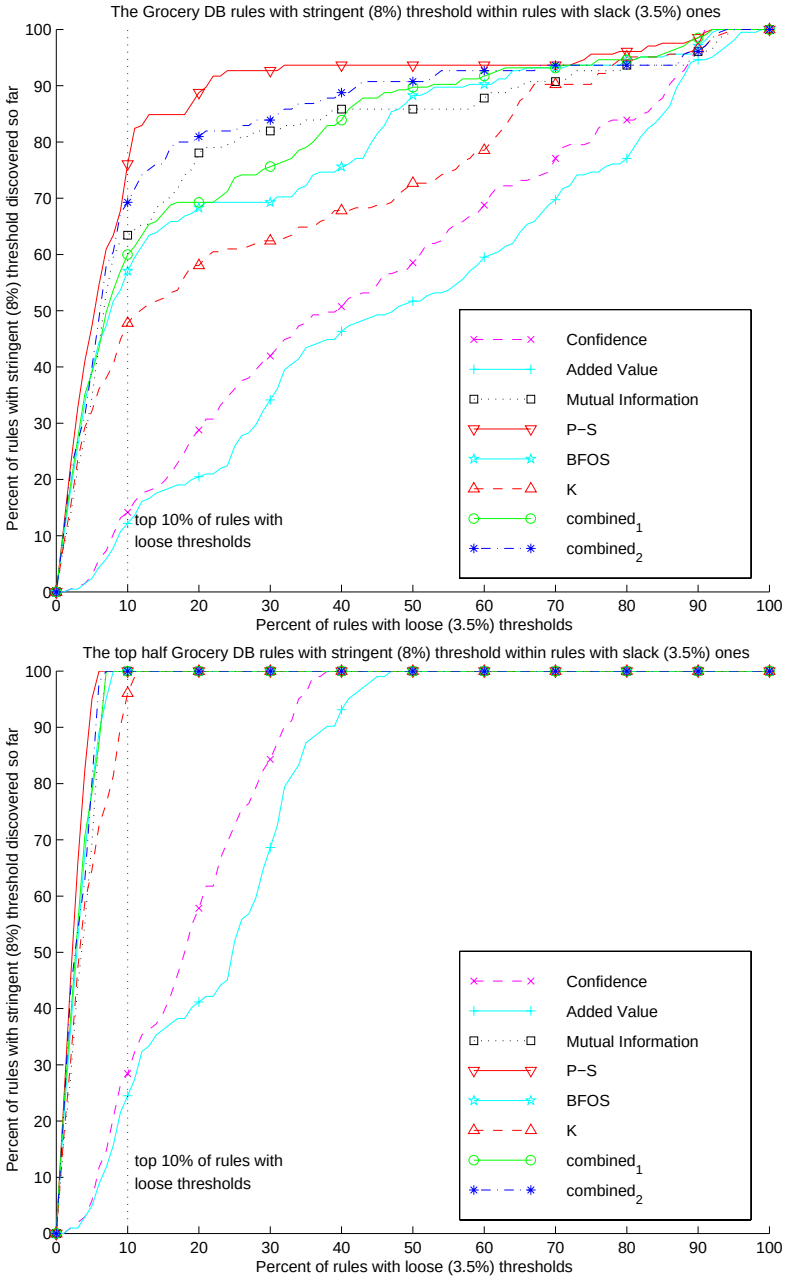

Figure 3.5: **Distribution of the mined rules from the Grocery Database with strict mining thresholds (8%) within the rules mined with lenient ones (3%).**

In the top graph of Figure 3.5 we see that the top 10% of the 2,279 rules mined from the Grocery Database using the lenient thresholds (where "top" is calculated using each one of the interestingness ranking criteria respectively) contain more than *half* of the 205 rules (mined from the Grocery Database using the strict thresholds) for all but the *confidence*, *AddedValue* and K ranking criteria (the latter contains 48% of the rules mined with the strict thresholds). It is important to note that it is the top 10% of the rules mined with the lenient thresholds sorted according to *each* criteria that include the top half of the rules (mined with the strict parameters), where again, top refers to the order imposed by sorting according to the respective ranking criterion.

The bottom graph in Figure 3.5 contains the same information only this time not all the rules mined with the strict thresholds are considered, but only those that were ranked at the top half, that is, the top 102 rules given each criterion respectively. It is extremely encouraging to notice that for most criteria (all but the *confidence*, *AddedValue* and K criteria, though with almost 96% of the rules for the K criterion), the top 10% of the rules mined with the lenient thresholds (227 rules) contain *all* the top half (102) rules mined with the strict thresholds (3.5% vs. 8%).

On the one hand, interesting rules with high support and confidence are ranked highly, but the interestingness criteria also add other rules, resulting in an interesting mix. This information reinforces our confidence in the sorting criteria we use: we can, with great certainty, run the data mining algorithm with rather lenient thresholds, knowing that the chosen sorting criterion will be able to sift through the *many* rules that are mined, and extract the ones that are of more value to the user.

In Figure 3.6 we see the results for the WWW Proxy Logs Database and for the Nursery Database. Once again, for the Mushroom Database, the results were different. The top 10% of the rules mined with the relatively lenient thresholds of 35% do not contain, for most of the ranking criteria, any of the rules mined with the strict thresholds of 45%. For the Adult Database, the top 10% of the rules mined with relatively lenient thresholds of 20% contain, in most cases, at least 20–30% of the rules mined with the strict thresholds of 35%. For the Chess Database, the top 10% of the rules mined with the relatively lenient thresholds of 10% contain as much as 20% of the rules mined with the strict thresholds of 20%. We attribute this to the very high thresholds used for the mining; we have found that if we use increasingly stricter thresholds, the results are more like those portrayed for

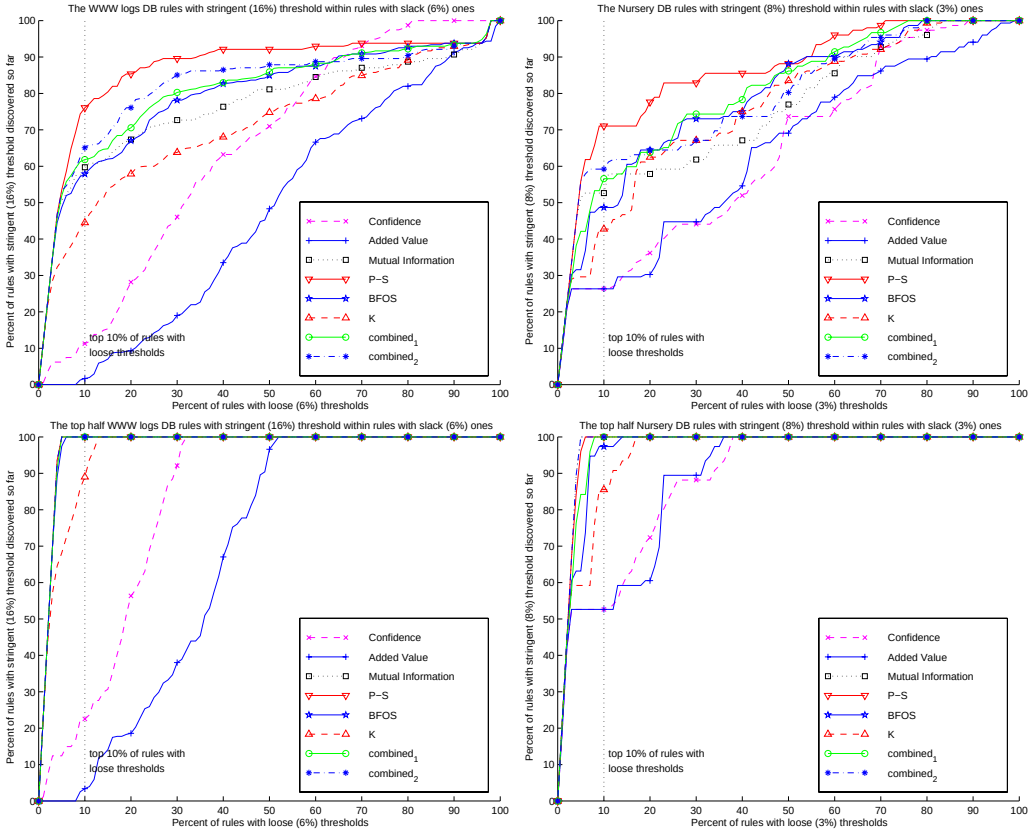

Figure 3.6: **Tradeoff between mining with strict and lenient thresholds depicted for the WWW Proxy Logs Database and the Nursery Database.** *The pair of graphs on the left depict the distribution of the rules from the WWW Proxy Logs Database mined with strict threshold (16%) within the rules mined with the lenient thresholds (6%). The pair of graphs on the right depict the distribution of the rules from the Nursery Database with strict thresholds (8%) within the rules with lenient ones (3%). For both pairs, the top graphs depict the distribution of all the rules mined with the strict thresholds within the rules mined with the more lenient thresholds, and the bottom graphs depict the distribution of the top half of the rules mined with the strict thresholds within the rules mined with the lenient thresholds for the two databases respectively (left and right).*

the first three databases, mined with much stricter thresholds.

3.3 Objective Interestingness: Summary and Discussion

Analytically defining a measure of interestingness is a very difficult task. We saw that ranking criteria can be used to determine the interest level of an association rule, and that there are *many* ways to define these ranking criteria. In fact, there is such a wide variety of ranking criteria to choose from, that the very task of selecting one of the many available interestingness criteria may seem to be almost as daunting as the initial task of constructing such a criterion. Our empirical evaluation shows that this is not always necessarily so. The ranking criteria can be classified into groups according to the manner in which they rank the rules. Choosing different criteria from the same groups will have minimal effect on the outcome (the interest-order that is imposed on the ranked rules). In some cases, when the rules were mined with very high thresholds, there was little difference between the ranking according to most of the objective interestingness criteria. Furthermore, our empirical evaluation shows that those criteria *are* worth using since they output a rich mixture of rules: ones with high support and confidence thresholds and others.

We take a closer look at the rules mined from the Grocery Database. The top two rules according to all (but the first two criteria, *confidence* and *AddedValue*) were:

- $\langle$`tomatoes`$\rangle \rightarrow \langle$`cucumbers`$\rangle$, and
- $\langle$`cucumbers`$\rangle \rightarrow \langle$`tomatoes`$\rangle$

indicating that a person who purchased tomatoes is likely to purchase cucumbers as well, and vice versa. We remember that this database describes the transactions of a grocery store in Israel. In Mediterranean countries, salads are a staple food, and in Israel salads generally contain both cucumbers and tomatoes.

As we proceeded to examine more rules (sorted by all but the first two criteria that behave very differently than the others), we discovered many rules that represent relationships between salad ingredients. These common

relationships make sense in the context of the Mediterranean appetite for salads.

We presented these results to our users at the company that provided us with the data. Their subjective opinions of the quality of the rules are the only "real" measure of the performance of the knowledge discovery process. The comments we received were mainly enthusiastic ones, since the rules we presented made sense to them. We also found an anomaly in the data that we could not explain. When we reported it, it boosted our users' confidence in our results, since the source of the irregularity pinpointed an experiment they ran, which they did not expect to be evident in the results. However, their enthusiasm was tinged with some degree of disappointment: most of the rules we presented were previously known, even if not always perfectly articulated.

The ultimate goal of the KDD process is to discover interesting [FPSS96b] rules, not ones that are previously known to the users of the KDD process. Unfortunately, the objective interestingness criteria fall short of that goal. In order to discover novel rules, we find ourselves needing to incorporate subjective interestingness criteria. We discuss subjective interestingness in the following chapters.

3.3.1 Other Related Works

The work presented in [SM99], which was the basis for the empirical evaluation of objective interestingness criteria in this chapter, indicates two things. On one hand, subjective interestingness criteria are ultimately needed to determine what is interesting to a specific user. Using only objective interestingness criteria, regardless of the specific criterion used, is rarely sufficient. On the other hand, applying the objective interestingness criteria can be a potentially important step in finding the interesting rules, as discussed in [Fre98], with many objective interestingness criteria developed [BJA99, SM99, HH00, HH01b, HH01a, TKS02]. Recently, [TKS02] described several key properties to be considered by a user before deciding what objective interestingness criteria to apply for a given domain application. One of the objective interestingness criteria that [TKS02] study is *AV*, which we introduced in this chapter (Definition 3). [TKS02] extends on the study of the properties and principles of objective interestingness measures which started with [PS91] who introduced three principles of objective interestingness criteria, which and have later been studied and expanded by [MM95, KS96].

[PH00] categorize objective constraints into five general classes. Very thorough, exemplary studies of objective interestingness criteria principles are presented in the works of [HH00, HH01b, HH01a]. Works that incorporate objective interestingness into the mining process are reviewed in Chapter 5 (based on [Sah02b]). Other types of objective interestingness works are reviewed in Chapter 7 (based on [Sah01]), where a new type of interestingness criteria, the impartial criteria, is also introduced.

The detailed analysis of the rules lent us much insight to favorable characteristics of interestingness criteria. Since to reach a complete interestingness solution, subjective interestingness criteria have to be applied we start by examining subjective interestingness in the following chapter where we introduce a new approach to subjective interestingness.

Chapter 4

Subjective Interestingness*

The important thing is not to stop questioning.

—Albert Einstein

Interestingness is ultimately a subjective problem since what is interesting to a user is ultimately subjective. Therefore, to completely address the problem of interestingness, subjective interestingness criteria, criteria that explicitly depend on users' needs and prior knowledge, will need to be applied.

In this chapter we introduce a new approach to subjective interestingness. Our approach is a simple and short minimalistic process of eliminating a substantial portion of *not*-interesting association rules from the list outputted by a data-mining algorithm. Instead of trying to establish what is interesting, we look for rules that are *not* interesting; more specifically, the simple rules whose elimination implies the automatic elimination of many other mined rules. This is significantly different from other approaches available in the literature. Our process requires very low-intensity user interaction: in three iterations, a user usually eliminates about 30% of the rules. A little over five iterations will usually halve the size of the problem. This elimination is a significant first step since size is the essence of the problem. It also shows that even within an extremely minimalistic framework to subjective interestingness, there are substantial advantages. We also present representative results of execution of the algorithm over three real databases.

*A preliminary version of this chapter was previously published in the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining [Sah99].

4.1 Introduction to Subjective Interestingness

> *It is of interest to note that while some dolphins are reported to have learned English—up to 50 words used in correct context—no human being has been reported to have learned dolphinese.*
>
> —Carl Sagan

What is interesting to users is ultimately subjective; something that is interesting to one user may be known or irrelevant, and therefore not interesting, to another user. To determine what is subjectively interesting we need to incorporate users' domain knowledge into the solution system. Modeling and incorporating domain knowledge has been the subject of intense research in many communities, most predominantly in the AI community. The general problem is still far from being solved, though it has been determined to be a very difficult one. Luckily, in order to determine the subjective interestingness of a rule, we do not need a complete model of all the user's domain knowledge, only the specific parts that pertain to the data at hand. Unfortunately, this, too, is a difficult problem.

4.1.1 Related Approaches in the KDD Literature

The KDD literature provided two main approaches to the problem of incorporating domain knowledge in order to determine the subjective interestingness of a rule. The first approach requires a domain expert or advanced user to formally (even if vaguely) express on demand, using a predefined grammar, what he or she finds (not) interesting or what a domain user already knows. [KMR+94] were the first to introduce this approach with the definition of pattern templates that describe the structure of interesting rules. [LHC97] define General Impressions, a more generalized form of templates that permits the expression of imprecise domain knowledge. [SVA97] introduce user constraints and taxonomical constraints into the mining process, and [NLHP98] suggests generalized user constraints that allow user feedback at established breakpoints. In the novel approach of [AT97] interestingness is determined according to user defined actionability hierarchy. [TS96, PT98, PT00] incorporate the set of user beliefs to determine interestingness. [AT01, AT99, TA02] introduce a new approach to incorporating interestingness through expert-driven validation of very large numbers of rules by the iterative application of various rule validation operators. These operators include a grouping method based on attribute hierarchies that allows

the domain expert to validate many rules at the same time. [AT01, AT99] concentrate on the personalization application domain and [TA02] is set in the bioinformatics domain, providing biologists tools to analyze the large number of rules mined on microarray data.

There are many variations of the methods that fall under this approach, with one common characteristic: the success of these strategies is conditioned upon the availability of a domain expert willing to go through the significant effort of completing the task of defining all the required domain knowledge, either specifically for the needs of the KDD process, or to fulfill another need. Unfortunately, acquiring such an expert for the duration of the process is a costly procedure, sometimes even an unfeasible condition. While sometimes the required domain knowledge may be obtainable from a pre-existing knowledge base (Section 4.6), oftentimes such expertise is not readily available.

The second approach, taken in [Sub98], constructs the knowledge base by having users classify every single rule. This approach does not demand an advanced user to come up with a list of what is or is not interesting in the domain. However, it requires very intensive user interaction, this time of a mundane nature. Although the process can be done incrementally, it, as the author says *"may be argued* [to be ...] *tedious"*. In some cases, the sheer amount of work involved in this approach may render it infeasible.

4.1.2 Our Method: Interestingness Via What Is Not Interesting

We tackle the problem of interestingness by taking advantage of the basic difficulty it presents: the majority of the mined rules are *not* interesting. Our minimalistic approach is to quickly eliminate large families of rules that are *not* interesting, while limiting our user-interaction to a few, simple classification questions. These questions are specifically chosen to facilitate the potential elimination of large numbers of association rules. At the same time, we start the construction of a domain knowledge base.

We start our algorithm with the exhaustive list of mined rules. This list can easily consist of many thousands of rules if the thresholds are set to relatively low values. If we were to present the list of mined rules for user classification, most rules would be classified as not interesting, indicating that they can be eliminated from the list of potentially interesting rules. We

capitalize on this feature: our approach is to ask a user to classify only a few rules, specifically chosen so that their elimination can bring about the automatic elimination of many other rules. The information we gather during the elimination process is stored in the knowledge base we construct. This approach has several benefits:

1. We circumvent the major difficulty of defining *why* a certain rule is interesting to a user (list of possible reasons in [Klo96]). Once a user denotes a rule as interesting, we take the classification at face value. Here we are primarily interested in what is *not* interesting (again, without having to define it explicitly) in order to arrive at what *is* interesting. This feature lends the name to our algorithm: **Interestingness Via What Is Not Interesting**.

2. We ask only minimal classification questions. The questions are not descriptive in nature (as in "provide us with a template of what is (not) interesting to you"), and therefore easier and quicker for a user to answer.

3. We make the classification process less complicated by having a user classify very simple rules. The rules we present for user classification are simple in the sense that they contain only a single attribute in both the assumption and the consequent: they are of the form $a \rightarrow b$ where both a and b are attributes. We never ask a user to classify more complex rules, of the form $\langle a \wedge b \wedge c \wedge \cdots \rangle \rightarrow \langle x \wedge y \wedge z \wedge \cdots \rangle$.

4. Every user classification can eliminate an entire family of rules, not only a single rule (see Section 4.4). Since we infer the utmost from every user interaction we need to ask fewer questions (Section 4.5). As we will see, usually, after the first three user classifications, approximately 30% of the rules are eliminated from the original list of mined rules.

5. We do not require a domain expert to classify the rules; a naive user can successfully classify most of the rules we present. For example, in the case of the Grocery Database (Section A.1.1), the first rules brought for classification were $\langle$`tomatoes`$\rangle \rightarrow \langle$`cucumbers`$\rangle$ and $\langle$`cucumbers`$\rangle \rightarrow \langle$`tomatoes`$\rangle$. Armed with the knowledge that salads are a staple food in Israel, commonly prepared from cucumbers and tomatoes, any user can easily classify these rules.

The process we present is a very low-intensity process in both the *volume* of work expect of a user (we only ask a few questions: low-volume interaction) and the *level* of interaction required of a user (we only ask simple classification questions: low-level interaction).

Our achievements from the user interactions are twofold: first, we incrementally construct a domain knowledge base by gathering the relevant domain knowledge gained from a user's classifications. The knowledge base is needed for future steps in the process of converging on the list of patterns that are interesting to a user in the current domain. At the same time, we work towards the creation of a concise list of the potentially interesting rules by progressively eliminating rules we learn are not of interest to a user. This elimination process is very fast. Three iterations of our algorithm are usually sufficient to eliminate approximately 30% of the rules. A little over five iterations will often reduce the list to half its original size. The Interestingness Via What Is Not Interesting approach we present in this chapter also aids us in the construction of a knowledge base. It is the first big step in the formation of a comprehensive solution to the universal (domain-independent) interestingness problem.

The goal of the Interestingness Via What Is Not Interesting algorithm is to present a minimalistic framework to subjective interestingness that uses very little domain knowledge and still carries substantial advantages in the form of deleting a significant portion of the not-interesting rules. To achieve this, we intentionally restrict the amount of domain knowledge the Interestingness Via What Is Not Interesting approach has access to. Given additional domain knowledge, much more can be achieved. Influential works such as [PT00, PT98] have set the bar on what elimination can be achieved given additional beliefs; these works propose methods to use domain knowledge in the form of a set of user beliefs in order to discover a *minimal* set of unexpected patterns that is orders of magnitude smaller than the exhaustive list of mined patterns. This is an important problem: in [FPSS96a], Fayyad et al. listed *"User interaction and prior knowledge"* as one of the *"research and application challenges for KDD"*.

We stress that the algorithm presented in this chapter does not attempt to provide a complete solution to the problem of determining, in any domain, precisely which rules are subjectively interesting. The overall problem, from the formal definition of interestingness, through the methodic construction of a domain knowledge base, to the selection of the few subjectively interesting rules is currently beyond the scope of a single work. It still requires a sig-

nificant cooperative research effort on part of the KDD community. Instead, we present a novel approach that very quickly, using a few simple questions, drastically reduces the size of problem, size being the essence of the problem.

The results we present in this chapter reflect that the Interestingness Via What Is Not Interesting approach is intended to be a first step towards the universal (domain-independent) subjective interestingness problem. Our goal in this algorithm is to eliminate a significant portion of the not-interesting rules, rather than to be a complete solution for converging on the short-list of interesting rules. As we will see in Section 4.5, we eliminate a considerable number of the not-interesting rules. We recommend to use this approach to eliminate many, but not all, of the not-interesting rules. The first few eliminations will yield the majority of the benefit of the algorithm. The motivation of this approach was that business users are often unwilling to respond to many tedious questions, and we therefore only present the results yielded from a few classification questions. In fact, at the beginning of each iteration of the algorithm we present users with an indicator of the potential impact of going through with the next iteration. This indicator is used to determine when the potential results (potential elimination of a portion of the not-interesting rules) of the next iteration no longer merit the effort of another user classification, and the algorithm should be stopped then.

4.1.3 Differentiation of the Three Subjective Interestingness Approaches: an Analogy

Knowing I lov'd my books, he furnish'd me
From mine own library with volumes that
I prize above my dukedom.

—William Shakespeare (The Tempest)

In Section 4.1.1 we reviewed the two existing approaches to employing domain knowledge in order to address the problem of determining subjective interestingness of patterns. In Section 4.1.2 we briefly introduced our new approach to the same problem. To highlight the differences between these three approaches, we present analogy to the problem of employing domain knowledge in order to recommend an interesting book to a reader (that is, determine which book[s] is/are interesting).

Approach 1. Ask the reader (user) to describe, in a predefined grammar, what types of books are interesting (or not interesting) to him/her.

The questions in this category can be more specific, asking the reader to describe certain characteristics of the books he/she finds interesting.

Approach 2. Ask the reader to classify all the books that the recommender currently has in the library as interesting or not-interesting. As new books get added to the library, the recommender will be able to determine if the new books are potentially interesting or not to the reader.

Approach 3. Ask the reader to classify a few specifically chosen short stories as interesting or not-interesting. With as few as five short stories classified by the reader, the recommender will be able to eliminate about half the books in the library as indubitably not interesting to the reader, significantly reducing the search space for interesting books.

The significantly different requirements of each of the three approaches dictate the difference between them. The first approach will yield the best book recommendations if the reader is able to accurately and precisely describe the characteristics of the types of books he/she likes (dislikes). However, these descriptive questions are generally very demanding and difficult for any user to answer. The second approach only asks classification questions, but the number of questions (one per book) makes this approach unappealing to most users. Additionally, this second approach, asks readers (users) to classify books. This type of classification may not be straightforward to all readers (it translates to the classification of long or complicated rules, which naive users may not find easy). The third approach asks very simple questions (specific short stories are brought up for reader classification), but its results are only the first phase towards making accurate, definite recommendations.

Approach 1 is the most popular method for employing subjective interestingness in the KDD literature. It is likely to yield the short list of potentially interesting rules. This is the first approach described in Section 4.1.1; in this approach, prior knowledge and user needs are obtained by explicitly asking domain experts to express all the required knowledge in a predefined grammar, whether specifically for this task or for another task. The second approach only uses classification questions, but a prodigious number of them and can only be used in certain applications. The third approach, which we present in this chapter (originally in [Sah99]), requires very low-intensity user interaction and thus presents a promising first step towards incorporating subjective interestingness.

Note that although we used a recommendation analogy, the Interestingness Via What Is Not Interesting approach is not directly related to recommendation systems such as movie-, music- and book-recommendation systems. These systems use one or a combination of the following approaches, (1) **Preferences and Constraints**: similar to Approach 1, where in this case, users express preferences for (or exclude) certain genres of movies, music or books according to a predefined scale (categorical or continuous). For each recommendation type, users can also express subcategory constraints. For example, for recommendation of movies users can also express preference to certain directors and/or actors. For books, users can also express preferences to specific authors and/or series, and for music these subcategory preferences can be to musicians, musician groups, labels, etc. In the Interestingness Via What Is Not Interesting approach we circumvent the need to use such constraints. (2) **Collaborative Filtering**: used in some form by most large recommendation systems, where a recommendation for (prediction on) an unseen item is made using the ratings previously made by "similar" users. This method therefore requires large knowledge base (many recommendations by many different users), a requirement we reject in the Interestingness Via What Is Not Interesting approach. Additionally, the goal of the Interestingness Via What Is Not Interesting approach is to eliminate a significant portion of the not-interesting rules, rather than to specifically recommend interesting ones, the goal of recommender systems.

The rest of this chapter is organized as follows. In Section 4.2 we provide definitions and notations that are used in the Interestingness Via What Is Not Interesting approach. We present and explain the different rule classifications available to the user in Section 4.3. In Section 4.4 we introduce the algorithm including the processing of the rules according to each of the classifications, followed by empirical validation of the algorithm in Section 4.5. We end with a summary in Section 4.6.

4.2 Definitions and Notations

Words have meanings and names have power.

—Babylon 5 (Lorien in Whatever Happened to Mr. Garibaldi)

In addition to the definitions and notations provided in Chapter 2, the Interestingness Via What Is Not Interesting algorithm we describe in this chapter requires the introduction of a few new concepts.

4.2.1 New Definitions: Ancestor Rule, Coverage List and Family

If you cannot get rid of the family skeleton, you may as well make it dance.

—George Bernard Shaw

Definition 4 *Let Ω be a list of association rules over Λ. Let $a, b \in \Lambda$, and $\rho = a \rightarrow b$. Note that ρ may or may not be in Ω. We define the* ***coverage list of ρ in Ω****, $CL_\Omega(\rho)$, as:*

$$\mathbf{CL_\Omega}(\rho) \stackrel{\mathbf{Def}}{=} \{r \in \Omega | r = A \rightarrow B \ , a \in A, b \in B\}. \tag{4.1}$$

The coverage list of ρ in Ω contains all the rules in Ω that have a in their assumption and b in their consequent ($\rho \notin CL_\Omega(\rho)$ when $\rho \notin \Omega$).

We now define the **family of ρ in Ω**, $family_\Omega(\rho)$, as:

$$\mathbf{family_\Omega}(\rho) \stackrel{\mathbf{Def}}{=} CL_\Omega(\rho) \cup \{\rho\}. \tag{4.2}$$

We define the **ancestor rule** of the family of ρ in Ω to be ρ. When there is no room for confusion, we can omit the Ω notation.

Note that the definitions above, of coverage list in Equation 4.1 and of a family in Equation 4.2, can be defined over any arbitrary set of rules, not only the exhaustive list of mined rules.

4.2.2 $RSCL_\Omega(\rho)$ and Its Properties

When you are courting a nice girl an hour seems like a second. When you sit on a red-hot cinder a second seems like an hour. That's relativity.

—Albert Einstein

An indicator of the size of the coverage list of ρ relative to Ω is $RSCL_\Omega(\rho)$, the **relative size of the coverage list of ρ in Ω**, which we define as:

$$RSCL_\Omega(\rho) \stackrel{\mathbf{Def}}{=} \frac{\| CL_\Omega(\rho) \setminus \{\rho\} \|}{\|\Omega\|}, \tag{4.3}$$

where $\|X\|$ is the number of elements in the set X. Note that if ρ is the only rule in Ω that contains a in its assumption and b in its consequent, then $RSCL_\Omega(\rho) = 0$.

Claim 3 *Let Ω be the exhaustive list of rules outputted by the data-mining algorithm. A rule $\rho = a \rightarrow b \notin \Omega$ may have $RSCL_\Omega(\rho) > 0$.*

Proof: To prove the claim, it is sufficient to show for a single example that $\exists \varrho \in \Omega : \varrho = xz \rightarrow y$, since $\rho = x \rightarrow y \notin \Omega$, and we want to show $RSCL_\Omega(\rho) > 0$. We construct such an example. Assume that Ω was attained by mining the association rules with a minimum support threshold of $s = 0.04$ and a minimum confidence threshold of $c = 0.4$. We consider a possible scenario where $support(x) = 0.4$, $support(xy) = 0.1$, $support(xz) = 0.1$ and $support(xyz) = 0.05$. We calculate the support and confidence of both rules, ρ and ϱ: $support(\varrho) = 0.05$, $confidence(\varrho) = 0.5$, and for ρ, $support(\rho) = 0.1$ and $confidence(\rho) = 0.25$. Since $support(\varrho) \geq s$ and $confidence(\varrho) \geq c$, we have $\varrho \in \Omega$. Since $\varrho = xz \rightarrow y$ and $\rho = x \rightarrow y$ and $\varrho \in \Omega$, we have $\varrho \in CL_\Omega(\rho)$. Therefore, $RSCL_\Omega(\rho) > 0$. Now, $support(\rho) \geq s$ but $confidence(\rho) < c$ thereby indicating that $\rho \notin \Omega$ and completing our proof. ■

4.3 User Classifications

> *"Just one question. Why?"*
> *"Why not?"*
> *"It's not an answer."*
> *"Oh, yes it is. It's simply not an answer you like or the answer you expect. There's a difference."*
>
> —Babylon 5 (Sakai and G'Kar in Mind War)

The rules we present for user-classification are ancestor rules only. Every time we present a rule, ρ, for user-classification, we want a user to tell us:

1. Whether the rule is true, which is equivalent to classifying the rule as TRUE or NOT-ALWAYS-TRUE[†], and,

2. Whether the user is (subjectively) interested in any rule of the family of the classified rule. This is equivalent to classifying the $family_\Omega(\rho)$ as "INTERESTING" or "NOT-INTERESTING".

The first classification, specifying whether or not a rule is true, is used in the construction of the knowledge base. Many statistically significant rules may

[†]Note that we use the classification "NOT-ALWAYS-TRUE" as opposed to "FALSE" which might imply to some users that there is no instance that can satisfy the rule.

be common knowledge even to a naive user with no domain-specific experience (for example, ⟨`pregnant`⟩→⟨`woman`⟩). We need to exclude these rules from the list of interesting rules we propose to present to the user. The second classification is a broader one, characterizing the entire family of the rule, that is, any mined association between the assumption and the consequent. These two categorizations define four possible classifications for each rule: "TRUE-NOT-INTERESTING", "NOT-ALWAYS-TRUE-INTERESTING", "NOT-ALWAYS-TRUE-NOT-INTERESTING", and "TRUE-INTERESTING."

Some of the classifications are not self-evident, for example, how a user can be interested in a family of a rule he or she classified as not-always-true. In this section we present and explain the four classifications. We will also see why it is very easy to make these classifications, even for a large family of rules. In Section 4.4 we describe the processing of the mined rules according to each classification.

4.3.1 The True-Not-Interesting Classification

The rule $r_{hm} =$ ⟨`husband`⟩→⟨`married-civ-spouse`⟩ is an example of a rule that is classified as "TRUE-NOT-INTERESTING." This rule was mined from the Adult Database (Section A.1.3). It has support and confidence levels of approximately 40% and 90% respectively.

The word "husband" is defined as the male partner in marriage. Any person familiar with this definition (not only a domain expert) can easily deduce that:

1. r_{hm} is a true and not-interesting rule, and,
2. any rule specifying when a husband is a married[‡] person is not interesting.

The first inference is the classification of r_{hm} as "TRUE"—a knowledge nugget we can add to our knowledge base. The second inference is the classification of the *entire* rule's family as not-interesting, indicating that all the rules that have the attribute `husband` in their assumption and the attribute `married` in their consequent are to be classified automatically as not-interesting as well (as is detailed in Section 4.4.3).

[‡]Since over 97% of the married people surveyed for the Adult Database were married to a civilian spouse, we ignore the `civ`ilian subcategorization for the sake of simplicity in our examples.

Note that although a user is required to classify an entire family, it is a very easy classification to make.

4.3.2 The Not-Always-True-Interesting Classification

The NOT-ALWAYS-TRUE-INTERESTING classification is perhaps the most confusing one. We examine the rule:

$$r_{mm} = \langle \texttt{male} \rangle \rightarrow \langle \texttt{married-civ-spouse} \rangle, \tag{4.4}$$

mined from the Adult Database (Section A.1.3). r_{mm} has support and confidence levels of approximately 40% and 60% respectively.

We consider a user interested in researching or targeting married males (possibly for a marketing campaign). This user may be interested in what qualifications on a male provide a higher confidence that the male is married. A hypothetical example for such a rule is $\langle \texttt{male} \bigwedge \texttt{owns minivan} \rangle \rightarrow \langle \texttt{married} \rangle$.

This user will choose to have rules that contain the attribute `male` in the assumption and the attribute `married` in the consequent be presented to him or her in the future. At the same time, this user recognizes that the rule $\langle \texttt{male} \rangle \rightarrow \langle \texttt{married-civ-spouse} \rangle$ is generally *not* true; not all males are married. Since the user determined the family of r_{mm} to be interesting, and at the same time, the actual rule to be not-true in general (a possible aberration in the data), r_{mm} is classified as "NOT-ALWAYS-TRUE-INTERESTING" by this user. The processing of this classification is detailed in Section 4.4.3.

4.3.3 The Not-Always-True-Not-Interesting Classification

We consider a different user, this one interested in targeting people with annual income greater than $50,000. Again, we examine the rule r_{mm} from Section 4.3.2 defined in Equation 4.4. This new user, who is interested in the conditions for annual income of $50,000 or more, classified r_{mm} as "NOT-ALWAYS-TRUE-NOT-INTERESTING". The rule was, again, determined to be untrue for the same reason: not all males are married. The user in this example is not interested in the conditions that determine whether a person is married, and therefore, this user classified the family of the rule as not-interesting. In the previous example (Section 4.3.2), this same rule family was classified, by a different user, as interesting.

The *subjective* interests of the different users determine whether the rule is classified as "NOT-ALWAYS-TRUE-INTERESTING" or as "NOT-ALWAYS-TRUE-NOT-INTERESTING."

4.3.4 The True-Interesting or Unqualified-Interesting Classifications

A user classifies any valid rule he or she determines to be subjectively interesting as "TRUE-INTERESTING." These rules are as varied as the subjective interests of different users. A user may be able to determine right away that a potentially interesting rule is not always true, and classify it as "NOT-ALWAYS-TRUE-INTERESTING." In some rare cases, a user may be able to determine that a rule is both interesting and true (which is the ultimate objective of determining interestingness). When a user is unable to determine whether a rule is true or false, further investigation is warranted before rule classification is possible. We therefore provide an additional classification, the "UNQUALIFIED-INTERESTING." Throughout our experiments, we have never run across a "TRUE-INTERESTING" classification. Ancestor rules that were determined to be interesting to a user were classified as either "NOT-ALWAYS-TRUE-INTERESTING" or as the "UNQUALIFIED-INTERESTING."

Our goal is to keep the Interestingness Via What Is Not Interesting approach minimalistic, making the classification process as simple as we can for users. To do that, we provide the guidelines that an ancestor rule a naive user cannot quickly classify as not-interesting and/or not-always-true (either as "NOT-ALWAYS-TRUE-NOT-INTERESTING," as "NOT-ALWAYS-TRUE-INTERESTING"), or as "TRUE-NOT-INTERESTING"), should be classified interesting ("UNQUALIFIED-INTERESTING"). In this we are making the worst-case assumption: whenever a naive user cannot immediately decide that the ancestor rule can be classified as not-interesting or not-always-true, we assume that that it is interesting and recommend it be classified as such. There are powerful advantages to including more classification classes, such as UNDECIDED, as in the influential expert-driven rule validation approach introduced in [AT01, AT99] in the personalization application domain and [TA02] in the bioinformatics domain. This important expert-driven rule validation approach gives a domain expert the flexibility to classify groups of rules as "interesting" or "not-interesting" when the expert is certain of the classification, and as "undecided" when further analysis is required to determine their classification.

(1) INITIALIZATION:
    Ω = list of rules outputted by the data-mining process;
(2) `while` $(\langle \exists r \in \Omega \rangle \bigwedge \langle$user is willing to continue$)\rangle$
(3) $r =$ **best_candidate**(Ω); /* Note: r does not necessarily
                                * appear in Ω */
(4) $c =$ **user_classification**(r);
(5) **process_classification**(c, r, Ω);
(6) `endwhile`

Figure 4.1: **Overview: the Interestingness Via What Is Not Interesting Algorithm**

4.4 The Algorithm: Interestingness Via What Is Not Interesting

> *Program: a set of instructions, given to the computer, describing the sequence of steps the computer performs in order to accomplish a specific task. The task must be specific, such as balancing your checkbook or editing your text. A general task, such as working for world peace, is something we can all do, but not something we can currently write programs to do.*
>
> —Unix User's Manual (Edit: A Tutorial)

In this section we describe our method of quickly and efficiently eliminating not-interesting rules from the list of rules outputted by the data-mining algorithm. This method, the **Interestingness Via What Is Not Interesting** algorithm, is outlined in Figure 4.1. We used the mining process described in Section 2.2 to initialize Ω in step (1), though any other association rule mining algorithm can be used.

The body of the algorithm consists of three iterative steps, steps (3)–(5) in Figure 4.1. The iterative process can continue as long as the list of rules we process, Ω, is non-empty. Since both the elimination process and the construction of the knowledge base are incremental, a user can stop the process at any time; the user is notified when stopping may be advantageous as detailed in Section 4.4.1.

The iterative process starts by selecting an ancestor rule and presenting it to the user for classification. We call the chosen ancestor rule the **best candidate**. A user's classification of the best candidate dictates the course

of action we take, usually consisting of a partial or complete elimination of the best candidate's family from Ω (Section 4.4.3). The selected rule is called best candidate since it is the ancestor rule that is the best candidate for elimination: it is chosen so that out of all the other ancestor rules, its elimination will bring about the elimination of the most rules.

The remainder of the section provides a detailed description of the iterative steps of the algorithm, and the processing of the list of mined association rules. Throughout this section, Ω will denote the current list of association rules. Examples of how our procedure works on real databases are provided in Section 4.5.

4.4.1 Selecting the Best Candidate

A man cannot be too careful in the choice of his enemies.

—Oscar Wilde

Our goal is to quickly eliminate from the list of rules outputted by a data-mining algorithm as many of the rules that we know are subjectively not-interesting to a given user. To ensure that this process of elimination of the not-interesting rules will be both a low-intensity and low-level process on part of a user, we ask a *few* (low-intensity) *simple* (low-level) questions, and remove as many rules as possible from Ω with every classification made. As we will see in Section 4.4.3, user classifications provide enough information to allow us to eliminate the ancestor rule's family partially or in its entirety. The larger the family, the more patterns we will be able to remove from Ω in a single step. Thus, we choose the best candidate to be the rule with the largest coverage list (see Section 4.4.3 for a thorough investigation).

Formally, the best candidate is chosen from Ψ:

$$\Psi = \{a \rightarrow b | a, b \in \Lambda, \forall c, d \in \Lambda, RSCL(a \rightarrow b) \geq RSCL(c \rightarrow d)\}. \qquad (4.5)$$

If more than one ancestor rule has the highest $RSCL$ value, we give preference to rules $a \rightarrow b \in \Psi$ such that $b \rightarrow a \in \Psi$[§]. Note that the selected best candidate may or may not be in Ω (see Claim 3). Our choice of the best candidate and its functionality are not affected by this factor.

Our algorithm is implemented so that the best candidate is always found in a single pass over Ω (low complexity). Remember that best candidates are

[§]From a pair of such rules, we choose the best candidate randomly.

ancestor rules, that is, of the form $a \rightarrow b$ where a and b are attributes. The brute force method of finding the best candidate is to calculate the $RSCL$ value for all pairs of attributes a, b, but that is a very wasteful method. Instead, we find the best candidate calculating the $RSCL$ value of only possibly ancestor rules, those in $\{a \rightarrow b | \exists \rho = A \rightarrow B \in \Omega, a \in A, b \in B\}$. We do this in a single pass over Ω. During the pass, for each rule $A \rightarrow B$, we increment a counter value for any ancestor rule $a \rightarrow b$ such that $a \in A$ and $b \in B$ and $a \rightarrow b \neq A \rightarrow B$. All those counters are initialized to one upon their first encounter with a rule that is not equal to the ancestor rule itself. After a single pass over Ω these counters hold the $RSCL$ value of each ancestor rule, the largest one is then selected to be the best candidate.

Example 4.4.1 *Let* $r_1 = x \rightarrow y$, $r_2 = x \rightarrow y \wedge z$, $r_3 = x \rightarrow z \wedge w$, *and* $r_4 = x \wedge y \rightarrow z$. *Let* $\mathcal{C} = \{r_1, r_2, r_3, r_4\}$ *be the set of rules we are examining. We have* $RSCL_{\mathcal{C}}(x \rightarrow y) = 1/4$ *since* $CL_{\mathcal{C}}(x \rightarrow y) = \{r_1, r_2\}$, *and by definition,* $RSCL_{\mathcal{C}}(x \rightarrow y) = \| CL_{\mathcal{C}}(a \rightarrow y) \setminus \{r_1\} \| / \| \mathcal{C} \|$. *Similarly, we have* $RSCL_{\mathcal{C}}(x \rightarrow z) = 3/4$ *where* $CL_{\mathcal{C}}(x \rightarrow z) = \{r_2, r_3, r_4\}$, $RSCL_{\mathcal{C}}(x \rightarrow w) = 1/4$ *and* $RSCL_{\mathcal{C}}(y \rightarrow z) = 1/4$. *The best candidate in this case is* $x \rightarrow z$.

4.4.2 User Classification of Best Candidate

> *In all affairs it's a healthy thing now and then to hang a question mark on the things you have long taken for granted.*
>
> —Bertrand Russell

Once selected, the best candidate, r, is presented for user classification, where the available rule-labels are detailed in Section 4.3. Along with the rule we present, if relevant (since the best candidate may not have been mined as in Claim 3) its support and confidence levels and any user defined objective interestingness criterion to assist in making the classifications. A list of possible criteria can be found in Chapter 3.

We ensure a fast and straightforward classification process by presenting users with ancestor rules only: a user more readily interprets simple rules than complex ones, and can more easily identify the family of a simple rule.

The $RSCL$ of the best candidate ancestor rule is always presented with the best candidate, to be used by a user to determine when to stop the iterative process. $RSCL_{\Omega}(r)$ is well suited for this purpose since it is an indicator of how many rules can be potentially eliminated in the current iteration. More rules will usually be eliminated in the first few iterations of

the algorithm than in future iterations. Once the $RSCL_\Omega(r)$ falls below a given threshold for more than two consecutive iterations, the user is notified. At this time, the user can decide whether it is still profitable to continue responding to questions when only a relatively small percent of the rules can be eliminated with each new classification made, as indicated by the $RSCL_\Omega(r)$. The value of the notification threshold can be adjusted by the user to match his/her needs. Note that a user can decide to stop the iterative process at any point, not only when notified that the $RSCL_\Omega(r)$ value fell below a given threshold.

4.4.3 Action Taken According to User Classification of the Best Candidate

> *Physics tells us that for every action there must be an equal and opposite reaction. They hate us, we hate them, they hate us back. And so, here we are: victims of mathematics.*
>
> —Babylon 5 (Londo in A Voice in the Wilderness #1)

In this section we detail the processing Ω undergoes according to the user classification, c, of the ancestor rule, $r = a \rightarrow b$, other than the collection and storage of the information in the knowledge base we construct. To aid us in our analysis, we will use the following claim:

Claim 4 *Let $A, B, C \subseteq \Lambda$ and $b \in \Lambda$ such that $A, B, C \neq \emptyset$ and $\{b\} \cup C = B$. Let Ω be the set of association rules mined by a data mining algorithm with minimum support threshold s and a minimum confidence threshold c. If $A \rightarrow B \in \Omega$ then $A \rightarrow C \in \Omega$.*

Proof: $A \rightarrow B \in \Omega$ implies that:

1. $support(A \rightarrow B) = \mathbf{Pr}[(A \cup B)] \geq s$, where $\mathbf{Pr}[(A \cup B)] = \mathbf{Pr}[\tau(A \cup B) = \text{TRUE}]$ as in Equation 2.2, and

2. $confidence(A \rightarrow B) = \mathbf{Pr}[(A \cup B)]/\mathbf{Pr}[A] \geq c$, where $\mathbf{Pr}[A] = \mathbf{Pr}[\tau(A) = \text{TRUE}]$ as in Equation 2.3.

Using 1., where, as in Equation 2.2, $\mathbf{Pr}[(A \cup C)] = \mathbf{Pr}[\tau(A \cup C) = \text{TRUE}]$, and $\mathbf{Pr}[(A \cup \{b\} \cup C)] = \mathbf{Pr}[\tau(A \cup \{b\} \cup C) = \text{TRUE}]$, we have:

$$\begin{aligned} support(A \rightarrow C) &= \mathbf{Pr}[(A \cup C)] \geq \mathbf{Pr}[(A \cup \{b\} \cup C)] = \mathbf{Pr}[(A \cup B)] \\ &= support(A \rightarrow B) \geq s. \end{aligned}$$

Similarly, using 2., we have:

$$\begin{aligned} \textit{confidence}(A{\rightarrow}C) &= \mathbf{Pr}[(A \cup C)]/\mathbf{Pr}[A] \geq \mathbf{Pr}[(A \cup \{b\} \cup C)]/\mathbf{Pr}[A] \\ &= \textit{confidence}(A{\rightarrow}B) \geq c. \end{aligned}$$

Thus, we have shown that $A{\rightarrow}C \in \Omega$, since we have that $\textit{support}(A{\rightarrow}C) \geq s$, and that $\textit{confidence}(A{\rightarrow}C) \geq c$, occur thereby proving our claim that $A{\rightarrow}C$ was also mined, that is, $A{\rightarrow}C \in \Omega$. ■

Using this claim, we show below that the elimination we perform according to each user classification of the best candidate only brings about the removal of rules that are *not*-interesting to the user who made the classifications. In other words, we show that we do not remove rules that could be potentially interesting to the users, and still are able to eliminate a significant portion of the rules.

Not-Always-True-Not-Interesting Classification of a Rule. Labeling the rule $r = a{\rightarrow}b$ as "NOT-ALWAYS-TRUE-NOT-INTERESTING" implies (Section 4.3.3) that: (1) r can be deleted from Ω, and (2) any rule $\rho = A{\rightarrow}B \in \Omega$ such that $a \in A$ and $b \in B$ can be deleted from Ω. In other words, we perform $\Omega = \Omega \setminus \textit{family}_{\Omega}(r)$.

Note that we did not eliminate rules of the form $a{\rightarrow}c$ for $c \neq b$ as they are not in $\textit{family}_{\Omega}(r)$. We may have eliminated rules of the form $a{\rightarrow}C$ where $b, c \in C$ if $a{\rightarrow}C \in \Omega$. Since the rule $a{\rightarrow}b$ is not always true, the rule $a{\rightarrow}C$ contains as much interesting information to a user as the rule $a{\rightarrow}C \setminus \{b\}$, which according to Claim 4, exists in the original Ω. Thus, this elimination does not discard any potentially interesting rules to the users.

Not-Always-True-Interesting Classification of a Rule. Labeling $r = a{\rightarrow}b$ as "NOT-ALWAYS-TRUE-INTERESTING" indicates (Section 4.3.2) that the rule is not always true and at the same time expresses a user's interest in r's family. A more detailed analysis of the ancestor rule's family is therefore required. Let:

$$\Upsilon_1 = \{a{\rightarrow}B | b \in B\}, \text{ and,} \tag{4.6}$$

$$\Upsilon_2 = \{A{\rightarrow}B | a \in A, b \in B : A \setminus \{a\}, B \setminus \{b\} \neq \emptyset\} \tag{4.7}$$

so that $\textit{family}_{\Omega}(r) = \Upsilon_1 \cup \Upsilon_2$. We analyze each of the two subgroups separately.

The rules in Υ_1 are characterized by a common, single-attributed assumption, a, and a consequent that contains the single attribute forming the ancestor rule's consequent. From the user's classification of r we know that the rule is not always true, implying that all the rules in Υ_1 are not always true, either, and can be eliminated. Formally put, we perform $\Omega = \Omega \setminus \Upsilon_1$. Note that this elimination also rids us of the ancestor rule itself (for $B = \{b\}$), if $r \in \Omega$.

The rules in the complimentary subgroup, Υ_2, are characterized by additional constraints in the assumption, which can convert the invalid rule, r, into a valid one. For example, while the rule $\langle$`male`$\rangle \rightarrow \langle$`husband`$\rangle$ may not be valid, adding the constraint `married` will turn the rule into a valid one in $\langle$`male` $\wedge$ `married`$\rangle \rightarrow \langle$`husband`$\rangle$. The rules with additional constraints in their assumption may be true, and since a user expressed interest in them, they cannot be deleted. We therefore take no action with regard to Υ_2. Note that rules of the form $A \rightarrow b$ where $a \in A$ are in Υ_2.

To summarize, a user classification of r as "NOT-ALWAYS-TRUE-INTERESTING" will be followed by performing $\Omega = \Omega \setminus \Upsilon_1$, ensuring that the algorithm does not eliminate any potentially interesting rules.

True-Not-Interesting Classification of a Rule. When a user classifies the best candidate r to be "TRUE-NOT-INTERESTING", it indicates (Section 4.3.1) that this user is not interested in associations between the assumption a and the consequent b, which is the family of r. This implies (again, using Claim 4 in the same way we did for the classification of a rule as "NOT-ALWAYS-TRUE-NOT-INTERESTING" above) that we can eliminate all the rules in the ancestor rule's family, that is, we perform $\Omega = \Omega \setminus \mathit{family}_\Omega(r)$.

We note that the difference between the classifications "NOT-ALWAYS-TRUE-NOT-INTERESTING" and "TRUE-NOT-INTERESTING" is not in the processing of Ω, but in the way we incorporate the information gained from the rule classification to our domain knowledge base.

True-Interesting or Unqualified-Interesting Classification of a rule. The goal of the algorithm we present in this section is to quickly remove not-interesting rules from Ω. It is therefore out of scope in this discussion to elaborate on the processing of rules classified as interesting

(qualified or otherwise). Rules that are classified as "UNQUALIFIED-INTERESTING" or "TRUE-INTERESTING" are not eliminated from the list of rules we process, but are not presented more than once for user classification. As a digression, we note that once a user classifies a rule as "UNQUALIFIED-INTERESTING" or "TRUE-INTERESTING" the user has the option of being presented with related candidates. We select these candidates from the family of the classified rule or from related families. Related rule families have an ancestor that shares the assumption or consequent.

For examples of how a user is expected to interact with this algorithm, see Appendix C.

4.5 Experiments With Real Data

However beautiful the strategy, you should occasionally look at the results.

—Winston Churchill

We ran our algorithm on three real databases, the WWW Proxy Logs Database the Grocery Database and the Adult Database that have different characteristics, for example, a widely varying number of attributes. These three databases are described in Section A.1. Representative results are provided below. We expected (Section 4.1.2) most of the rules to be labeled as not-interesting: "NOT-ALWAYS-TRUE-NOT-INTERESTING" or "TRUE-NOT-INTERESTING". This assumption was verified by our experiments below, resulting in more than a fifty percent decrease in the size of the problem within a few iterations.

4.5.1 Results Over the WWW Proxy Logs Database

The first database we used was the WWW Proxy Logs Database, described in Section A.1.2. We ran the association rule mining algorithm (Section 2.2) on this database with 6%, 5% and 3.5% support and confidence thresholds, to test our algorithm's performance on varying sizes of rule lists. These runs yielded 9,206, 13,644 and 29,910 rules respectively.

After three iterations of the algorithm about a third (33%) of the rules were eliminated. Approximately half the rules were eliminated after the

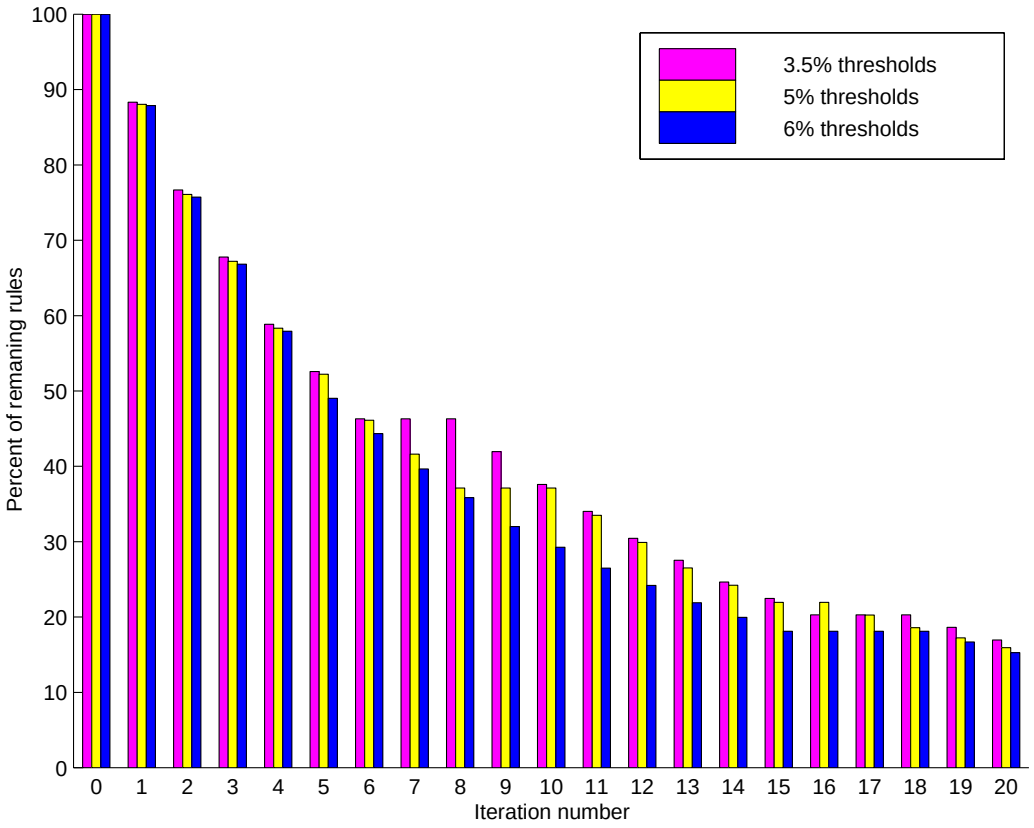

Figure 4.2: **Results of running the Interestingness Via What Is Not Interesting algorithm on the WWW Proxy Logs Database.**

fifth iteration. After ten iterations only about a third of the rules originally outputted by the data-mining algorithm remained as potentially interesting, with well under a third of the rules in one of the cases (the 6% mining thresholds). After the 20th iteration *only* approximately 16% of the rules remained, showing the extremely fast elimination rate of the algorithm.

A summary of the number of rules remaining after each of the first 20 iterations can be found in Figure 4.2, where after the zeroth iteration, that is, before the first iteration, we have 100% of the mined rules remaining. Note that in some cases there is no decrease in the number of remaining rules between two iterations. Those are the iterations the domain expert classified the best candidate as "UNQUALIFIED-INTERESTING" (there were no "TRUE-INTERESTING" classifications). Most of the classifications the domain expert provided were "TRUE-NOT-INTERESTING", indicating that the expert was familiar with the patterns the ancestor rules described and knew them to be true.

4.5.2 Results Over the Grocery Database

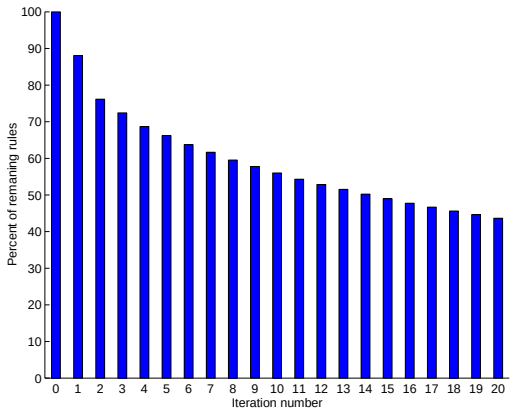

Figure 4.3: **Results of running the Interestingness Via What Is Not Interesting algorithm on the Grocery Database.**

The Grocery Database, described in Section A.1.1, has 1,757 boolean attributes. With so many different attributes in this sparse database, we expected the mined rules to be more sparse, that is, that the ancestor rules

would spawn smaller families. The outcome of the experiment supported our hypothesis. We found that only the first two ancestor rules had large families (with $RSCL$ value of over 10%). This brought about a more subdued decrease in the size of the list of mined rules, decreasing by approximately 30% after the third iteration, but reaching half its original size only after the fourteenth iteration. Since we always provide the $RSCL_{\Omega}(r)$ value of each ancestor rule we present to users for classification, users choose to stop the iterative process when the $RSCL_{\Omega}(r)$ value falls below the users' specific profitability value. The profit of a classification is the number, or percentage, of rules eliminated. The cost is the effort of making a single classification.

The results of the first 20 iterations of the algorithms run over the set of 3,046 rules mined with support and confidence levels of 3.5% are summarized in the histogram of Figure 4.3. Practically all the rules we presented for classification were classified as "TRUE-NOT-INTERESTING" (most of the rules illustrated relationships between common salad ingredients in Israel), reinforcing our comments and hypotheses in Section 4.1.2, that most of the mined rules are not interesting to the user of the KDD process.

4.5.3 Results Over the Adult Database

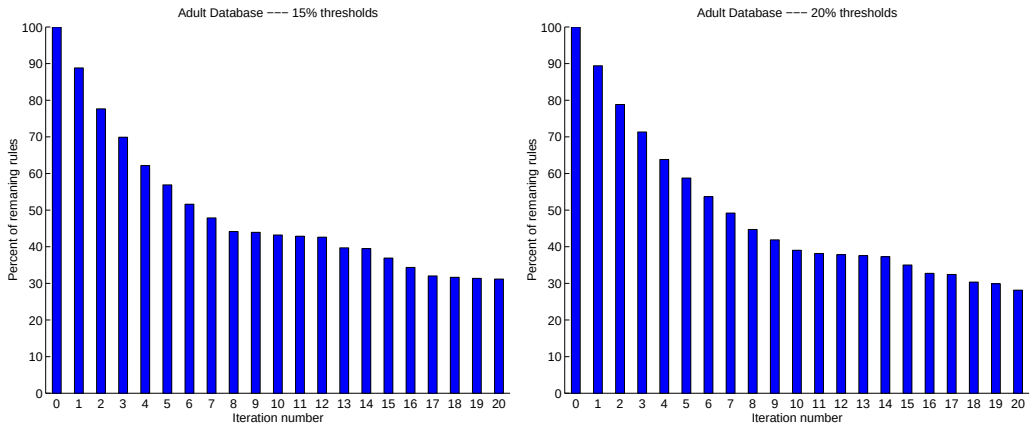

Figure 4.4: **Results of running the Interestingness Via What Is Not Interesting algorithm on the Adult Database.**

The third database we used was the Adult Database, which we describe in Section A.1.3. We mined for association rules using support and threshold

values of 20% and 15%, yielding 13,906 and 31,950 rules respectively. The histograms in Figure 4.4 present a summary of the results of the first 20 iterations of the algorithm.

The Adult Database was compiled to predict annual income exceeding $50,000 based on census data. We gave this task to our domain expert who classified most of the rules as "NOT-ALWAYS-TRUE-NOT-INTERESTING", indicating aberrations and peculiarities found in the data, and "TRUE-NOT-INTERESTING" indicating that the expert was familiar with the patterns the ancestor rules described, and knew them to be true. Again, there were no "TRUE-INTERESTING" classifications, though there were several "NOT-ALWAYS-TRUE-INTERESTING".

4.6 Summary

Subjective interestingness is a critical component of the interestingness solution since what is interesting is ultimately subjective. The Interestingness Via What Is Not Interesting algorithm presented in this chapter, based on [Sah99], has generated significant interest. It is the first step towards a comprehensive tool, suitable for any domain, that discovers the subjectively interesting rules from the exhaustive list of mined rules. The algorithm uses user classifications of ancestor rules (1) to quickly eliminate the not-interesting rules, and (2) to start constructing the domain knowledge base. A user benefits most from the iterative elimination process when the $RSCL$ value of the best candidate remains high, indicating a large family may be potentially processed. The $RSCL$ value is presented to the user to allow him/her to determine when to terminate the process.

Note that theoretically, the processing suggested in the Interestingness Via What Is Not Interesting approach could be continued iteratively until all the not-interesting rules have been eliminated. However, it is our recommendation that this approach will only be used through its first few iterations. Those iterations are very powerful as they eliminate the majority of the rules that are not interesting extremely quickly and with very little effort on the user's behalf. The $RSCL$ measure is provided to the user so that the user will be able to understand the potential impact of the next iteration, and with this information decide whether or not to execute it. Later iterations in the process are necessarily less (or at most, as) powerful. This algorithm should therefore be used as a first step to determine subjective interestingness rather

as the only step.

To determine what is subjectively interesting, users' domain knowledge needs to be incorporated into the interestingness solution. The minimalistic approach we presented in this work is differentiated from other subjective interestingness approaches in that it requires very little domain knowledge and is applicable to all domains. Other subjective interestingness approaches tend to need large amounts of domain knowledge to be successful. In a few domains it is possible to incorporate the required domain knowledge from pre-existing knowledge bases. The novelty of text mined rules is estimated using a predefined lexical knowledge-base of English words in [BMPG01], where the degree of dissimilarity between the assumption and the consequent of a rule, assessed using the English-language knowledge based, indicates novelty. In [LMY01] unexpected information is discovered by comparing new (competitor) websites to the user's own websites (which contains the user expectations). [CTS00] capitalizes on the web's structure, content and usage to determine interestingness. [LMY01, BMPG01, CTS00] can use pre-existing knowledge bases to supply the needed information instead of acquiring it directly from the domain expert. A domain where a knowledge base exists naturally is when detecting rule changes over time, as in [LHM01a].

The Interestingness Via What Is Not Interesting approach is very effective in reducing the number of subjectively not-interesting rules, and yet it requires very low-intensity (in both volume and level) user interaction. We only present a few very simple classification questions of ancestor rules, asking the user to label them as (not-)true and as (not-)interesting. We saw that with these simple questions we can quickly eliminate many rules, yielding a dramatic decrease in the size of the original list outputted by the data-mining algorithm—as much as half its size after only five classification questions. Since size is the essence of the problem, this first step provides sound grounds for continued subjective interestingness research. In the next chapter (based on [Sah02b]) we investigate how to incorporate the Interestingness Via What Is Not Interesting approach into the mining process. In the following chapter, Chapter 6 (based on [Sah02a]), we further investigate and generalize on this approach, by creating general groupings of rules according to similarity to facilitate the exploration and understanding of interestingness.

Chapter 5

On Incorporating Subjective Interestingness Into the Mining Process*

And if you find her poor—Ithaca will not have deceived you.
Wise as you will have become, so full of experience,
You will have understood by then what these Ithacas mean.

—Constantine P. Cavafy, last three lines of Ithaca

Subjective interestingness is at the heart of the successful mining of association rules; it is critical in determining which of the mined rules are interesting to users to successfully meet their expectations. In this chapter we investigate how the Interestingness Via What Is Not Interesting (presented in Chapter 4) can be incorporated into the mining process, the benefits and disadvantages of doing so, and examine the results of its application to real databases. Remember that the Interestingness Via What Is Not Interesting approach requires very low level domain knowledge and interaction in order to significantly eliminate the majority of the rules that are subjectively not interesting.

*A preliminary version of this chapter was accepted for publication in the IEEE International Conference on Data Mining (ICDM-2002) [Sah02b].

5.1 Introduction

The beginning of knowledge is the discovery of something we do not understand.

—Frank Herbert

Since interestingness is ultimately subjective, to completely resolve the interestingness problem subjective interestingness criteria will have to be applied.

Incorporating subjective interestingness criteria into the mining process has obvious advantages: fewer patterns will be outputted, and those patterns are more likely to be interesting to users of the KDD process. However, a potential drawback of this tactic is that the mined results may be tailored to match only the pre-specified subjective criteria. Exploring slightly different interestingness criteria could then require re-executing the entire, normally lengthy, mining process. Additionally, in the general case, the required subjective interestingness criteria are not available in a pre-existing knowledge base or readily obtained from a domain expert.

To reduce the number of mined patterns to make their post-processing easier, we need a method that incorporates *general* subjective interestingness criteria into the mining process that:

- are very easy to define, and,
- apply to the interests of a wide audience base.

5.1.1 Methods for Using Domain Knowledge and Related Works

There are three general approaches in the KDD literature to employing domain knowledge to determine the subjective interestingness of patterns.

1. In the first approach the prior knowledge and user needs are obtained by explicitly asking domain experts to express all the required information in a predefined grammar. The grammars used vary greatly in their formalism and the degrees of uncertainty they permit. This approach is currently the most popular method for employing subjective interestingness in the KDD literature. It is likely to yield the short list of potentially interesting rules, but its success depends on the availability of large amounts of domain knowledge, provided either by domain

experts or accessible from pre-existing knowledge bases. Works that fall under this approach are reviewed in Sections 4.1.1 and 4.6.

2. The second approach, taken in [Sub98], has users, not necessarily experts, classify all the mined rules. The information gathered is then used to classify new rules. We discuss this approach in Section 4.1.1.

3. The third approach, the Interestingness Via What Is Not Interesting approach, introduced in Chapter 4, is to eliminate from the list of mined rules the majority of the un-interesting rules by having a naive user classify only a few simple rules. The output of this technique is a superset of the list of interesting associations.

The first approach has been previously integrated into the mining process, but cannot be used for the general goal of incorporating interestingness into the mining process since it requires extensive amounts of domain knowledge to be successful (extensive domain knowledge is, in the general case, difficult to acquire and is user-dependent). That means that neither one of two requirements listed in the beginning of the introduction are satisfied by this approach: the criteria are *not* easy to define and they do *not* apply to the interests of a wide audience base. The second approach acquires the large amount of domain knowledge it requires in a different way, but still requires a large amount of domain knowledge. The third approach only uses very little domain knowledge, that is assumed to be known even to naive users, to eliminate a significant portion of the not-interesting rules, and thus presents a very promising first step towards incorporating subjective interestingness. We follow this approach.

Additional approaches that incorporate interestingness criteria into the mining process beyond those that fall under the methods above are [Zak00] who mined a summary of non-redundant association rules from which all other rules can be inferred, [BJAG99] who introduced a novel user-specified minimum confidence improvement threshold, and [PH00] who integrated convertible constraints. These approaches use objective interestingness criteria reviewed and differentiated from subjective criteria in Chapters 3 and 7.

5.1.2 Our Approach

In this chapter we incorporate the methodology of the third approach, the Interestingness Via What Is Not Interesting approach (Chapter 4), into the

mining to create a simple, general and effective mining algorithm. This algorithm only requires responses to a few, simple classifications in order to eliminate a significant portion of the association rules that are not-interesting to a wide audience base. We also examine the results and repercussions, both benefits and disadvantages, of the application of this method on the same databases used in Chapter 4.

Note that as in the Interestingness Via What Is Not Interesting algorithm of Chapter 4 our goal in incorporating the simple classifications into the mining process is to reduce the number of not-interesting associations mined rather than to completely eliminate them. The output of this mining process will be a reduced set of rules, with fewer not-interesting rules than would have been outputted otherwise. This output will be post-processed to be tailored to the unique subjective interestingness requirements of a specific user using further subjective criteria.

The rest of the chapter is organized as follows: in Section 5.2 we define binuclear-families and ancestor itemsets. We introduce the algorithm that stems from incorporating the Interestingness Via What Is Not Interesting approach into the mining in Section 5.3. In Section 5.4 we present and discuss the results of applying the algorithm to real databases. We end with our conclusions in Section 5.5.

5.2 Definitions and Notations

If you want to make an apple pie from scratch, you must first create the universe.

—Carl Sagan

We extend the definitions of a family and of an ancestor rule from Section 4.2.

Definition 5 *Let Ω be the list of association rules mined over Λ. Let $a, b \in \Lambda$. We define the* ***binuclear family*** *of a given 2-itemset $\{a, b\}$ to be:*

$$\textbf{\textit{binuclear-family}}_{\Omega}(\{a,b\}) \stackrel{\textbf{Def}}{=} \textit{family}_{\Omega}(a \rightarrow b) \bigcup \textit{family}_{\Omega}(b \rightarrow a), \qquad (5.1)$$

where family was defined in Equation 4.2: $\textit{family}_{\Omega}(x \rightarrow y) = \{X \rightarrow Y \in \Omega | x \in X, y \in Y\}$.

$\{a, b\}$ is referred to as the **ancestor itemset** and is said to **span** the *binuclear-family*$_\Omega(\{a, b\})$. As was the case with ancestor rules, $a \rightarrow b$ and $b \rightarrow a$ may or may not be in Ω.

A binuclear-family is also be defined to be spanned by an ancestor rule:

$$\textit{binuclear-family}_\Omega(a \rightarrow b) \stackrel{\textbf{Def}}{=} \textit{binuclear-family}_\Omega(\{a, b\}). \tag{5.2}$$

Note that *binuclear-family*$_\Omega(a \rightarrow b)$ = *binuclear-family*$_\Omega(b \rightarrow a)$.

5.3 The Algorithm

You have to know the past to understand the present.

—Carl Sagan

5.3.1 Ancestor Itemsets: Motivation and Challenge

In this chapter we introduce an algorithm that incorporates into the mining process the type of subjective criteria used for post-processing interestingness in the Interestingness Via What Is Not Interesting algorithm (see Chapter 4). The goals of this algorithm are:

1. **Simplicity**: keep the process simple by asking users only a few, easy classification questions and circumvent the need to formally define what is interesting.

2. **Generality**: the results of the mining process with the integrated subjective interestingness will be general enough to be applicable to a wide user base for further interestingness post-processing.

3. **Effectiveness**: the integration of this type of subjective interestingness into the mining process will result in the mining of much fewer association rules. For example, incorporating the first classification into the mining process resulted, on average, in the mining of fewer than 80% of the rules that would have been mined otherwise (see Section 5.4 for details).

The Interestingness Via What Is Not Interesting approach uses domain knowledge of the type *"not interested in family*$_\Omega(\rho)$*"* for interestingness post-processing (Section 4.3). This type of domain knowledge is particularly well

suited to accomplish our three goals since it is simple and yields effective results. The families of the two ancestor rules $\rho = a \rightarrow b$ and $\tilde{\rho} = b \rightarrow a$, $family_{\Omega}(a \rightarrow b)$ and $family_{\Omega}(b \rightarrow a)$, cannot be distinguished during the mining process, when only itemsets are recognized. In order to incorporate the type of user preferences used in the Interestingness Via What Is Not Interesting approach into the mining process, we have to use generalized domain knowledge of the kind "*not interested in binuclear-family*$_{\Omega}(\rho)$".

One of the most important advantages that the generalization of a family of an ancestor rule to a binuclear-family maintains is the simple, easy classification of the ancestor itemset, the extension of the ancestor rule, as in Example 5.3.1, avoiding to the need to formally define what is interesting. We show how we guarantee the three goals of simplicity, generality and effectiveness in Section 5.3.2.

Example 5.3.1 *Users familiar with the making of Israeli salads, whose two main ingredients are tomatoes and cucumbers, will not be interested in rules that show the association between customers who purchase tomatoes and customers who purchase cucumbers and vice versa. This is equivalent to classifying* $I_{ct} = \{\texttt{cucumbers,tomatoes}\}$*, and hence, its binuclear-family, as not interesting.*

We can decrease the number of mined patterns by eliminating one or more of the frequent itemsets discovered during the mining process. This, however, does *not* imply that the frequent itemset $I_{ct} = \{\texttt{cucumbers,tomatoes}\}$ in Example 5.3.1 can be eliminated; $r = \langle \texttt{cucumbers} \wedge \texttt{tomatoes} \rangle \rightarrow \langle \texttt{onions} \rangle$ may be interesting to some users. r is derived from itemset $I_{\texttt{cto}} = \{\texttt{cucumbers}, \texttt{tomatoes}, \texttt{onions}\}$. I_{cto} will be generated as a frequent itemset during the mining process only if all its 2-itemset subsets are frequent (large). This is illustrated in Figure 2.1. Thus, if I_{ct} is deleted, $I_{\texttt{cto}}$ will not be generated as a frequent itemset and the potentially interesting rule r will never be discovered.

The challenge in deleting a frequent itemset $I = \{x, y\}$ is in ensuring that even though I is removed, rules of the type:

$$X \rightarrow Z \text{ and } Z \rightarrow X : x, y \in X \text{ and } x, y \notin Z \tag{5.3}$$

that would have been mined had I not been removed, will still be mined, while maintaining the simplicity, generality and effectiveness of the algorithm.

5.3.2 Ancestor Itemsets: Selection and Classification

In the Interestingness Via What Is Not Interesting approach (Chapter 4), two dimensions are addressed in each ancestor rule classification by a user: (1) is the ancestor rule true/not always true, and, (2) is the user subjectively interested/not-interested in the family spanned by the ancestor rule. In this algorithm we depart from this two dimensional classification, and only ask a single-dimensioned classification question: is the ancestor *itemset* subjectively interesting/not-interesting to the user. In other words, we ask whether a user is interested in any rule of the binuclear-family spanned by the ancestor itemset. If a user classifies an ancestor itemset as not-interesting, it is marked for elimination (used in Section 5.3.3). If a user classifies an ancestor itemset as interesting, it will not be eliminated.

We select for user classification the 1–3 most frequent ancestor 2-itemsets in order of their frequency. In this way, the most frequent 2-itemset in the database is the first ancestor itemset to be brought for user classification, followed by the next most frequent 2-itemset, etc.

These limited selection and classification processes, of the 1–3 most frequent 2-itemsets, naturally guarantees our algorithms' three goals. $\mathcal{B} = \textit{binuclear-family}_{\Omega}(\{a, b\})$ is easy to classify because it consists simply of $\textit{family}_{\Omega}(a{\rightarrow}b) \cup \textit{family}_{\Omega}(b{\rightarrow}a)$, and if $a, b \in X$ then $X{\rightarrow}Y, Y{\rightarrow}X \notin \mathcal{B}$. For example, to classify the ancestor itemset $I_{ct} = \{\texttt{cucumbers},\texttt{tomatoes}\}$ in Example 5.3.1, the most frequent itemset in the Grocery Database, *only* relationships between customers who buy cucumbers and customers who purchase tomatoes need to be considered, without needing to consider all shopping baskets that include cucumbers and tomatoes. Classifying an ancestor itemset is equivalent to classifying two ancestor *rules* as interesting/not-interesting, two easy classifications. We limit the ancestor itemsets to be 2-itemsets to make sure that this classification will be easy. Larger itemsets, itemsets with more than two attributes, are considered too complicated for the naive classification we want to preserve[†]. The other reason the classi-

[†]If we extended the definition of an ancestor itemset to a 3-itemset, $\{a, b, c\}$, its binuclear-family would be the union of the "extended" families of $a{\rightarrow}bc$, $ab{\rightarrow}c$, $ac{\rightarrow}b$, $b{\rightarrow}ac$, $bc{\rightarrow}a$, and $c{\rightarrow}ab$, and possibly, depending on the extended definition, also of: $a{\rightarrow}b$, $a{\rightarrow}c$, $b{\rightarrow}a$, $b{\rightarrow}c$, $c{\rightarrow}a$, and $c{\rightarrow}b$. Classifying 6–12 "families" in order to classify one single "binuclear-family" spanned by a 3-itemset is not a simple process. The number of such classification will, of course, increase with larger potential ancestor itemsets. We therefore restrict the definition and use of ancestor itemsets to be limited to 2-itemsets. We assume that in the general interestingness case users are not able to determine a priori which

fication is simple is that only the most frequent 2-itemsets are brought for classification. The most frequent 2-itemsets describe the most common relationships in the database, likely to be known to even the most naive users, and therefore easy to classify, thus achieving our *simplicity* goal. Since these most frequent 1–3 2-itemsets describe relationships that naive users are probably familiar with, a wide audience base is very likely to classify them similarly, also yielding *generality*. For example, I_{ct} in Example 5.3.1, is a relationship known even outside the grocery-marketing industry. The more classified ancestor itemsets (subjective domain knowledge) are used, the higher the likelihood of tailoring the output of the mining process to the interests of only a specific audience group.

Finally, since the most frequent 1–3 itemsets are likely to have very large binuclear-families, likely to be classified as not-interesting (since a wide audience base is familiar with them), the limit on the number of ancestor itemsets also yields *effectiveness*. Eliminating any one of the most frequent itemsets is likely to be have a major impact on the number of mined rules, since these itemsets tend to be included in larger frequent itemsets and, therefore, in more rules that will be not be mined. For example, when mining the Grocery Database with 6% thresholds, deleting the most frequent itemset in the database, I_{ct} = {`cucumbers`,`tomatoes`}, yielded an output of only 494 mined association rules, approximately 26.5% *less* rules than when no ancestor itemsets were deleted during the mining process. If we delete the 2-itemset {`bananas`,`summer squash`}, that has approximately 7% support, only 2 less rules will be mined. Remember that in the Interestingness Via What Is Not Interesting approach, where subjective interestingness is incorporated iteratively, as a post-processing step, prior to each classification, users can rely on $RSCL$ (Section 4.2.2) for an indication of the potential impact of the next classification they make. This is an important indication since the impact of classifying ancestor rules decreases with the number of ancestor rules processed (Section 4.5). In this case, when interestingness is incorporated in one step, during the mining process, this kind of indication is not possible.

We end this section with some implementation notes on the ancestor itemset selection process. There are several ways to implement this selection. The most naive way is to run the first two iterations of the Apriori-TID mining

single attributes are interesting to them. Such constraints, if they are available from the user, can be introduced as in [SVA97].

(1) Υ = initialize list of ancestor itemsets;
(2) $\mathcal{I}_\Upsilon$ =`new-attributes`$(\Upsilon, \mathcal{I})$; $\Lambda_\mathcal{I} = \Lambda \cup \mathcal{I}_\Upsilon$;
(3) L_1 = frequent 1-itemsets over $\Lambda_\mathcal{I}$;
$\widehat{C}_1$ = *database* $\mathcal{DB}_\mathcal{I}$ ($\mathcal{DB}$ over $\Lambda_\mathcal{I}$);
(4) $C_2 = \{\{a, b\} | a, b \in L_1, a \neq b\}$;
/* Generation of frequent 2-itemsets candidates */
(5) `determine-support`$(C_2, \widehat{C}_2, \widehat{C}_1)$;
(6) `delete-subjectively-not-interesting`(C_2);

Figure 5.1: **Part I of algorithm for incorporating subjective interestingness into the mining process**.

(Section 2.2), sort the 2-itemsets according to their respective support, and select the itemsets with the largest support. We have found that a more efficient process is to mine the frequent 1-itemsets from a uniformly selected sample set, the size of a quarter of the original database, using 25% support threshold. Then, produce *all* the 2-itemsets and from those select the 1–3 itemsets with the highest support. The number of 2-itemsets mined using this process tends to be relatively small. This is a short process, yielding the same results as the naive one. For the three databases used in our experiments, it took from a few seconds to a couple of minutes.

5.3.3 The Mining Process

The domain knowledge available to us is simple: a short list of pairs of the kind $\langle I, c \rangle$, where I is a 2-itemset selected as in Section 5.3.2 and c is its classification, "INTERESTING" or "NOT-INTERESTING". For $I = \{a, b\}$ classified as NOT-INTERESTING we cannot simply limit the search space of frequent itemsets by deleting the itemset $\{a, b\}$, as explained in Section 5.3.1. What we can do is alter the search space to allow us to take advantage of this information. Figure 5.1 outlines the first part of the algorithm we introduce to accomplish this.

In line (1) of Figure 5.1 we initialize the list of ancestor itemsets classified for deletion, Υ. We select the ancestor itemsets as in Section 5.3.2, present them for user classification, and initialize Υ to be the ancestor itemsets classified as NOT-INTERESTING. In line (2) of Figure 5.1 we translate Υ into

new attributes, $\mathcal{I}$, expand the set of attributes, Λ, to $\Lambda_{\mathcal{I}}$, and expand the database into $\mathcal{DB}_{\mathcal{I}}$. We describe this process in the Creating New Attributes for the Mining Process subsection below. Lines (3)–(5) of Figure 5.1 cover the construction of L_1, $\widehat{C}_1$, C_2 and $\widehat{C}_2$, detailed below. Remember that we use the same notations L_k, C_k and $\widehat{C}_k$ introduced in [AHS+96] and reviewed in Section 2.2.1. In line (6), we eliminate a few of the frequent itemsets, as explained below. This concludes the Part I of the algorithm. The second part, consisting of iterations for $k \geq 3$, is then executed as in Apriori-TID, described in Section 2.2. The rule generation completes the mining process, and it is performed according to `ap-genrules` in [AHS+96].

Creating New Attributes for the Mining Process

The list of ancestor itemsets classified for deletion, Υ, contains one or more elements, each of the type $\{a_i, b_i\}$ where $a_i, b_i \in \Lambda$. If during the mining process $\{a_i, b_i\}$ is generated as a frequent itemset, it will be deleted. To ensure that our altered mining algorithm will still output the type of rules in Equation 5.3 that would have been mined if the itemset $\{a_i, b_i\}$ was *not* deleted, we introduce new attributes. We start with the general definition.

Given $X = \{x_1, \ldots, x_n\}$, we can create a new attribute υ_X by defining for every transaction, τ in the database:

$$\tau(\upsilon_X) \stackrel{\textbf{Def}}{=} \begin{cases} \text{TRUE} & \text{if } \forall x_i \in X, \tau(x_i) = \text{TRUE}, \\ \text{FALSE} & \text{otherwise.} \end{cases} \tag{5.4}$$

Now, for each $\{a_i, b_i\} \in \Upsilon$, we create a new attribute $\upsilon_{\{a_i,b_i\}}$ in $\mathcal{I}_\Upsilon$ defined by Equation 5.4 for $X = \{a_i, b_i\}$, $n = 2$, that is:

$$\tau(\upsilon_{\{a_i,b_i\}}) \stackrel{\textbf{Def}}{=} \begin{cases} \text{TRUE} & \text{if } (\tau(a_i) = \text{TRUE}) \wedge (\tau(b_i) = \text{TRUE}), \\ \text{FALSE} & \text{otherwise.} \end{cases}$$

Additionally, for any $\{a_i, b_i\}, \{a_j, b_j\} \in \Upsilon$ where $\{a_i, b_i\} \cap \{a_j, b_j\} \neq \emptyset$, we define an additional attribute in $\mathcal{I}_\Upsilon$. Without loss of generality we can assume that if $\{a_i, b_i\} \cap \{a_j, b_j\} \neq \emptyset$, then $a_i = b_j$, and then we define and additional attribute in $\mathcal{I}_\Upsilon$: $\upsilon_{\{a_i,b_i,a_j\}}$ defined using Equation 5.4 for $X = \{a_i, b_i, a_j\}$, $n = 3$.

We now define the extended attribute set: $\Lambda_{\mathcal{I}} \stackrel{\textbf{Def}}{=} \Lambda \cup \mathcal{I}_\Upsilon$, where $\mathcal{I}_\Upsilon = \{\upsilon_i\}_i$ is the list of new attributes created for each element in Υ and for their intersections. We also define $\mathcal{DB}_{\mathcal{I}}$ to be the extension of $\mathcal{DB}$ spanned by $\Lambda_{\mathcal{I}}$

and defined by Equation 5.4. We will refer to $\Lambda_{\mathcal{I}}$ as the expanded attribute set and to $\mathcal{DB}_{\mathcal{I}}$ as the expanded database.

We conclude this section by examining the creation of new attributes for two different examples: Υ_1 in Example 5.3.2, and Υ_2 in Example 5.3.3.

Example 5.3.2 *Let* $\Upsilon_1 = \{\{\texttt{tomatoes,cucumbers}\}, \{\texttt{dill,parsley}\}\}$ *be the list of ancestor itemsets marked for deletion. In this case, two new attributes will be created:* $v_{\{\texttt{tomatoes,cucumbers}\}}$ *which has the value* TRUE *for all transactions where both tomatoes and cucumbers have been purchased, and* $v_{\{\texttt{dill,parsley}\}}$ *which has the value* TRUE *for all transactions where both dill and parsley have been purchased.*

Example 5.3.3 *Let* $\Upsilon_2 = \{\ \{\texttt{tomatoes,cucumbers}\}, \{\texttt{tomatoes, onions}\}\}$. *In this case, the two ancestor itemsets classified for deletion are not disjoint, but have a common attribute:* `tomatoes`. *To support this case,* three *new attributes will be created:* $\mathcal{I}_\Upsilon = \{v_{\{\texttt{tomatoes,cucumbers}\}}, v_{\{\texttt{tomatoes,onions}\}}, v_{\{\texttt{tomatoes,cucumbers,onions}\}}\}$. *The first new attribute,* $v_{\{\texttt{tomatoes,cucumbers}\}}$, *has the value* TRUE *in transactions where both tomatoes and cucumbers have been purchased,* $v_{\{\texttt{tomatoes,onions}\}}$ *has the value* TRUE *in transactions where both tomatoes and onions have been purchased, and* $v_{\{\texttt{tomatoes,cucumbers,onions}\}}$ *has the value* TRUE *in transactions where tomatoes, cucumbers and onions have been purchased. The need for the third attribute will be clarified in the deletion process.*

Starting the Modified Mining Process

$\Lambda_{\mathcal{I}}$ is the set of candidate frequent 1-itemsets, since every 1-itemset is a potentially frequent 1-itemset. In line (3) of Figure 5.1 we initialize L_1, the set of frequent 1-itemsets over $\mathcal{DB}_{\mathcal{I}}$ with a single pass over $\mathcal{DB}_{\mathcal{I}}$. Since the ancestor itemsets in Υ are the most frequent 2-itemsets, almost all mining processes (with exception of ones with extremely high support thresholds that yield very few to no rules) will discover all the new attributes, $\mathcal{I}_\Upsilon$, as frequent 1-itemsets. During the same pass on $\mathcal{DB}_{\mathcal{I}}$, $\widehat{C}_1$ is constructed over $\mathcal{DB}_{\mathcal{I}}$.

The second iteration of the algorithm, for $k = 2$, starts with the generation of candidate frequent 2-itemsets on line (4) of Figure 5.1. Using the downward closure property of support (Observation 1 in Section 2.2.1) we know that all the 1-itemsets subsets of frequent 2-itemsets are also frequent.

Therefore, C_2, the set of candidate frequent 2-itemsets, is constructed from L_1.

In line (5) of Figure 5.1, $\widehat{C}_2$, the representation of the transactions that the 2-itemsets in C_2, is constructed as described in Section 2.2.1. We use $\widehat{C}_2$ to eliminate the candidate itemsets in C_2 that do not have at least the predefined minimum support threshold, s. After completing the execution of line (5), C_2 contains all the frequent 2-itemsets. Notice that C_2 and $\widehat{C}_2$ are constructed over the expended attribute set and the expanded database.

Deleting Ancestor Itemsets

In line (6) of Figure 5.1 we incorporate the domain knowledge available from the classification of the ancestor itemsets. For $\Upsilon_1 = \{\upsilon_{ab}\}$, where there is only one ancestor itemset to be deleted, $\varrho = \{a, b\}$, we delete the following itemsets:

1. $\{a, b\}$, since it is the itemset classified as not interesting and marked for elimination,
2. $\{a, \upsilon_{ab}\}$, since it is equivalent to $\{a, a, b\} = \varrho$, as well as,
3. $\{b, \upsilon_{ab}\}$ which is also equivalent to ϱ.

Note that if ϱ is a frequent (large) itemset, all three of the above 2-itemsets are frequent (in C_2), and will actually be deleted from C_2. This observation will continue to hold for all the itemsets below.

For $\Upsilon_2 = \{\varrho_1, \varrho_2\}$, where $\varrho_1 = \{a, b\}$ and $\varrho_2 = \{c, d\}$, one of two cases will occur: (1) $\{a, b\} \cap \{c, d\} = \emptyset$, or, (2) $\{a, b\} \cap \{c, d\} \neq \emptyset$.

Case 1: $\{a, b\} \cap \{c, d\} = \emptyset$ then $\Upsilon_2 = \{\upsilon_{ab}, \upsilon_{cd}\}$, and we delete: **(1)** $\{a, b\}$, **(2)** $\{a, \upsilon_{ab}\}$, **(3)** $\{b, \upsilon_{ab}\}$, **(4)** $\{c, d\}$, **(5)** $\{c, \upsilon_{cd}\}$, and **(6)** $\{d, \upsilon_{cd}\}$, using the same reasoning as for Υ_1 with equivalencies to υ_{ab} and υ_{cd}.

Case 2: $\varrho_1 = \{a, b\}$, $\varrho_2 = \{a\,(= c), d\}$, then $\mathcal{I}_\Upsilon = \{\upsilon_{ab}, \upsilon_{ad}, \upsilon_{abd}\}$, and we delete the following 14 2-itemsets:

(1) and **(2)** $\{a, b\}$ and $\{a, d\}$ are deleted as these are the ancestor itemsets marked for elimination.

(3) and **(4)** $\{a, \upsilon_{ab}\}$ and $\{b, \upsilon_{ab}\}$ are deleted since these 2-itemsets are equivalent to $\{a, b\} = \upsilon_{ab}$, which we marked for elimination in (1) above.

(5) and **(6)** $\{a, \upsilon_{ad}\}$ and $\{d, \upsilon_{ad}\}$ are deleted as they are equivalent to $\{a, d\} = \upsilon_{ad}$, which we marked for elimination in (2) above.

(7)–(11), $\{a, \upsilon_{abd}\}$, $\{b, \upsilon_{abd}\}$, $\{d, \upsilon_{abd}\}$, $\{\upsilon_{ab}, \upsilon_{abd}\}$ and $\{\upsilon_{ad}, \upsilon_{abd}\}$ are deleted as these five itemsets are equivalent to $\{a, b, d\} = \upsilon_{abd}$, which we already have accounted for. And finally,

(12)–(14) $\{d, \upsilon_{ab}\}$, $\{b, \upsilon_{ad}\}$ and $\{\upsilon_{ab}, \upsilon_{ad}\}$ are deleted since these three itemsets are equivalent to $\{a, b, d\} = \upsilon_{abd}$, which we already have accounted for. Note that this set of three deletions, as well as deletions (7)–(11), are not symmetric as $\upsilon_{bd} \notin \Upsilon_2$. We briefly explain how to tackle that case, when $\upsilon_{bd} \in \Upsilon$ at the end of this section.

Case 2 clarifies the need for the introduction of a new attribute, υ_{abd}, for cases where the ancestor itemsets to be deleted are *not* mutually exclusive. We follow the example set by Υ_2 in Example 5.3.3 where $\Upsilon_2 =$ {{`tomatoes`,`cucumbers`}, {`tomatoes`,`onions`}}. Let a=`tomatoes`, b= `cucumbers`, and d=`onions` ($a = c$). As we see in Figure 2.1, for the 3-itemset I_{cto} = {`tomatoes`,`cucumbers`, `onions`} to be created, the three 2-itemsets I_{ct} ={`cucumbers`,`tomatoes`}, I_{to} ={`tomatoes`,`onions`} and I_{co} ={`cucumbers`,`onions`} need to be frequent itemsets. The two ancestor itemsets, I_{ct} and I_{to}, will be eliminated according to (1) and (2) of Case 2 above. Without the introduction of $\upsilon_{abd} = I_{cto}$ as a new attribute, I_{cto} will not be discovered as a frequent 3-itemset, and a rule such as ⟨`cucumbers`∧`tomatoes`∧`onions`⟩→ ⟨`cottage cheese 5%`⟩, that is potentially interesting to a user who classified the ancestor itemsets I_{ct} and I_{co} as not interesting, will not be mined.

Note that we do not need to eliminate any itemsets in the third iteration, when the 3-itemsets are constructed. Itemsets such as $\{a, x, \upsilon_{ab}\}$ that we would want deleted, will be automatically deleted during the pruning stage of candidate itemsets since at least one of its 2-itemsets. For example, $\{a, \upsilon_{ab}\}$, is not a member of C_2 as it has been deleted in one of the 14 deletions above. In this manner, the deletion of the itemsets incorporates the domain knowledge available to us into the mining, and the incorporation of the new attributes guarantees that association rules that are potentially interesting to users (that contain the two attributes in an ancestor itemset marked for deletion either in the assumption or consequent of the rule) will still be mined.

It is worth noting that for $\Upsilon_3 = \{\{a, b\}, \{a, d\}, \{b, d\}\}$, we will have $\mathcal{I}_\Upsilon = \{\upsilon_{ab}, \upsilon_{ad}, \upsilon_{abd}, \upsilon_{bd}\}$. In this case, the 21 2-itemset deletions will be required: the 14 deletions in Case 2 above, as well as,

(15) delete $\{b, d\}$ as this is an ancestor itemsets in Υ_3 marked for deletion.

(**16**) **and (17)** delete $\{b, \upsilon_{bd}\}$ and $\{d, \upsilon_{bd}\}$ since these two 2-itemsets are equivalent to $\{b, d\} = \upsilon_{bd}$.

(18)–**(21)** delete $\{a, \upsilon_{bd}\}$, $\{\upsilon_{ab}, \upsilon_{bd}\}$, $\{\upsilon_{ad}, \upsilon_{bd}\}$ and $\{\upsilon_{abd}, \upsilon_{bd}\}$ since these four itemsets are equivalent to $\{a, b, d\} = \upsilon_{abd}$.

For completeness, we include a very simple claim that guarantees the correctness of the algorithm, where the proof highlights the arguments provided throughout the section.

Claim 5 *Let Ω be the exhaustive list of association rules mined over $\mathcal{DB}$ with support and confidence thresholds of s and c respectively. Let $\mathcal{A}$ be the data mining algorithm into which ancestor itemsets classified as not-interesting were incorporated described in Figure 5.1. Let $\widetilde{\Omega} \subseteq \Omega$ be the set of association rules mined by $\mathcal{A}$ over $\mathcal{DB}$ with the same support and confidence levels s and c. For a user who classified the ancestor itemsets as not interesting,* (1) *$\widetilde{\Omega}$ is a superset of all the interesting association rules, and,* (2) *$\widetilde{\Omega}$ contains none of the rules in the binuclear-families of the ancestor itemsets classified as not-interesting.*

Proof: The proofs for (1) and (2) follow immediately from the algorithm definition with the introduction of the new attributes corresponding to the ancestor itemsets classified for deletion. Let $\{a, b\}$ be an ancestor itemset classified as not-interesting. To show (1) we need to show that for $a, b \in C, a, b \notin D$, if $r_1 = C \rightarrow D \in \Omega$ then $r_1 \in \widetilde{\Omega}$ and if $r_2 = D \rightarrow C \in \Omega$ then $r_2 \in \widetilde{\Omega}$. This follows directly from the introduction of the new attributes corresponding to the ancestor itemsets classified for deletion. Formally, if $r_1 \in \Omega$ (or if $r_2 \in \Omega$) then $C \cup D \in L$, where L is the entire list of frequent itemsets mined over $\mathcal{DB}$. Now, $a, b \in C \cup D$ indicates that $\{a, b\} \in L_2$ (that we could have assumed since it was one of the ancestor itemsets brought for user classification, indicating it was one of the frequent 2-itemsets), which in turn means that $\{a\}, \{b\}, \upsilon_{ab} \in L_{1\mathcal{A}}$ where $L_{1\mathcal{A}}$ is the set of frequent 1-itemsets generated in $\mathcal{A}$. $\{a, b\}$ will be deleted on line (6) of the algorithm, but since $\upsilon_{ab} \in L_{1\mathcal{A}}$ and since $r_1 \in \Omega$, for $I = (C \cup D \cup \{\upsilon_{ab}\}) \setminus \{a, b\}$ we have that $I \in L_{\mathcal{A}}$ where $L_{\mathcal{A}}$ is the set of all frequent itemsets generated by $\mathcal{A}$. From I both r_1 and r_2 will be generated since $r_1, r_2 \in \Omega$ means that

confidence$(r_1) \geq c$ and *confidence*$(r_2) \geq c$, easily completing the proof for this case.

To show (2) we need to show that rules of the type $r_3 = A \rightarrow B$ and $r_4 = B \rightarrow A$ where $a \in A$, $b \in B$ that are not interesting to a user who classified $\{a, b\}$ will not be mined by $\mathcal{A}$, that is, if $r_3, r_4 \in$ *binuclear-family*$(\{a, b\})$ then $r_3, r_4 \notin \widetilde{\Omega}$. Again, this follows directly from the algorithm we introduced. A necessary condition for r_3 (or r_4) to be mined by $\mathcal{A}$ is that $I' = \{a, b\} \in L_{\mathcal{A}}$. Since I' is the itemsets that the user classified as not-interesting, it is eliminated on line (6) of the algorithm, so that $I' \notin L_{2\mathcal{A}} \subseteq L_{\mathcal{A}}$ thereby showing (2). The proof for is very similar when more than one ancestor itemset classified for deletion. Note that only within this proof we make a notational distinction between L and $L_{\mathcal{A}}$. ■

5.4 Experiments With Real Data

We ran our algorithm on the same databases used in Section 4.5 in the experiments on the Interestingness Via What Is Not Interesting post-processing approach: the Grocery Database, the WWW Proxy Logs Database and the Adult Databasedescribed in Appendix A.1. The Grocery Database was the sparsest database we used, with 1.4% average density. We mined this database with 6% and 3.5% support and confidence thresholds, yielding 672 and 3,046 rules respectively. The WWW Proxy Logs Database was the densest database of the three. We mined this database with 6%, 5% and 3.5% support and confidence thresholds, yielding 9,206, 13,644 and 29,910 rules respectively. We mined the Adult Database with support and threshold values of 20% and 15%, yielding 13,906 and 31,950 rules were outputted respectively. For each mining threshold, we mined each database using zero, one, two and three ancestor itemsets classified for deletion. Not surprisingly, the ancestor itemsets corresponded to the ancestor rules used in the Interestingness Via What Is Not Interesting. This allowed us to readily compare the results.

5.4.1 Reduction in the Number of Rules Mined

Figure 5.2 depicts a comparison of the results of mining the WWW Proxy Logs Database, the Grocery Database and the Adult Database with zero, one, two and three ancestor itemsets classified for deletion, and two mining

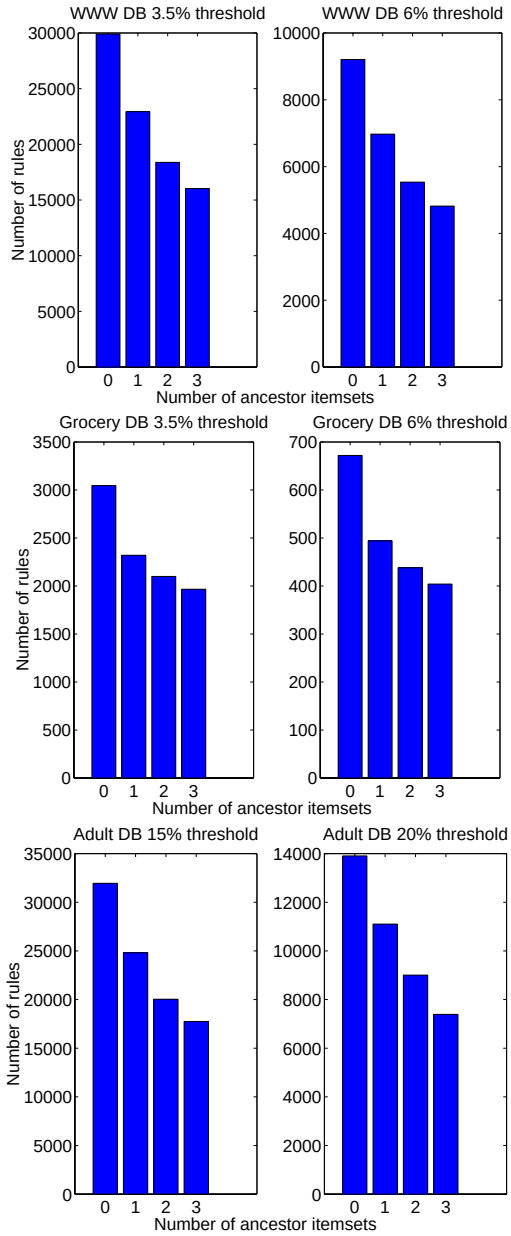

Figure 5.2: **Reduction in the number of mined rules for the WWW Proxy Logs Database, the Grocery Database and the Adult Database.** *Each pair of histograms describes the reduction in the number of rules mined using 0, 1, 2 and 3 ancestor itemsets classified for deletion, when two different mining thresholds are used for each database.*

thresholds each. Each histogram details the number of rules mined with a specific threshold, and each bar represents the number of rules mined when 0, 1, 2 and 3 ancestor itemsets were used in the mining process. For example, for the Grocery Database, the three ancestor itemsets were: $\iota' = \{\texttt{tomatoes}, \texttt{cucumbers}\}$, $\iota'' = \{\texttt{tomatoes},\texttt{onions}\}$, and $\iota''' = \{\texttt{cucumbers},\texttt{onions}\}$. The reduction in the number of rules mined from the Grocery Database when deleting the first ancestor itemset, ι', during the mining process was approximately 24%. As expected, this was equivalent to the post-processing deletion of association rules in the Interestingness Via What Is Not Interesting approach (Chapter 4) brought about by deleting the families of the two ancestor *rules* $\rho' = \langle\texttt{tomatoes}\rangle \rightarrow \langle\texttt{cucumbers}\rangle$ and $\rho'' = \langle\texttt{cucumbers}\rangle \rightarrow \langle\texttt{tomatoes}\rangle$. This was expected since $\textit{binuclear-family}(\iota') = \textit{family}(\rho') \cup \textit{family}(\rho'')$. ρ' and ρ'' were the first and second ancestor rules, respectively, classified as true-not-interesting when running the Interestingness Via What Is Not Interesting algorithm on the Grocery Database.

The reduction in the number of rules mined as result of deleting the first few ancestor itemsets is more substantial than the reduction brought by the deletion of later ancestor itemsets. This is parallel to the deletion of a larger number of rules by the first few ancestor rules in the Interestingness Via What Is Not Interesting post-processing approach as compared to the deletions of later ancestor rules as we saw in Section 4.5. The explanation to both events is the same, captured by the $RSCL$ value; the families of ancestor rules with larger $RSCL$ values are selected first for user classification and potential deletion (Section 4.4.2). Similarly, the binuclear-families of ancestor itemsets with larger support are selected first for user classification and potential deletion during the mining process (Section 5.3.2).

For the data depicted in Figure 5.2, the average reduction in the number of rules mined due to the deletion of the first ancestor itemset was more than 23%. The average reduction in the number of rules mined due to the deletion of the second ancestor itemset was just under 17%, and for the third ancestor itemset it was just under 12%. This decreasing impact of the deletions is one of the reasons we limit the number of ancestor itemsets brought for user classification to a maximum of three. Note that as the Grocery Database is very sparse, the reduction in the number of mined associations due to the deletion of the second and third ancestor itemsets was approximately 10% and 7% respectively, whereas for the WWW Proxy Logs Database it was still approximately 20% for the second ancestor itemset deletion, and for the Adult Database it was approximately 19%.

Overall, the reduction in the number of mined rules due to the deletion of the three ancestor itemsets was approximately 47% for the WWW Database, was 36% and 40% for 3.5% and 6% mining thresholds for the Grocery Database (the sparsest database), and 45% and 47% for the Adult Database. This is empirical verification that we do meet the goals we set for our algorithm.

5.4.2 Resource Consumption Discussion

The mining process is normally dominated by the time it takes to calculate the support of the candidate itemsets. We therefore start to analyze the performance of our new algorithm by first examining the behavior of the size of $\widehat{C}_k$, the transactions supporting C_k. Remember that we have used the Apriori-TID algorithm for association rule mining.

The top graph in Figure 5.3 depicts the size of $\widehat{C}_k$ (y-axis) for the Adult Database mined with 15% thresholds, for each of the iterations (the x-axis denotes the iteration number, k). The four lines depict the number of transactions in $\widehat{C}_k$ when 0, 1, 2 and 3 ancestor itemsets are used for deletion during the mining process. We note that after the fourth iteration, the size of $\widehat{C}_k$ starts to vary for the different cases, when 0, 1, 2 or 3 ancestor itemsets are used in the mining, decreasing much more rapidly when more ancestor itemsets are deleted during the mining process. In fact, when 3 ancestor itemsets were used in the mining process, 2 *less* iterations were required to complete the mining process than if no ancestor itemsets were used. When 1 or 2 ancestor itemsets were used, the mining process completed an iteration earlier than if no ancestor itemsets were used. This behavior was manifested after the second iteration for the WWW Proxy Logs Database when mining with 3.5%, 5% or 6% thresholds, as well as a similar reduction in the total number of iterations.

For large databases—too large enough to fit in main memory—a reduction in the number of iterations required to discover all the frequent itemsets is extremely significant. In those cases each iteration, k, requires a pass over the database (disk access) to determine the support of the candidate k-frequent (k-large) itemsets. Eliminating any one of these passes will consequently result in a significant saving in execution run-time since the execution time of mining processes is usually dominated by the time it takes to calculate the support of the candidate frequent itemsets.

The bottom graph in Figure 5.3 depicts the number of frequent k-itemsets

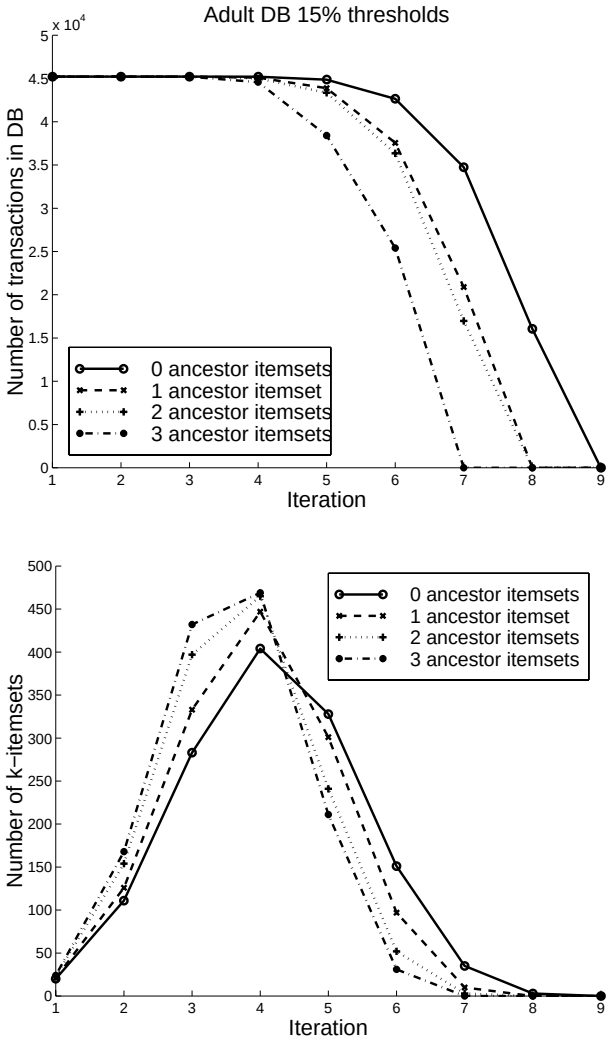

Figure 5.3: **Adult Database Apriori-TID results with 15% thresholds.** *The top graph depicts the size of $\widehat{C}_k$, the number of transactions supporting C_k, for every iteration (on the x-axis) when zero, one, two and 3 ancestor itemsets are deleted during the mining process (the different lines). The bottom graph depicts the number of frequent k-itemsets (the size of L_k) for each iteration.*

(the size of L_k on the x-axis) discovered on each iteration, when 0, 1, 2 and 3 ancestor itemsets are deleted during the Adult Database mining process with 15% thresholds. Again, after the fourth iteration, the number of frequent itemsets discovered in each iteration decreases more rapidly as more ancestor itemsets are incorporated into (deleted during) the mining process. The behavior of the candidate k-itemsets (C_k prior to the enforcement of the support threshold criterion) was similar to that of the frequent k-itemsets

The cumulative iteration runtime, in seconds, for the three databases is presented in Figure 5.4. These runtimes were recorded using a Compaq ProLiant DL580, with four 700MHz Xeon CPUs, 1GB RAM, running Solaris 7 and Perl 5.005_03. The iteration runtime for later iterations ($k > 4$ for the Adult Database and $k > 2$ for the WWW Proxy Logs Database) decreases with the number of ancestor itemsets incorporated into the mining process. This can be expected considering the alteration of the search space: we could only flatten the search space with the domain knowledge available to us. That means that fewer iterations are needed to complete the search, but more candidate itemsets are discovered during those first iterations. The number of k-itemsets is large for the first few iterations, as is the number of transactions that support them. This number decreases significantly for larger k-s, but over the relatively small databases that we experimented with—that can be easily stored in main memory—not enough to always offset the extra execution time added in the first few iterations, especially when using the Apriori-TID algorithm, as we did, which is the worse case scenario resulting in these cases with a cumulative runtime that is longer the more ancestor itemsets are used.

Mining the Adult Database with 20% thresholds and no ancestor itemsets took 1:27.27 hours (all the mining execution times were recorded using the Apriori-TID algorithm). The mining took 1:32.77 hours (an extra 5.5 minutes) when the first ancestor itemset was used for deletion in the mining, and 1:42.22 and 1:47.65 hours respectively when two and three ancestor itemsets were used. The mining execution times for the Adult Database with 15% thresholds, when using 0, 1, 2 and 3 ancestor itemsets were: 2:56.60 hours, 3:08.67 hours (a 6.8% increase), 3:31.48 hours and 3:36.97 hours respectively. These general results held for the other databases, as in the middle and bottom graphs in Figure 5.4 that depict the cumulative iteration times for the Grocery Database and the WWW Proxy Logs Database when mined with 3.5% thresholds. The times on the y-axis are recorded in seconds. Note the small number of iterations while mining the sparse Grocery Database,

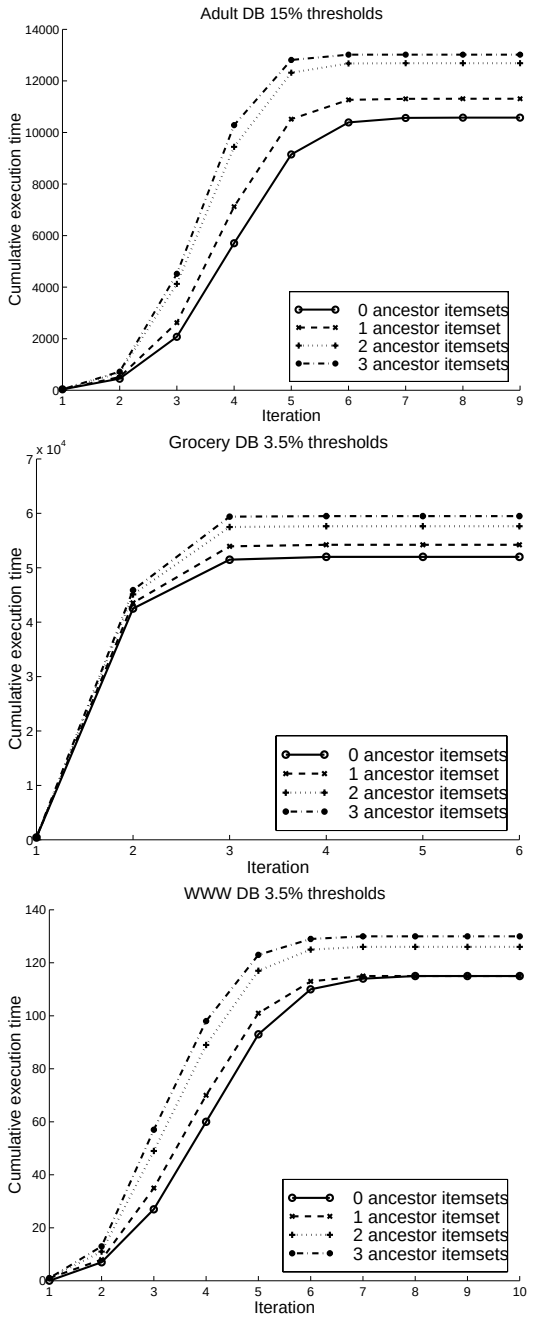

Figure 5.4: **Cumulative iteration runtimes, in seconds, for the Adult Database, the Grocery Database and the WWW Proxy Logs Database.**

compared to the large number of iterations while mining the dense WWW Database.

In many KDD applications disk access is one of the more time-consuming tasks. Our algorithm reduces the number of passes required over the database. In our experiments, due to the relatively small datasets that we used, we experienced the case that the computation time increased and somewhat unfavorably impacted the overall runtime. One might conjecture that for large databases, too large to store $\widehat{C}_k$ in main memory and hence requiring a pass over the entire database for each iteration, the savings brought about by performing fewer iterations over the data (disk access) would have an important positive impact, significantly decreasing the overall execution time.

It is interesting to observe that the relatively short execution times of mining the WWW Proxy Logs Database combined with the large number of rules mined from this dense database, resulted in better *overall* runtime results (consisting of both Apriori-TID and `ap-genrules` runtimes) when incorporating the first ancestor rule into the mining process. The runtime of the Apriori-TID algorithm when the WWW Proxy Logs Database was mined with 3.5% thresholds was 115 seconds with either 0 or 1 ancestor itemsets. The total runtime of that mining process was 140 seconds when no ancestor itemsets were used, and only 130 seconds when one ancestor itemset was used (with 29,910 vs. 22,932 rules produced). The runtime of the Apriori-TID algorithm when the WWW Proxy Logs Database was mined with 6% thresholds was 51 seconds with either 0 or 1 ancestor itemsets. The total mining process runtime was 65 seconds when no ancestor itemsets were used, and 54 seconds when one ancestor itemset was used (with 9,206 vs. 6,972 rules outputted).

5.5 Conclusions

Since what is interesting is ultimately subjective, to fully address the problem of interestingness, subjective interestingness criteria must be used. Incorporating some subjective interestingness into the mining process can result in an output that contains significantly fewer rules, which are more likely to be interesting, without restricting the output to a narrow audience only. In this chapter (the work presented in this chapter is based on [Sah02b]) we investigated how to do this by incorporating into the mining process the domain knowledge used for post-processing in Chapter 4, to create a simple, general

and effective mining process for the user.

As we expected, the reduction in the number of rules outputted by our altered mining process, into which we incorporated the classification of ancestor itemsets, is similar to the post-processing deletion by the Interestingness Via What Is Not Interesting approach (Chapter 4) of the two corresponding ancestor rule families. The domain knowledge we incorporate can only be used to alter the search space by flattening it. That results in more itemsets produced in the first few iterations, making those iterations' execution time longer. The execution time of later iterations is reduced, and the last one or two iterations are often eliminated. Over the three databases we experimented with the execution time of this new mining process that outputs much fewer rules is longer than that of mining the entire set of association rules. The significant gain is in the reduced number of iterations. In fact, we expect that over very large databases the elimination of one or two iterations will be significant enough to more than compensate for the larger execution times of the first few iterations. This will be especially true for databases that are too large to store in memory, when a pass over the database is required for each iteration. Each pass over the database would then require time-consuming disk access, and eliminating even a single iteration will likely be very significant in reducing the overall runtime. In general, there is no consensus on whether it is advisable to push constraints, objective or subjective, into the mining process. Recently [HG02] *"argue that this approach* [to push mining constraints into the mining algorithm] *is based on an understanding of KDD that is no longer up-to-date."* Their premise is that KDD is an iterative process of discovery rather than *"pure hypothesis investigation"*, and that pushing constraints into the mining process puts the mining at risk of being reduced to hypothesis testing. An additional risk they mention is that repeated mining may be required when constraints are used in the mining process. We saw that in our approach this is *not* the case. [HG02] do suggest to execute an initial, unconstrained mining run (if feasible) to output all the potentially interesting rules, followed by iterative processing of the list of outputted rules. In fact, [HG02] say that since *"During rule generation the machine does not rely on any human interaction. So letting the machine stupidly counting frequencies in the data actually does not bind any human resources."* Additionally, [SR00] argue that rules with higher order semantics can be extracted by post-processing the results of the mining process.

Chapter 6

Exploring Interestingness Via Clustering: A Framework*

Instead of stubbornly attempting to use surrealism for purposes of subversion, it is necessary to try to make of surrealism something as solid, complete and classic as the works of museums.

—Salvador Dalí

Determining interestingness is a notoriously difficult problem: we have seen that it is subjective and elusive to capture. We have also seen that interestingness is a central problem, and is becoming more and more important with the increasingly larger databases that are being mined. In this chapter we introduce and investigate a framework for association rule clustering that enables the automation of much of the laborious manual effort normally involved in the exploration and understanding of interestingness. Clustering is ideally suited for this task; it is the unsupervised organization of patterns into groups, so that patterns in the same group are more similar to each other than to patterns in other groups. We also define data-driven labeling of these clusters, called the ancestor coverage, which provides an intuitive, concise representation of the clusters.

*A preliminary version of this chapter will be published in the IEEE International Conference on Data Mining (ICDM-2002) [Sah02a].

6.1 Introduction

Determining which patterns are interesting is an important but known to be a very difficult problem. Many approaches, requiring varying degrees of user intervention, have been formulated to tackle the problem (see Chapters 4 and 5 for a review). In this work we were motivated by the Interestingness Via What Is Not Interesting approach described in Chapter 4. This approach circumvents the need to capture extensive amounts of domain knowledge, and instead, automatically identifies similar groups of rules (families) as the first step in replacing a significant amount of the tedious, manual effort commonly involved when investigating interestingness.

The advantages of post-processing and exploration of the mined association rules are presented and thoroughly discussed in [SR00]. As [LHH00] emphasize, the difficulty in analyzing the mined rules is in how to organize them effectively. It becomes clear that automated tools to enable the KDD users to understand the mined results are required. We tackle this challenge.

In this chapter we introduce a general clustering framework to automate the exploration of masses of mined rules by organizing them in groups according to similarity. The grouping and naming scheme clustering provides will facilitate KDD users' exploration and understanding of the interestingness of mined rules. In this, we depart from the typical application of clustering to data in KDD to that of clustering association rules. To make the interpretation of the resulting clusters easy, we also introduce a data-inferred, concise representation of the clusters, called the ancestor coverage. Details of our contributions follow below. Note that the input for the association rule clustering framework we introduce in this chapter is the exhaustive list of mined patterns.

6.1.1 Clustering and Association Rules

Clustering is the unsupervised organization of similar objects into natural groups, or clusters, so that objects within a cluster are more similar to each other than to objects in other clusters [JMF99, Har75, DH73, HK01b]. Since clustering does not require the pre-classification or labeling of patterns, it is ideally suited for interestingness exploration; the output of the mining process is a list of unlabeled (as interesting or not-interesting) patterns, the association rules. Clustering provides a way to group these unlabeled association rules into meaningful, labeled groups.

In the KDD process, clustering is frequently used to gain insight into the distribution of the *data* and the emerging groups in the data for further use or investigation [HK01b, KHK99]. In the context of association rules, clustering was previously used to partition intervals of interest over numeric data domains. These intervals were used as input for association rule mining algorithms; rules were identified over these intervals as in [MY97]. In [LSW97], association rules with two numeric attributes in the assumption were clustered by joining assumption attribute intervals. [Han95] proposed to the run the mining process over sets of clustered data. In our work, we use clustering differently, on an input of association rules, and for a different purpose, to investigate and understand the interestingness of those rules. Indirectly related are works are on summarization such as [LHM99b]'s summary that consists of less-complex Direction Setting rules, [AY98a]'s summary that prefers more-complex rules for any given support and confidence levels, and [LHH00] who summarizes relationships in data and their exceptions. The work in Chapter 4 can be viewed as a form of partial summarization.

[TKR+95] were the first to use groupings to improve the understandability of rule covers with a common consequent over data domains constrained by a rule-confidence monotonicity assumption. [TKR+95] grouped 29 such cover rules *"that make statements about the same database rows [...]"*. In this work we introduce a *general* framework for clustering *un*constrained association rules over *un*constrained domains in order to enable the exploration of interestingness. To make the clusters easy to investigate, we define a concise, data-driven cluster representation. Additionally, we investigate the drawbacks and advantages of existing and new similarity measures, and the results of their application on real databases. [TKR+95] introduced a similarity measure defined by the number of transactions two rules with equal consequents differ on. [DL98] proposed a distance metric to detect unexpected rules in a defined neighborhood where unexpectedness is determined according to unexpected confidence or density, not for clustering. A measure to differentiate between attributes with overlapping and non-overlapping numeric values was proposed in [GB98]. Out of scope are objective criteria [HH01a, TKS02], Chapter 3, as well as measures of surprise or exception rules such as [SK98, LHM01a]. Note that throughout this discussion we will use similarity measures that do not require explicit use of domain knowledge since our goal is to introduce a general clustering framework. The powerful and intuitive similarity based association rule grouping introduced in [AT01, AT99] depends on the existence of an attribute hierarchy as well as

a cut defined on the hierarchy. The attribute hierarchy, organized as a tree, is often defined by the domain expert although in some domains such a hierarchy may already exist, or may be constructed automatically. A subset of the nodes of the tree that contains one node exactly from each path from a leaf node to the root of the tree is called a cut. This cut is specified by the domain expert. The rule resulting from the mapping of the attributes of a mined association rule to their corresponding nodes in the cut is called an aggregated rule. Association rules that are mapped to the same aggregated rules are similar and therefore grouped together. Thus, the method of [AT01, AT99] is an intuitive way of clustering rules together according to the similarity inferred from the attribute hierarchy and specified cut without defining a distance metric. Since the cut and, generally, the attribute hierarchy are defined by the domain expert, we will not be able to use this intuitive similarity measure in the general clustering framework.

The rest of the chapter is organized as follows. We introduce some new definitions and notations used in the clustering framework in Section 6.2. In Section 6.3 we introduce and discuss the clustering framework to enable interestingness exploration in a domain- and constraint-independent setting. In Section 6.4 we provide a discussion as a result of the empirical analysis of clustering thousands of association rules mined from real databases, including the definition of the data-driven cluster representation, and some implementation issues. We conclude in Section 6.5 with a summary.

6.2 Definitions and Preliminaries

Let $a, b \in \Lambda$, and $\rho = \ a \rightarrow b$. An indicator of the size of the coverage list of ρ in Ω is the $RSCL_\Omega(\rho)$, defined in Equation 4.3. We define $RSFS_\mathcal{C}(\rho)$, a variation of $RSCL_\Omega(\rho)$. $RSFS_\mathcal{C}(\rho)$ is the **Relative Size of the Family of ρ over a Set of association rules $\mathcal{C}$**, defined as:

$$\mathbf{RSFS}_\mathcal{C}(a \rightarrow b) \stackrel{\mathbf{Def}}{=} \frac{\|\{r \in \mathcal{C} | r = \ A \rightarrow B \ , a \in A, b \in B\}\|}{\|\Omega\|}. \qquad (6.1)$$

Note that $RSFS_\mathcal{C}(\rho)$ is defined over an arbitrary set of rules $\mathcal{C}$. Therefore, ρ may or may not be in $\mathcal{C}$. The concept of **ancestor coverage** will be introduced in Section 6.4.1, where we use $RSFS_\mathcal{C}(\rho)$ to construct the ancestor coverage.

An additional term we introduce is **inclusion**. **Inclusion of two sets**, A and B is said to occur when one of the two sets is a subset of the other, that is if $A \subseteq B$ or if $B \subseteq A$. If $A \subseteq B$ we will refer to A as **included** in B.

Inclusion of two rules, $A \rightarrow B$ and $C \rightarrow D$, is said to occur when the assumption of the first rule is included in the assumption of the second rule *and* the consequent of the first rule is included in the consequent of the second rule *or* vice versa, the assumption and consequent of the second rule are included in that of the first rule respectively. Formally, inclusion of the two rules $A \rightarrow B$ and $C \rightarrow D$ is said to occur if $(A \subseteq C) \wedge (B \subseteq D)$, *or* if $(C \subseteq A) \wedge (D \subseteq B)$.

6.3 Interestingness Exploration Through Clustering

Clustering is ideally suited for the exploration and understanding of interestingness of association rules: clustering groups patterns that possess similarities to each other. If some association rules in a cluster have a certain property, other rules in that cluster are expected to have the same properties. The labeling of the association rule cluster is provided by the ancestor coverage, defined in Section 6.4.1. Clustering processes are easy to apply and often yield interesting results, even when the solutions were discovered by intuitive methods and have no established properties [DH73]. These results can guide the application of more rigorous procedures.

In this section we introduce a framework for clustering of the *results* of the mining process, the association rules, rather than of the data used for the mining process. In this sense, our work can be seen as an example of the higher order mining introduced by [SR00], and a departure from the typical use of clustering in data-mining. The clustering process typically includes the following steps [JMF99, Har75] (1) profiling, (2) similarity measure definition, (3) clustering algorithm, and (4) representation and evaluation of the clustering, presented in Sections 6.3.1, 6.3.2, 6.3.3 and 6.4 respectively.

6.3.1 Association Rule Profiles

A cluster of association rules is a set of similar rules. Before we can define the "similarity" of rules, we need to define what characterizes association rules, or their profiles.

There are no theoretical guidelines to determine the features best suited for use in a similarity measure, how those features should be normalized, or which is the most appropriate similarity measure to be used [JMF99]. Normally, it is assumed that the pattern profile has been determined prior to clustering. This profile is very important since a skillful selection of the feature vector to be used can yield simple and easily understandable clustering that will provide much insight into the interestingness of the rules, whereas a poor selection can yield complex clustering that provides little or no insight. In our case, where Λ can be a large set of attributes, the potential feature vector used for clustering can be very large.

We choose to concentrate on the five characteristics that fully define an association rule $r = A \rightarrow B$:

1. The assumption: A,
2. The consequent: B,
3. The rule support: the percentage of transactions that contain $A \cup B$,
4. The assumption support: the percentage of transactions that contain A, and,
5. The consequent support: the percentage of transactions that contain B.

The first two features of association rules are mutually exclusive attribute sets. Note that we do not include $A \cup B$ as an independent feature since it can be uniquely derived from the assumption and the consequent. The three support level features are real numbers between 0 and 1. Comparing and weighting of components of a feature vector with varying domains, significance and scales such as the ones we have listed above is an unsolved challenge [JMF99, DH73]. We discuss this in the context of the similarity measures.

6.3.2 Similarity Measures for Association Rules

The problem of association rule clustering is that of organizing similar rules into groups, so that rules within one cluster are more similar to each other than to rules in different clusters. The challenge in this is that clustering is subjective and rules can be partitioned differently for different purposes.

The question is how to define this natural similarity between two association rules.

We follow the reasoning in [DH73] and abandon the adherence to a distance metric and use a nonmetric similarity function. A distance metric is a function of two elements that obeys the metric properties of identity, positivity, symmetry and the triangle inequality (Section 3.1.3). Symmetric measures will suffice, as long as their values are smaller when the inputs to the measures are more similar, and we will not check for compliance with the metric properties. Note that in some cases it may be beneficial to have a non-symmetric similarity measure for association rules. For example, examine the two rules $r_{AB} = A \rightarrow B$ and $r_{ABC} = A \rightarrow BC$. Since r_{AB} can be "deduced" from r_{ABC} and r_{ABC} cannot be "deduced" from r_{AB} (without the matching support and confidence levels of the rule, of course), there could conceivably be cases where a non-symmetric similarity measure may serve better. However, since clustering is normally performed using symmetric similarity measures, we will adhere to those in this work.

Below we review d_{sd} and d_{iset}, two very different, representative measures available in the literature in the context of our expectation of a good similarity measure: that the dissimilarity between rules in the same cluster will be significantly less than that of rules in different clusters. Since neither of those measures satisfies this, we introduce a third similarity measure that complies with this expectation, d_{sc}.

Support-Difference Measure

[TKR+95] defined a similarity measure between associations rules that share an equal consequent, $A \rightarrow B$ and $C \rightarrow B$, as:

$$d_{ms}(A \rightarrow B, C \rightarrow B) = \\ [support(A \cup B) + support(B \cup C) - 2 \cdot support(A \cup B \cup C)] \cdot \|\mathcal{D}\|,$$

where $\mathcal{D}$ is the set of all transactions. This measure is naturally extended to unconstrained association rules as:

$$d_{sd}(A \rightarrow B, C \rightarrow D) = \\ [support(A \cup B) + support(C \cup D) - 2 \cdot support(A \cup B \cup C \cup D)] \cdot \|\mathcal{D}\| \tag{6.2}$$

similarly to how it was done in [DL98], with a change of scale.

The d_{sd} measure appears to be a very natural rule-similarity measure, and it may fit many needs, but in extending the measure beyond association rules with a common consequent, d_{sd} cannot distinguish between rules that are potentially very different. Namely, the d_{sd} measure does not differentiate between attributes in the assumption and the consequent. For example, consider the two rules $r_1 = \langle \texttt{tomatoes} \wedge \texttt{sugar} \rangle \rightarrow \langle \texttt{cucumbers} \rangle$, and $r_2 = \langle \texttt{tomatoes} \wedge \texttt{cucumbers} \rangle \rightarrow \langle \texttt{sugar} \rangle$. A user who is not interested in the family of $r_{tc} = \langle \texttt{tomatoes} \rangle \rightarrow \langle \texttt{cucumbers} \rangle$, that is, in association rules that contain tomatoes in their assumption and cucumbers in their consequent, would not be interested in r_1. However, this user could potentially be interested in what elements in a shopping basket may increase the probability that a customer will also purchase sugar, as in r_2. The two rules r_1 and r_2 are therefore *not* similar to this user, and yet $d_{sd}(r_1, r_2) = 0$. This does not match our expectation of a good similarity measure: d_{sd} can rank rules that are *less* similar to the user (r_1 and r_2) as closer to each other than rules that are more similar, for example, r_{tc} and r_1 (since $d_{sd}(r_{tc}, r_1) > 0 = d_{sd}(r_1, r_2)$).

[DL98] noted a special case of the example above to show that d_{sd} is not a distance metric: d_{sd} does not comply with the positivity property ($\forall p \neq q, d(p, q) > 0$). [DL98] used the example of two rules, $R_1 = A \rightarrow B$ and $R_2 = B \rightarrow A$, where $d_{sd}(R_1, R_2) = 0$ to show this. Remember that we do not require distance metrics, and similarity measures are sufficient for our needs in defining the general clustering framework.

Note that the d_{sd} measure does not use all the features of the association rule profile defined in Section 6.3.1; it only uses the last three features.

Attribute Distance Measure

[DL98] introduce a novel distance measure with the goal of detecting unexpected confidence or density in neighborhoods, potentially interesting rules. This measure, defined in Equation 6.4, shown in [DL98] to be a distance metric, relies entirely on the attributes in the rules and their distribution amongst the assumption and consequent:

$$d_{iset}(A \rightarrow B, C \rightarrow D) = \\ \delta_1 \cdot \|(A \cup B) \oplus (C \cup D)\| + \delta_2 \cdot \|A \oplus C\| + \delta_3 \cdot \|B \oplus D\|. \tag{6.3}$$

[DL98] recommends the values: $\delta_1 = 1$, $\delta_2 = (n-1)/n^2$, and, $\delta_3 = 1/n^2$, where $n = \|\Lambda\|$. For databases with a large number of attributes, such

as the Grocery Database where $n = 1757$ (Section A.1.1), d_{iset} imposes a strong similarity bias for rules over the same set of attributes: $A \rightarrow B$ and $C \rightarrow D$ such that $A \cup B = C \cup D$. In fact, the values of both δ_2 and δ_3 fall under 0.09 for $n > 10$. This property is *not* always favorable. [DL98] do not discuss how to set the weights for a large number of attributes. We therefore have also used the d_{iset} measure with equal δ_i weights to circumvent this problem. However, we still run across a problem where two rules mined from the Grocery Database, $r_3 = \langle \texttt{cucumbers} \wedge \texttt{tomatoes} \rangle \rightarrow \langle \texttt{onions} \rangle$ and $r_4 = \langle \texttt{cucumbers} \wedge \texttt{tomatoes} \rangle \rightarrow \langle \texttt{distribution center 3} \rangle$, such that $support(r_3) = 0.17$ and $confidence(r_3) = 0.41$, and such that $support(r_4) = 0.037$ and $confidence(r_4) = 0.09$, are ranked very closely by d_{iset}, whereas there is a significant difference between them, as marked by $d_{sd}(r_3, r_4) = 0.21 \cdot \|\mathcal{D}_{GroceryDatabase}\|$, a relatively large distance for rules with the same assumption (specifically, $d_{sd}(r_3, r_4) = 14,138$ since the size of the Grocery Database is 67470, see Section A.1.1). In fact, according to d_{iset} adding any single attribute to the assumption (or the consequent) of a rule results in a rule that is equally distant to the original rule. At the same time, a single additional attribute can drastically change the significance of a rule in many different ways as we will exemplify in the discussion below.

Note that this distance measure only uses the first two features of the association rule profile defined in Section 6.3.1.

New Similarity Measure

In Equation 6.4 we introduce a new similarity measure, d_{sc}, that addresses the drawbacks of the measures mentioned above and draws on their strengths. We first define the $diff_{sup}$ function, the **difference-in-support** function, defined following the idea in [TKR+95] as in d_{sd} for two sets of attributes X and Y:

$$diff_{sup}(X, Y) = support(X) + support(Y) - 2 \cdot support(X \cup Y).$$

Note that $diff_{sup}$ is defined over sets of attributes rather than rules. $diff_{sup}$ can be viewed as a similarity measure, and similarly to d_{sd}, $diff_{sup}$ is not a distance metric as it does not comply with the positivity property. For example, for $X = \{a\}$ and $Y = \{a, b\}$ such that $support(X) = support(Y)$, we have $diff_{sup}(X, Y) = 0$ for $X \neq Y$.

Now,

$$d_{sc}(A{\rightarrow}B, C{\rightarrow}D) \stackrel{\textbf{Def}}{=} \tag{6.4}$$
$$[1 + \mathit{diff}_{sup}(A, C)]\frac{\|A \oplus C\|}{\|A \cup C\|}\gamma_1 + [1 + \mathit{diff}_{sup}(B, D)]\frac{\|B \oplus D\|}{\|B \cup D\|}\gamma_2 +$$
$$[1 + \mathit{diff}_{sup}(A \cup B, C \cup D)]\frac{\|(A \cup B) \oplus (C \cup D)\|}{\|A \cup B \cup C \cup D\|}\gamma_3.$$

According to d_{sc}, the dissimilarity between two rules is the weighted sum of dissimilarities between the assumptions (first element in d_{sc}), the consequents (second element in d_{sc}) and the attribute sets that make up the two rules (last element in d_{sc}), providing a natural similarity measure in the sense that it ranks rules that most users would perceive to be similar as closer, and rules that are less similar to each other as more distant. Each dissimilarity component (assumption, consequent and rule attribute set) is measured through the combination of the dissimilarities in support (as measured by diff_{sup}), and the attribute set (as measured by the `xor` ratio), and so the more similar two assumptions (consequents or rule attribute sets) are, the smaller this value is, and the more dissimilar they are, the larger this value will be. Note that 1 is added to the diff_{sup} value, to ensure that when the support difference is zero, this value does not cancel out the weight of the potential difference in the attributes; it is entirely possible that $\mathit{diff}_{sup}(X, Y) = 0$ and at the same time $X \oplus Y \neq \emptyset$. A further motivation for using $\|X \oplus Y\|/\|X \cup Y\|$ as the similarity measure of the respective attribute sets is that it constitutes a typical similarity measure between two binary vectors and is used extensively in statistics and marketing applications.

As in [DL98], choices for the values of the gamma-weights reflect preferences, in our case for inclusion. The gamma-weights are therefore not constants; they are functions of the respective assumptions and consequents. Inclusion of itemsets or rules, defined in Section 6.2, is not captured by the `xor` relationship. For example, let $A_1 = \{a, b\}$, $A_2 = \{a\}$, $A_3 = \{a, b, d\}$ and $A_4 = \{a, b, c\}$ be the assumptions of four different rules. The attribute dissimilarity between A_1 and A_2, $\|A_1 \oplus A_2\|/\|A_1 \cup A_2\| = 1/2$. This is equal to the attribute dissimilarity between A_3 and A_4, 2/4. However, $A_2 \subset A_1$, whereas $A_3 \not\subset A_4$ and $A_4 \not\subset A_3$. In that sense, A_1 and A_2 are closer than A_3 is to A_4. We use the gamma-weights to reflect this inclusion, and set γ_1 in Equation 6.4 to a value γ, $0 < \gamma < 1$ if $A \subseteq C$ or if $C \subseteq A$, and

to 1 otherwise. Note that we do not treat $A = C$ differently since in that case $\|A \oplus C\| = 0$, and the assumption will not contribute to d_{sc}. Since the same analysis holds for consequents, we set γ_2 in Equation 6.4 to the same value γ, if $B \subseteq D$ or if $D \subseteq B$, and to 1 otherwise. For $r' = A \rightarrow B$ and $r'' = C \rightarrow D$, where $A \subseteq C$ and $D \subseteq B$, the rules r' and r'' are not included in each other, even though the assumption r' is included in that of r'', and the consequent of r'' is included in that of r'. Rule inclusion occurs for the two rules r' and r'' when $A \subseteq C$ and $B \subseteq D$, or $C \subseteq A$ and $D \subseteq B$. To weigh in rule inclusion, we set γ_3 in Equation 6.4 to the same value γ, $0 < \gamma < 1$ if $((A \subseteq C) \wedge (B \subseteq D)) \vee ((C \subseteq A) \wedge (D \subseteq B))$, and to 1 otherwise. We set all the gamma-weights to the same value, γ, and for the rest of this discussion, we refer to the value $\gamma = 1/2$ (we received similar results for other values).

Note that d_{sc} uses *all* five association rule profile features listed in Section 6.3.1, and the problematic similarities for the rules discussed in previous sections are remedied. For reference, $d_{sc}(r_1, r_2) = 2.38$, $d_{sc}(r_3, r_4) = 1.96$, and $d_{sc}(r_{ct}, r_{tce}) = 0.57$ where $r_{tc} = \langle$`tomatoes`$\rangle \rightarrow \langle$`cucumbers`$\rangle$, $r_{tce} = \langle$`tomatoes`$\rangle \rightarrow \langle$`cucumbers`$\wedge$`dozen#2 eggs`$\rangle$. An example of a cluster follows below.

Example 6.3.1 *We follow the formation of a new cluster characterized by the director of the produce department as "Mediterranean Salad," since it describes shopping patterns of customers who purchase the common ingredients of a Mediterranean Salad: tomatoes, cucumbers, onions and herbs such as dill and parsley. The first four rules to be successively joined to form this new Mediterranean Salad cluster, $\mathcal{C}_{ms}$, are:* $\langle$`tomatoes`$\wedge$`cucumbers`$\wedge$ `onions`$\rangle \rightarrow \langle$`dill`$\rangle$, $\langle$`cucumbers`$\wedge$`onions`$\rangle \rightarrow \langle$`dill`$\rangle$, $\langle$`tomatoes`$\wedge$`onions`$\rangle \rightarrow$ $\langle$`dill`$\rangle$ *and* $\langle$`tomatoes`$\wedge$`cucumbers`$\rangle \rightarrow \langle$`dill`$\rangle$. *This cluster, $\mathcal{C}_{ms}$, is then merged with a previously formed cluster, the cluster:* { $\langle$`tomatoes`$\wedge$`cucumbers`$\wedge$`parsley`$\rangle \rightarrow \langle$`dill`$\rangle$, $\langle$`cucumbers`$\wedge$`parsley`$\rangle \rightarrow \langle$`dill`$\rangle$, $\langle$`tomatoes`$\wedge$ `parsley`$\rangle \rightarrow \langle$`dill`$\rangle$ }. *The next few rules to be merged into the cluster have larger consequents, such as* $\langle$`cucumbers`$\wedge$`parsley`$\rangle \rightarrow \langle$`tomatoes`$\wedge$`dill`$\rangle$.

Similarity Measures Discussion

Upon closer examination of the d_{sd} similarity measures we note that d_{sd}'s first preference is to cluster rules that were generated from the same itemsets. Let

I be an itemset and $A, B \subset I$, $A, B \neq \emptyset$, $A \neq B$.

$$\begin{aligned} & d_{sd}(A \rightarrow I \setminus A), B \rightarrow I \setminus B) = \\ & \quad = [support(A \cup (I \setminus A)) + support(B \cup (I \setminus B)) - \\ & \qquad\qquad 2 \cdot (A \cup (I \setminus A) \cup B \cup (I \setminus B))] \cdot \|\mathcal{D}\| \\ & \quad = [support(I) + support(I) - 2 \cdot support(I)] \cdot \|\mathcal{D}\| \\ & \quad = 0 \end{aligned}$$

This clustering of rules around their generating itemsets is not always ideal. We saw that for a user who classified the family of r_{tc} =⟨`tomatoes`⟩→⟨`cucumbers`⟩ as not interesting, the rule r_1 =⟨`tomatoes`∧`sugar`⟩→⟨`cucumbers`⟩, generated from I_{tsc} = {`tomatoes`,`sugar`,`cucumbers`}, is not-interesting, whereas the rule r_2 =⟨`tomatoes`∧`cucumbers`⟩→⟨`sugar`⟩, also generated from I_{tsc} *is* interesting. $d_{sd}(r_1, r_2) = 0$, and they could be the first two rules to be clustered together which would not be beneficial to a user (who would want interesting and not-interesting rules to be dissimilar). The number of pairs of rules that have a d_{sd} value of zero is significant since many itemsets are likely to generate more than one rule.

The d_{iset} distance measure displays a similar characteristic, manifested in a different fashion. For equal delta-weights (a similar analysis will hold for other constant weights), the smallest distance between two not-equal rules as measured by d_{iset} is 2. $d_{iset}(r', r'') = 2$ when $r' = A \rightarrow B$ and $r'' = C \rightarrow D$ are such that (1) $A = C$ and $(((B \subset D) \wedge (\|D \setminus B\| = 1)) \vee ((D \subset B) \wedge (\|B \setminus D\| = 1)))$, or (2) $B = D$ and $(((A \subset C) \wedge (\|C \setminus A\| = 1)) \vee ((C \subset A) \wedge (\|A \setminus C\| = 1)))$, that is, the only difference between the rules is in one extra element in either the assumption or the consequent of one of the rules. We examine three rules: $r_{adb} = ad \rightarrow b$, $r_{ab} = a \rightarrow b$ and $r_{acb} = ac \rightarrow b$ such that $confidence(r_{adb}) = 0.05$, $confidence(r_{ab}) = 0.5$ and $confidence(r_{acb}) = 1.0$. Now $d_{iset}(r_{adb}, r_{ab}) = d_{iset}(r_{acb}, r_{ab}) = 2$, however the rules are very different: the addition of the d constraint reduces the confidence to 5% whereas the addition of the c constraint *raises* the confidence to absolute certainty: 100%. This difference is clearly manifested by using $d_{sc}(r_{adb}, r_{ab}) = 0.2287$, an order of magnitude larger than $d_{sc}(r_{acb}, r_{ab}) = 0.0525$ where $support(\{a, b\}) = 0.205$, $support(\{a, b, d\}) = 0.005$, $support(\{a\}) = 0.41$, $support(\{a, d\}) = 0.1$, $support(\{a, c\}) = 0.205$, and $support(\{a, b, c\}) = 0.205$. For reference, $d_{sd}(r_{adb}, r_{ab}) = 40$ whereas $d_{sd}(r_{adc}, r_{ab}) = 0$. For a user interested in targeting the population characterized by b (could be purchase of a certain commodity, such as beer, or a

population segment feature, such as being married), the rule r_{abc} is obviously much more interesting than r_{abd}. Additionally, there is a large number of rules at each "distance level" (a distance measure of 2, 4, etc.), and choosing which of those measures to use to cluster first is random. This similarity measure therefore has an intrinsic shortcoming in not always being able to distinguish between interesting and not-interesting rules.

The new measure, d_{sc}, combines the best of both measures (d_{sd} and d_{iset}) to create a new measure where rules that are more similar (by taking into account all five features that fully define association rules) to each other receive a more similar measure by d_{sc} than rules that are less similar to each other. We have seen that both d_{sd} and d_{iset} have fundamental shortcomings in providing a natural similarity measure for clustering of association rules in order to explore their interestingness. These basic shortcomings stem from the fact that those two measures only use *some* of the features in the association rules profile defined in Section 6.3.1 (though it must be remembered that the two measures were not intended to be used for clustering unconstrained association rules over unconstrained domains for the purpose of interestingness exploration). The new measure we introduce does rely on *all* the features of the association rule profile that fully define an association rule. It is therefore a more meaningful and the preferred measure to the other measures have examined.

Similarity Measures Comparison

The main purpose of this work is to introduce and encourage the use of clustering of association rules in order to explore and understand their interestingness rather than to suggest the ultimate association rule similarity measure. However, although we have seen that d_{sc} is a more meaningful similarity measure, in order to provide evidence that d_{sc} performs well for clustering association rules, we examine the average intra-cluster distances when clustering the Adult Database using each one of the three similarity measures in Figure 6.1. The three subgraphs on the top row of Figure 6.1 depict the average intra-cluster distances for each clustering iteration (for the first 5,000 iterations as portrayed on the x-axis) when the clustering is performed using the d_{sc} similarity measure. The average intra-cluster distances are measured using d_{sc} on the leftmost subgraph, using d_{sd} on the middle subgraph and using d_{iset} on the rightmost subgraph. Similarly, the three subgraphs in the middle row depict the average intra-cluster distances when

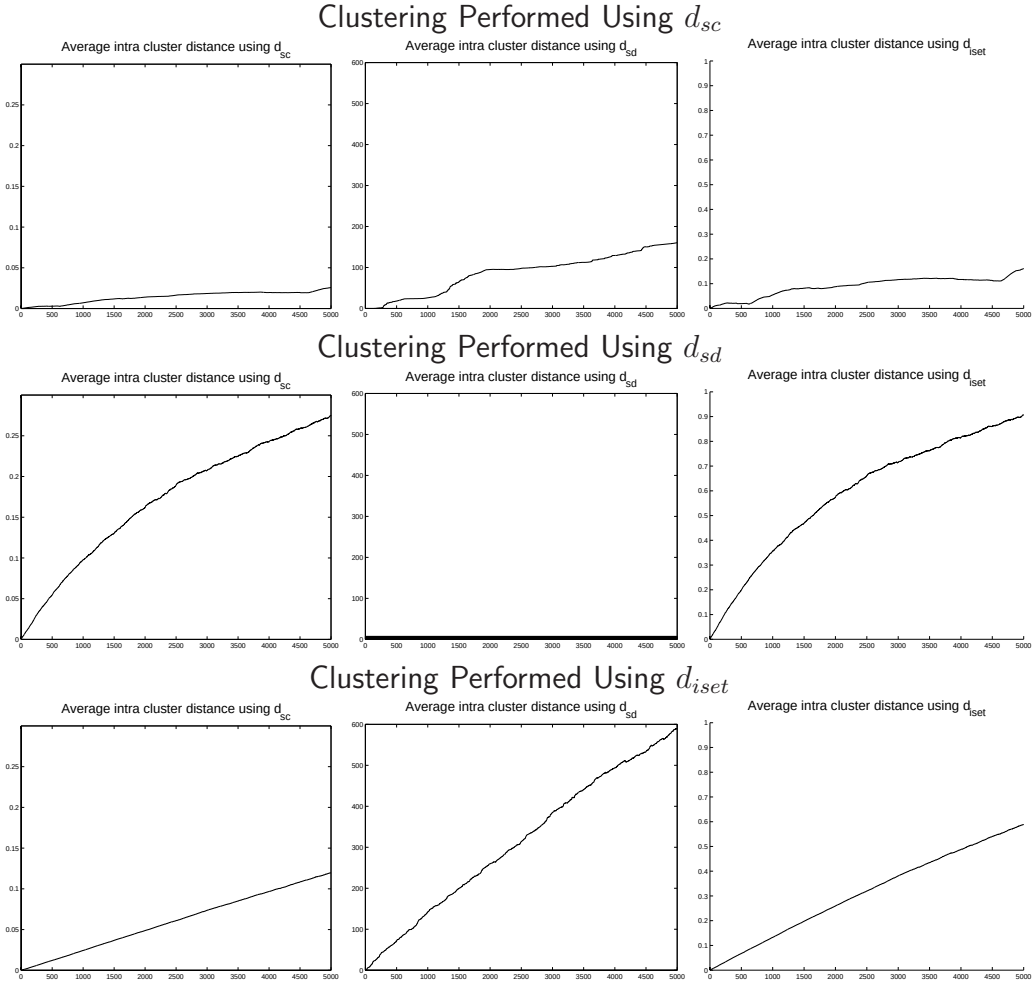

Figure 6.1: **The average intra-cluster distance measured using d_{sc}, d_{sd} and d_{iset} (subgraphs on the leftmost, middle and rightmost columns respectively) where the clustering is performed using d_{sc} (top row), d_{sd} (middle row) and d_{iset} (bottom row) on the Adult Database**

the clustering is performed using d_{sd} and the distances are measured using d_{sc} on the leftmost subgraph, using d_{sd} on the middle subgraph and using d_{iset} on the rightmost subgraph. Since there are more than five thousand pairs of association rules that have a similarity measure of zero according to d_{sd}, when clustering using d_{sd}, d_{sd} will record a zero intra-cluster distance for the first five thousand clustering iterations as portrayed in the flat line in the middle subgraph in Figure 6.1. The three subgraphs on the bottom row of Figure 6.1 describe the average intra-cluster distances when the clustering is performed using d_{iset} and the distances are measured using d_{sc}, d_{sd} and d_{iset} on the leftmost, middle and rightmost respectively. To make the results clearer, the same scale was used for each column of subgraphs since each column delineates the intra-cluster distances measured using the same similarity measure when the clustering was performed by each of the three similarity measures. We then see in the subgraphs in the leftmost column that when using d_{sc} to measure the intra-cluster distances where the clustering was performed by each of the three measures, the tightest clusters were achieved when clustering using d_{sc}. The same results can be seen when d_{sd} and d_{iset} were used to measure the intra-cluster distances in the next two columns (with the qualification on the middle subgraph when clustering according to d_{sd} groups pairs of rules generated from the same itemsets over these iterations, yielding a zero d_{sd} similarity measure and a flat line in that subgraph).

6.3.3 Clustering Algorithms

There are two general types of clustering algorithms, hierarchical and partitional. Hierarchical clustering algorithms create a hierarchical decomposition of the data, so that if two data points (association rules) are in the same cluster at a certain level in the hierarchy, they will remain together, in the same cluster, at all higher levels of clustering. Every hierarchical clustering spans a dendrogram that shows how the samples are grouped. There are two main classes of hierarchical clustering algorithms:

1. **Agglomerative, or bottom-up, clustering algorithms** that start with $m = \|\Omega\|$ singleton clusters and form the hierarchy by successively merging clusters, and,

2. **Divisive clustering algorithms** that start with one cluster that in-

cludes all the patterns and form the hierarchy by successively splitting clusters.

Partitional algorithms find data partitions that optimize (extremetize) the chosen similarity criterion. Since the partition problem is exponential in the size of its elements, iterative optimization is the approach commonly taken by partitional algorithms. This approach can only guarantee local optimization. There are partitional approaches that attempt to find the global optimal solution, but they are computationally prohibitive for large data sets [JMF99]. Comprehensive reviews of the clustering methodologies can be found in [HK01b, JMF99, Har75, DH73].

We chose to use the agglomerative hierarchical clustering method to explore the interestingness of association rules for several reasons. The dendrogram that portrays the rule-grouping at each level is extremely useful for thorough investigation of the clusters; users can choose to examine the association rules at any given clustering level, as well as follow the exact sequence that led to each clustering decision with a straightforward distance calculation available to explain how the decision was made. Additionally, with hierarchical clustering, there is no need to tackle the difficult questions that are part of partitional algorithms such as what a good starting point for the iterative optimization is and what is the ideal choice for final number of clusters. For completeness, we outline the basic agglomerative hierarchical algorithm we started with:

1. Initialize $\|\Omega\|$ singleton clusters, each containing a single association rule.

2. Find the nearest pair of clusters.

3. Merge the rules in the two distinct clusters into one cluster.

4. If the number of remaining clusters is 1, stop. Otherwise go to step 2.

The nearest pair of clusters in step 2 is determined by the two most similar rules that are members of different clusters. Rule similarity is determined using a similarity measure such as the ones described in Section 6.3.2, and the nearest clusters are the pair of clusters that those rules belong to respectively.

Changes made to the algorithm during the implementation are discussed in Section 6.4.

6.4 Discussion of Empirical Analysis

We used three databases for the clustering empirical analysis: the Adult Database, the WWW Proxy Logs Database, and the sparsest, and therefore most challenging database, the Grocery Database. These databases are described in Section A.1.

6.4.1 Representation and Evaluation of Clustering

Cluster Representation and Interpretation

To provide a compact representation, clusters need to be abstracted or named. The rule covers of [TKR+95] will not serve as a representation in the general case since they only work in monotonic domains, domains where there are no confidence rule exceptions, which is not always the case. A confidence rule exception occurs if a more specialized rule, a rule with a larger assumption and the same consequent, has a lower confidence.

We introduce the **ancestor coverage** for cluster representation. We define the **ancestor coverage** of a cluster of association rules, $\mathcal{C}$, to be the set of ancestor rules: $\{a \rightarrow b | a, b \in \Lambda, \forall r = X \rightarrow Y \in \mathcal{C}, r \in family_{\Omega}(a \rightarrow b)\}$. For the representation to be effective, we need to find a minimal ancestor coverage set. For computational efficiency over large clusters we use a greedy algorithm to find the ancestor coverage.

To find the minimal ancestor coverage, we need to depart from the $RSCL$ indicator for coverage list size (Section 4.2.2), and use $RSFS$ (Section 6.2). Although the difference between $RSCL$ and $RSFS$ when calculated over the same set of rules, $\mathcal{C}$, is at most $1/\|\mathcal{C}\|$, using $RSCL$ as opposed to $RSFS$ can translate to a substantially larger ancestor coverage. For example, let $\mathcal{C}_1 = \{r_{ab} = a \rightarrow b, r_{abc} = a \rightarrow b \wedge c\}$. The rules in $\mathcal{C}_1$ are included in two families: $r_{abc}, r_{ab} \in family_{\Omega}(a \rightarrow b)$, and $r_{abc} \in family_{\Omega}(a \rightarrow c)$. Since by definition $RSCL$ does *not* include the ancestor rule, we have that over $\mathcal{C}$, $RSCL_{\mathcal{C}}(a \rightarrow b) = RSCL_{\mathcal{C}}(a \rightarrow c)$. This equality indicates that either ancestor rule could be chosen to be in the ancestor coverage. However, if $a \rightarrow c$ is selected first, the ancestor coverage will include both ancestor rules, even though $\{a \rightarrow b\}$ is the minimal ancestor coverage. Using $RSFS$ ensures a minimal greedy ancestor coverage.

We set the ancestor coverage of a cluster $\mathcal{C}$ through a greedy algorithm using $RSFS$. In each iteration of the greedy algorithm we find the ancestor

rule with the largest $RSFS$ value of the rules remaining to be covered. In other words, let $\mathcal{C}'$ be the set of rules in the cluster $\mathcal{C}$ remaining to be covered: rules that are *not* members of a family of any of the ancestor rules currently in the ancestor coverage. Let $\mathcal{A}_\mathcal{C}$ be ancestor coverage of $\mathcal{C}$. We initialize $\mathcal{C}' = \mathcal{C}$ and $\mathcal{A}_\mathcal{C} = \emptyset$. In each iteration of the algorithm, we find the ancestor rule, ρ, with the largest $RSFS$ value over $\mathcal{C}'$. We then add ρ to $\mathcal{A}_\mathcal{C}$ ($\mathcal{A}_\mathcal{C} = \mathcal{A}_\mathcal{C} \cup \{\rho\}$) and delete from $\mathcal{C}'$ all the rules covered by ρ (rules in the family of ρ spanned over $\mathcal{C}'$). We continue this process iteratively until $\mathcal{C}'$ is empty, and all of $\mathcal{C}$ is covered by the families spanned by the ancestor rules in $\mathcal{A}_\mathcal{C}$. The complexity of finding the ancestor rule with the largest $RSFS$ value over $\mathcal{C}'$ is linear in the size of $\mathcal{C}'$, similarly to the way the ancestor rule with the largest $RSCL$ value over Ω is found in Section 4.4.1. In brief, we calculate the $RSFS$ value of all the possible ancestor rules, $\mathcal{A}$, over $\mathcal{C}'$: $\mathcal{A} = \{a \rightarrow b | \exists \rho = A \rightarrow B \in \mathcal{C}', a \in A, b \in B\}$. Calculating $RSFS_{\mathcal{C}'}(a \rightarrow b)$ for every $a \rightarrow b \in \mathcal{A}$ can be done in a single pass over $\mathcal{C}'$. During the pass, for each rule $A \rightarrow B \in \mathcal{C}'$, we increment a counter value for any ancestor rule $a \rightarrow b : a \in A, b \in B$. All these counters are initialized to one on the first time they are accessed. After a single pass over $\mathcal{C}'$, these counters hold the $RSFS$ (and not $RSCL$) value of each of the ancestor rules in $\mathcal{A}$. A pointer to the ancestor rule with the largest $RSFS$ value is kept updated while modifying the counters in order to eliminate the need for a pass over $\mathcal{A}$. The ancestor rule with the largest $RSFS$ value is chosen as the next member of $\mathcal{A}_\mathcal{C}$ for the iteration.

The ancestor coverage cluster representation is the desired intuitive, entirely data-driven cluster labeling or representation. This inferred cluster labeling is also extremely concise and efficient. For example, one of the larger clusters in the WWW Proxy Logs Database after almost eight thousand iterations, using d_{sc}, contained 1,180 rules, but its cluster coverage contained only 20 ancestor rules. At that level in the hierarchical clustering, the average cluster size was 31.7 rules, and the average cluster coverage only contained 2.4 rules. The relative small size of the ancestor coverages of the clusters remained the same over the other databases, and over the different iterations as well. For example, the average ancestor coverage was 5% of the cluster size for the Adult Database, and just over 18% for the much more sparse Grocery Database. Note that the more dense a database is, the smaller the relative size of the ancestor coverages with respect to the cluster they cover tend to be. The cluster sizes also indicate the density and distribution of the database.

The ancestor coverages are also easy to interpret; ancestor rules are easy

to process, much more so than rules with more attributes (remember that an ancestor rule has only one attribute in its assumption and one in its consequent). Exploration of the clusters via their ancestor coverage is much easier than that of the cluster itself since the ancestor coverage is normally at least an order of magnitude smaller than the cluster. Ancestor rules are easy to classify (Chapter 4), and by quickly reviewing the list of ancestor rules in the ancestor coverage of the rule, users can immediately determine whether they want to further explore the cluster or not. Further exploration of the cluster is enabled through the dendrogram that we build during the hierarchical clustering (this is one of the reasons we chose to use hierarchical clustering). Thus, users can drill down into clusters that are potentially interesting to them, and easily avoid spending time examining not-interesting clusters.

Remember that the ancestor coverage for each cluster is available at each level of the agglomerative hierarchical clustering. That means that the association rules spanned by the coverage of a cluster is not necessarily equal to the cluster itself. For example, let $\mathcal{C}_0 = \{a \rightarrow b\}$, $\mathcal{C}_0' = \{a \rightarrow b \wedge c\}$, and let $\mathcal{C}_1 = \{r_{ab} = a \rightarrow b, r_{abc} = a \rightarrow b \wedge c\}$, as above. $\mathcal{A}_{\mathcal{C}_0}$, the ancestor coverage of $\mathcal{C}_0$ is $\{a \rightarrow b\}$. Assume that in the next iteration $\mathcal{C}_0$ is merged with $\mathcal{C}_0'$ to form $\mathcal{C}_1$. The ancestor coverage of $\mathcal{C}_1$ is still $\mathcal{A}_{\mathcal{C}_0}$ even though $\mathcal{C}_1 \supset \mathcal{C}_0$. That means that the coverage is likely to contain the cluster, and it is problematic to achieve equality of the coverage with the cluster it covers. Since the ancestor coverage is intended as a representation of the cluster, to provide an abstraction or a naming scheme for the cluster, this is not a significant drawback. It still provides users with a general description of the association rules in the cluster which permits the users to quickly determine which of these clusters they would like to further analyze.

6.4.2 Outlier Analysis

An interesting application of clustering is in the discovery of "outlier" association rules, association rules that are significantly different from the rest of the mined association rules (distance based outlier detection is mentioned in [HK01b]). The association rule outliers are manifested in the form of singleton clusters, or very small clusters, during an advanced stage of the hierarchical clustering process, that is, when relatively few clusters remain. These association rules can be very interesting to users as they point out behavior that is unlike the "normal" behavior portrayed in the data. This

kind of interesting behavior is likely to describe a small portion of the population, and therefore is likely not be known to the owners of the database. For example, we discovered, when only 250 clusters remained in the hierarchical clustering of the Grocery Database, an outlier rule that stated that almost a quarter of the people who purchase a 1.5 liter bottle of Coca-Cola will spend between 300 and 400NIS on their shopping basket. During our manual analysis this rule did not stand out. In fact, without the outlier analysis we would have been unlikely to detect it, since it had under 4% support. These kind of rules, with low support, are of interest to many researchers. [HG02, CDF+01, LHM99a] cite examples of domains where infrequent co-occurrences are interesting and suggest varied approaches to this problem.

It is important to note that outliers may reflect noise in data, such as that due to incorrectly entered data, etc. It is therefore possible that users may not want to be presented with outliers if the data is noisy.

6.4.3 Cluster Evaluation

An important part of the clustering process is the evaluation of the results. Although the final evaluation can only be provided by the users, measures of the quality of the clusters, commonly used for partitional clustering, can provide insight as to how legitimate vs. arbitrary the clustering is. The quality measure we have used is the **votes of confidence measure**: the number of pairs of rules within a cluster (prior to its merging with the next cluster) that, according to the similarity measure, should be in the same cluster, that is, be the next candidates to be merged into one cluster that are already in the same cluster. For a measure of goodness we used this number compared to the maximum number of votes of confidence for a cluster of size m: $m(m-1)/2-1$.

6.4.4 Clustering Implementation Issues

There are many challenges in implementing clustering of a large set of association rules. Some were discussed in Section 6.3, and more are reviewed in [HK01b, JMF99]. The biggest challenge we have come across in implementing and running the agglomerative clustering was in the processing-time and memory demands when clustering thousands of association rules (we ran the clustering algorithm for all databases on a Compaq ProLiant DL580, 4 700MHz Xeon CPUs with 2GB RAM). Some clustering algorithms use the

similarity matrix, the symmetric matrix of the $n(n-1)/2$ proximity values. Storing such a matrix, and determining the two most similar items in it over many iterations requires prohibitive amounts of memory and computational power/time. We therefore changed the algorithm and chose to maintain only the similarity measures that are most likely to be used in the hierarchical clustering process to reach 1 cluster from the $\|\Omega\|$ initial singleton clusters. Those are the smaller similarity measures. Their number is obviously larger than $\|\Omega\|$ since some measures will be between members of the same cluster. For example, if two clusters, $\mathcal{C}_i$ and $\mathcal{C}_j$ were joined due to the similarity between $r_1 \in \mathcal{C}_i$ and $r_2 \in \mathcal{C}_j$, and the next two most similar rules were r_2 and $r_3 \in \mathcal{C}_i$, the similarity measure between r_2 and r_3 will not cause a merging of clusters (the two rules are already in one cluster), but will be used to verify the quality of the clusters as in Section 6.4.3. This change reduced the memory requirements and execution time by several orders of magnitude. Note that although this is a minor technical modification to the algorithm (we store only a partial similarity matrix and iteratively select the next smallest distance for each iteration from it), it is critical in making the clustering feasible when clustering a large number, over ten thousand, of rules. These ramifications did not arise in previous works that considered a small number of rules.

6.5 Summary

In this work we presented a cohesive association rule clustering framework for the exploration and understanding of the interestingness of mined rules based on the work published in [Sah02a]. This clustering automates much of the tedious manual labor otherwise needed, and is an alternative to methods that require the availability significant amounts of domain knowledge expressed in predefined grammars or knowledge bases to determine interestingness. We investigated the necessary steps to establish the clustering over general, unconstrained association rules in an unconstrained domain. This is the first work where clustering of a large number of association rules, ranging from a few thousand to over ten thousand, was tackled. We also introduced a new similarity measure that ranks association rules according to all their five defining features, providing a natural dissimilarity measure in the sense that more similar rules are ranked as closer to each other, with a smaller value, and less similar rules are ranked as more distant, with a larger value.

In much the same way that the selection of the objective interestingness criterion used to rank rules imposes a bias on which rules will be marked as interesting (Chapter 7), the similarity measure used for clustering determines the outputted clusters.

We additionally defined the ancestor coverage. The ancestor coverage is a concise, entirely data-driven representation of the clusters. This representation, or labeling, allows a quick identification of each cluster by users, and enables the simple exploration of the cluster schema. Thus, hierarchical clustering provides an automated grouping and naming scheme that enables the exploration of interestingness through the use of the dendrogram and the ancestor coverages, as well as the discovery of outlier association rules, which are potentially interesting rules.

From the analysis clustering of association rules stands to be a very promising and fruitful venue for exploring and understanding interestingness.

Chapter 7

Impartial Interestingness: Interestingness PreProcessing*

> *And one should bear in mind that there is nothing more difficult to execute, nor more dubious of success, nor more dangerous to administer than to introduce a new system of things; for he who introduces it has all those who profit from the old system as his enemies, and he has only lukewarm allies in all those who might profit from the new system.*
>
> —Nicollo Machiavelli in The Prince

Users run the KDD process expecting to receive interesting patterns as the output. Overwhelming users with an output consisting of a myriad of patterns is counterproductive. This is especially true since most of these rules are not interesting to the users. For the KDD process to successfully meet users' expectations, we have seen that an important part of the process must be dedicated to determining which patterns are interesting to its users. In the previous chapters we have investigated diverse approaches to tackling the problem of interestingness. As we will see in this chapter, those approaches have one thing in common: they all rely, to some degree, on user preferences, needs and prior knowledge.

In this chapter we define the Impartial Interestingness Criteria, a new type of interestingness criteria that complements the existing subjective and objective criteria. The impartial criteria form the Interestingness PreProcessing Step which we introduce as part of a new framework for interesting-

*A preliminary version of this chapter was published in the IEEE International Conference on Data Mining (ICDM-2001) [Sah01].

ness analysis. In a similar fashion to data-preprocessing, this preprocessing by the impartial criteria should always be applied before the application of Interestingness Processing.

A strict requirement and the biggest challenge in defining impartial interestingness criteria is in ensuring that this interestingness preprocessing does not eliminate any potentially interesting patterns. To accomplish this impartial interestingness criteria must be domain-independent, task-independent and user-independent. The generic nature of the impartial interestingness criteria differentiates them from existing interestingness criteria, and at the same time makes them very useful; all impartial interestingness criteria can be applied to any task, in any domain, by any user, to eliminate rules that are not interesting from the list of mined rules. This significantly reduces the size of the problem of determining interestingness, and size is the essence of the problem. Additionally, since applying impartial interestingness criteria does not require any user interaction, these criteria can be applied automatically, making even relatively weak impartial criteria very significant. This generic nature of the impartial interestingness criteria makes them very valuable, but generally also rare: they are very challenging to define. In this chapter we will also define the first two impartial interestingness criteria and present the empirical results of their applications to six real databases. The results indicate that the impartial interestingness criteria, and hence the Interestingness PreProcessing Step, are very powerful: in most cases, an average of *half* the rules mined were eliminated by the application of these two impartial interestingness criteria. These results are particularly significant since no user-interaction was required to achieve them.

7.1 Introduction

It is either easy or impossible.

—Salvador Dalí

Characterizing what is interesting is a difficult problem. Numerous attempts have been made to formulate these qualities; they range from evidence and simplicity to novelty and usefulness, as we discussed in Section 1.3. There are many different approaches to determining which of the mined patterns are interesting (which we review in Section 7.3). Determining interestingness according to different interestingness criteria can result in different sets of patterns marked as interesting. This dependence of the output of the interestingness analysis on the interestingness criteria used is clear when domain knowl-

edge is used explicitly (as in [KMR+94, TS96, LHC97, Sah99, PT00, LMY01] etc.)—something that is new and interesting to one user may be common knowledge to another user.

When domain knowledge is applied implicitly in the interestingness criteria, the dependence of the patterns marked as interesting on this domain knowledge is not as clear. For example, *interest* [BMS97] and *AddedValue* (Definition 3 in Section 3.1.1, originally in [SM99]) are two objective interestingness criteria: they use measures that depend only on the structure of the data and the patterns that can be extracted from it. Neither criteria uses any domain knowledge to *rank* the mined patterns according to their respective interestingness, as defined by each criterion. However, the interest ranking imposed by these two criteria on the patterns differs significantly, indicating that the manner of *selecting* which interestingness criteria to use to rank the patterns as interesting is important. The *selection* of which of the interestingness criteria is to be used imposes a bias on the type of rules that will be outputted as interesting. It is therefore an important process that should be made explicitly and not be ignored.

In this chapter we introduce a new type of interestingness criteria: impartial interestingness criteria. Impartial interestingness criteria form the Interestingness PreProcessing Step which is the first step of the interestingness analysis framework we define. Interestingness PreProcessing is applied to the list of mined association rules, as depicted in the interestingness analysis framework in Figure 7.1, before any Interestingness Processing. The Interestingness PreProcessing reduces the size of the problem, that is, the number of potentially interesting rules that need to be subsequently processed for interestingness. Since size is the essence of interestingness problem this is an important step in the interestingness analysis process. We show through an empirical evaluation the powerful results of the application of impartial interestingness criteria. Note that Interestingness PreProcessing eliminates *only* rules that are *not* potentially interesting, without imposing any bias on the type of rules that will be outputted.

The importance of the impartial interestingness criteria (the Interestingness PreProcessing Step), and what differentiates them from the Interestingness Processing criteria (reviewed in Section 7.3), is their generic nature. The elimination of not-interesting association rules during the Interestingness PreProcessing step, by impartial criteria, is done independently of:

1. The data used in the mining process,

2. The domain of the problem,

3. The task addressed by the KDD process, and,

4. The specific interests, needs and prior knowledge of the user.

These criteria are **impartial** to all these considerations and hence their name: **impartial interestingness criteria**.

As we will see in Section 7.4, all existing interestingness criteria fall under the Interestingness Processing Step: selecting and/or using any of these criteria requires some sort of human intervention. That means these criteria impose a bias on the type of rules that will be outputted as interesting. Domain knowledge needs to be applied to make a judicious selection of the (one or more) interestingness criteria to be used to determine which patterns are interesting. Making this selection randomly, that is, selecting the interestingness criteria without conscious deliberation, may result in the interestingness analysis producing only partial results that do *not* include all the patterns that users may find potentially interesting. Since we know that only very few of the mined patterns are interesting, it is important not to erroneously dismiss any of the interesting patterns during the analysis. The impartial interestingness criteria do not require any kind of human intervention, passive or active. This yields an enormous benefit; the impartial interestingness criteria can be *automatically* used following any association mining process, before any Interestingness Processing is applied, and therefore they are **Interestingness PreProcessing** criteria.

The introduction of the new impartial interestingness criteria and the Interestingness PreProcessing Step into an Interestingness Framework, as in Figure 7.1, is our first contribution in this chapter. Our second contribution is in defining the first two impartial interestingness criteria. Since impartial interestingness criteria do not impose any bias on the rules they mark as potentially interesting, they can be run sequentially, one after the other. This generic nature of the impartial interestingness criteria makes them rare: it is difficult to define criteria that can be applied to all cases without setting any thresholds or imposing a preference to a specific type of rules. Because of their generic nature, these criteria may be perceived as weak, that is, having less power than objective or subjective interestingness criteria in eliminating rules that are not interesting. That is *not* necessarily the case. To demonstrate the power of the impartial interestingness criteria, we also provide an empirical evaluation of the results of the application of the two impartial

criteria on six real databases. These results are very encouraging yielding in almost all cases an average deletion of more than half of the mined rules.

We stress that the impartial interestingness criteria are meant to complement the existing types of interestingness criteria and not to replace them. The Interestingness PreProcessing Step during which the impartial interestingness criteria are executed is the first step towards the comprehensive solution to determining which rules are interesting, independently of the domain, task and user, by eliminating some of the rules that are not interesting, as in Figure 7.1. The overall problem of determining, in any domain, precisely which rules are interesting will require the incorporation of subjective interestingness criteria, since what is interesting is ultimately subjective. Our goal, described in the Intuitive Problem Statement of Interestingness (Section 7.2), is to reduce the size of the interestingness problem by eliminating a significant portion of the not-interesting rules. As we explain in Section 7.2, this goal is different from the goal of summarization.

This chapter is organized as follows: in Section 7.2 we scope the problem that we address in this chapter, intuitively define the problem statement we tackle and emphasize why this scope *precludes* summarization. This is an important step since for any progress in the interestingness analysis to be made, we must first define the ground rules or the scope of the problem. We review the available interestingness processing methods in Section 7.3. In Section 7.4 we show the explicit and/or implicit dependencies of these methods on domain or prior knowledge, and define the interestingness analysis framework that includes the new impartial interestingness criteria in the Interestingness PreProcessing Step. We introduce the two first impartial interestingness criteria in Section 7.5, and present the powerful empirical results of their application on six different databases in Section 7.6. We conclude this chapter with a summary in Section 7.7.

7.2 Interestingness: Intuitive Problem Statement

Wisdom begins in wonder.

—Socrates

Throughout this work we have seen several approaches to the problem of interestingness. In Chapter 6 we presented an approach that groups similar association rules together to facilitate the exploration and understanding

of their interestingness. [LHH00] argue *"that the key problem is not with too many rules, but with our inability to organize and present the rules in such a way that is easily analyzed by the user"* and present a technique to summarize the mined rules. Summarization is currently one of the more popular methods to reach a concise organization of mined rules that is then presented to users, with many approaches falling under this method (Section 7.3.2). These approaches share an important common characteristic: *their output needs to be further analyzed and processed by users in order to explore, understand and extract the interesting rules.*

While summarization can facilitate the interestingness exploration process, our ultimate goal is to *reach the short list of interesting rules*; our goal here is *not* to present users with a solution that will, eventually, enable them to *generate* the interesting rules. This is an important distinction as portrayed in Example 7.2.1.

Example 7.2.1 *We consider the following three rules:* $r_{cv} = \langle \texttt{crime} \rangle \rightarrow \langle \texttt{victim} \rangle$ *that states that every crime has a victim (since by definition criminal law is designed and applied to protect victims) where* confidence$(r_{cv}) = 1$, $r_s = \langle \texttt{crime} \rangle \rightarrow \langle \texttt{no unhappy parties} \rangle$, *where we have* confidence$(r_s) < 1$ *(since there are cases where criminal law is applied when all the involved parties are satisfied and no victim has come forth), and* $r_{vlc} = \langle \texttt{crime} \rangle \rightarrow \langle \texttt{victim} \wedge \texttt{no unhappy parties} \rangle$. *The scenario described by* r_{vlc} *is often referred to as the* ***victimless crime scenario*** *and it invokes great interest and discussion. The other two rules,* r_{cv} *and* r_s*, constitute a mathematically correct summary of* r_{vlc} *from which* r_{vlc} *could be generated. However, it is* r_{vlc} *that is the interesting rule—that describes the interesting scenario—whereas each of the summarizing rules,* r_{cv} *and* r_s*, by itself is known and* not *interesting.*

Applying formal redundancy definitions will often result in summaries. Examine Definition 6, a very comprehensive and yet concise definition for redundant rules, according to which the interesting rule in Example 7.2.1, r_{vlc}, is redundant and will *not* be included in the list of outputted rules, whereas the two *not*-interesting rules will be included. Our goal in this work, which we state below in the Intuitive Problem Statement of Interestingness, is to approach the list of interesting rules by discovering a superset of the set of interesting rules. Removing the victimless crime scenario rule (the "redundant" rule) *contradicts* this goal. We will therefore need to refrain

from applying such definitions in this work.

Definition 6 *Let Ω be a set of triplets $\langle r, c, s\rangle$ where r is a rule, s its support and c its confidence. A dataset $\mathcal{DB}$ supports Ω if for every $\langle r, c, s\rangle \in \Omega$, r has support s and confidence c in $\mathcal{DB}$. Given Ω a rule r with support s and confidence c is* ***redundant*** *if for any dataset that supports Ω, the dataset also supports rule r with support s and confidence c.*

Summarization can provide many benefits, but it forces us away from our goal of procuring the short list of interesting rules. Example 7.2.1 is not unique in this respect. Two more examples of how summarization precludes the interesting rule are provided in Examples 7.2.2 and 7.2.3.

Example 7.2.2 *From a completely different domain, consider the following three rules:*

- *A sturgeon* always *fails to digest plant protein ($r_{f1} = \langle$`sturgeon`$\rangle \rightarrow \langle$`cannot digest plant protein`$\rangle$), since sturgeons are true carnivores and do not have the correct enzymes to digest plants,* confidence$(r_{f1}) = 1$.
- *A sturgeon* sometimes *eats plant protein ($r_{f2} = \langle$`sturgeon`$\rangle \rightarrow \langle$`eats plant protein`$\rangle$ where* confidence$(r_{f2}) < 1$*), and, finally,*
- $r_{f3} = \langle$`sturgeon`$\rangle \rightarrow \langle$`eats plant protein` $\wedge$ `cannot digest plant protein`$\rangle$.

The mathematically correct summary of the three rules is $\{r_{f1}, r_{f2}\}$ from which r_{f3} can be generated. r_{f1} and r_{f2}—each by itself—are not interesting, but r_{f3} is *interesting: it states that sometimes sturgeon eats plant protein even though it does not benefit the fish, a very intriguing proposition.*

Example 7.2.3 *For an illustrative market basket analysis example, consider the following three rules: $r_1 = \langle$`peanut butter`$\rangle \rightarrow \langle$`bread`$\rangle$, $r_2 = \langle$`peanut butter`$\rangle \rightarrow \langle$`jelly`$\rangle$, and $r_3 = \langle$`peanut butter`$\rangle \rightarrow \langle$`jelly`$\wedge$`bread`$\rangle$, such that* $\boldsymbol{Pr}[$`bread`$] = 0.61$, $\boldsymbol{Pr}[$`peanut butter`$] = 0.4$, $\boldsymbol{Pr}[$`jelly`$] = 0.32$, $\boldsymbol{Pr}[$`peanut butter`$\wedge$`jelly`$] = 0.3$, $\boldsymbol{Pr}[$`bread`$\wedge$`peanut butter`$] = 0.4$, $\boldsymbol{Pr}[$`bread`$\wedge$`jelly`$\wedge$`peanut butter`$] = 0.3$. *We have* confidence$(r_1) = 1$ *and* confidence$(r_2) =$ confidence$(r_3) = 0.75$. *A mathematically correct summary of the three rules would be $S = \{r_1, r_2\}$ since r_3 can be generated from S. However, when*

presented with r_3 a marketing person can speculate with great confidence that in the United States r_3 describes shoppers that are going to prepare peanut butter and jelly (PB&J) sandwiches. The marketing person can therefore use r_3 to cross-sell other products that PB&J sandwich consumers are likely to purchase (for example, by placing them, or coupons for them, next to or on the way to the peanut butter section). r_3 is therefore interesting to this marketing person. From the summary of the three rules, S, r_3 can be generated, but since the PB&J sandwich consumption cannot be assumed from S, r_1 and r_2 are not interesting. To provide the marketing person the short list of interesting rules we would need to include r_3. The summary of the rules can useful for other purposes, but not in order to produce the short list of interesting rules as an output.

To differentiate and scope the general problem we are addressing here, we provide the following problem statement:

Intuitive Problem Statement of Interestingness: *Let Ω be the set of association rules outputted by a mining algorithm run with support and confidence thresholds of s and c. Find $\widehat{\Omega} \subseteq \Omega$, the set of "interesting rules"*[†].

It is important to note that we are addressing the problem of finding a subset of interesting rules. We do not attempt to infer from rules in Ω rules that have not been mined, that is, that do not appear in Ω, that could be of interest to the user. This is a supposition often made implicitly. Since it is the only supposition we make of a user's interests, we make it explicitly here.

Note that finding the exact subset, $\widehat{\Omega}$, that contains only interesting rules, is the ultimate goal of the interestingness solution. Finding a subset of the mined rules, $\widetilde{\Omega}$, such that $\widehat{\Omega} \subseteq \widetilde{\Omega} \subset \Omega$[‡], is our goal in introducing the impartial interestingness criteria as the Interestingness PreProcessing Step: we seek to eliminate rules that are not-interesting from the exhaustive list of rules outputted by a data mining algorithm. That is, our objective is to decrease the size of the problem, and size in KDD is the essence of the problem. While summarization can be helpful to users in exploring the rules, it is important

[†]We do not need to formally define which rules fall under the "interesting rules" category, thus circumventing a difficulty on a philosophical level which has been baffling researchers and impeding progress in this area.

[‡]Since we do not use any subjective interestingness criteria and interestingness is ultimately subjective, we know that $\widetilde{\Omega} \neq \widehat{\Omega}$.

to remember that we confine our solution space to those conforming to finding $\widetilde{\Omega}$ as in the Intuitive Problem Statement of Interestingness presented above; this means that our goal does *not* include finding a summary $\breve{\Omega} \subset \Omega$ from which $\widehat{\Omega}$ can be generated. As we saw in Examples 7.2.1–7.2.3, the summary may not include all the potentially interesting rules: $\breve{\Omega} \not\subset \widehat{\Omega}$.

The second important point is that there are types, or classes, of rules that are widely accepted by the KDD community as not interesting, for example, if they are *less* actionable than other rules that users already have. Reducing the predictability of a rule makes for a less actionable (interesting) rule, and is a fundamental concept in determining what association rule is interesting. [BJAG99] illustrated this by using the example of two pairs of rules. The first pair was $r_{1,1} = \langle \texttt{bread} \wedge \texttt{butter} \rangle \rightarrow \langle \texttt{milk} \rangle$ with $\mathit{confidence}(r_{1,1}) = 0.8$, and $r_{1,2} = \langle \rangle \rightarrow \langle \texttt{milk} \rangle$ with $\mathit{confidence}(r_{1,2}) = 0.85$. For a user interested in understanding the population of milk buyers, $r_{1,1}$ could be a potentially interesting rule, and the relationship between people who purchase bread and butter and people who purchase milk may be a good topic for market research. However, given $r_{1,2}$ that says that 85% of the people who walk into the store buy milk anyway, $r_{1,1}$ *"is quite uninteresting for this purpose since it characterizes a population that is even less likely to buy milk than the average shopper"* [BJAG99]. It is entirely possible that from $r_{1,1}$ and $r_{1,2}$ we may be able to derive new rules (that have not been mined) that pinpoint at a subset of the population that does *not* purchase butter that is more likely to purchase milk, but that is outside the scope of the Intuitive Problem Statement of Interestingness presented above.

Similarly, [BJAG99] used another pair of rules: $r_{2,1} = \langle \texttt{eggs} \wedge \texttt{cereal} \rangle \rightarrow \langle \texttt{milk} \rangle$ where $\mathit{confidence}(r_{2,1}) = 0.95$ and $r_{2,2} = \langle \texttt{cereal} \rangle \rightarrow \langle \texttt{milk} \rangle$ such that $\mathit{confidence}(r_{2,2}) = 0.99$. Now, $r_{2,1}$ has significantly higher confidence than the a priori probability that milk is purchased ($r_{1,1}$), and therefore it seems to be potentially interesting (with respect to $r_{1,2}$). However, $r_{2,2}$ states that the purchase of cereal *alone* implies that milk is purchased with 0.99 confidence, more than $r_{2,1}$'s predictive ability, changes how interesting $r_{2,1}$ is. It means that the population of milk buyers is better reached through $r_{2,2}$ than through $r_{2,1}$. Again, given $r_{2,1}$ and $r_{2,2}$, we may be able to *derive* or *generate* a *new* rule (that has not been mined) that pinpoints to a sub-population segment that includes customers who purchased cereal but did not purchase eggs, that person is even more likely to buy milk. Remember, *deriving* such a rule is outside the scope of the Intuitive Problem Statement of Interestingness presented above, and $r_{2,2}$ is *more* interesting than $r_{2,1}$ for users *within*

the rules in Ω, the exhaustive list of mined rules. This scoping of the problem, done in the Intuitive Problem Statement of Interestingness, is necessary because without it we will not be able to rule out *any* rule as not interesting, since even false rules could be potentially interesting.

It is also worth noting that Definition 6 cannot eliminate the not-interesting rules that are considered redundant and do need to be eliminated such as $r_{1,1}$ and $r_{2,1}$ in the example from [BJAG99]. So we have on the one hand interesting rules that are deleted by such redundancy definitions that should *not* be deleted (as in Examples 7.2.1–7.2.3), and on the other hand, not-interesting rules that *should* be deleted by them and are not. We will therefore refer to the Intuitive Problem Statement of Interestingness throughout the reminder of this chapter and refrain from using redundancy definitions such as Definition 6.

7.3 Interestingness Processing Techniques

In this section we review the types of existing Interestingness Processing techniques. These are the techniques, the impartial interestingness criteria, used in the second part of the interestingness framework we outline in Figure 7.1.

7.3.1 Explicit Use of Domain Knowledge

There are three main approaches to the *explicit* use of domain knowledge which includes specific user interests, prior knowledge and current needs. The first two approaches are reviewed in Section 4.1.1. The third approach is the Interestingness Via What Is Not Interesting approach, introduced in Chapter 4, originally in [Sah99]. See Section 4.1.3 for a differentiation of the three subjective interestingness approaches through an analogy.

7.3.2 Implicit Use of Domain Knowledge

Many different approaches use domain knowledge *implicitly*. These approaches do not compel users to specify directly what is interesting to them, but require information that only a user can provide. This information ranges from how to select the interestingness criteria that are applied, to the initialization of parameters used by these criteria. We examine the existing methods in this category below. Note that the methods that fall under this category are

normally referred to as objective interestingness criteria, whereas methods that use domain knowledge explicitly are commonly referred to as subjective interestingness criteria.

Ranking Patterns

To rank association rules according to their interestingness, a mapping, f, is introduced from the set of mined rules, Ω, to the domain of real numbers,

$$f : \Omega \rightarrow \mathbb{R}. \tag{7.1}$$

The number an association rule is mapped to is an indication of how interesting the association is; the larger the number a rule is mapped to, the more interesting the rule is assumed to be. There is a wide variety of these mappings, or ranking criteria, presented and discussed in Chapter 3.

Pruning and Application of Constraints

The mappings in Equation 7.1 can also be used as pruning techniques: all the association rules mapped to an interest score lower than a user-defined threshold are pruned or filtered out as not interesting. Note that in this section we only refer to pruning and application of constraints performed using objective interestingness criteria. Explicit application of domain knowledge for interestingness pruning and constraints (for example, attributes or components of rules defined by a user as interesting) falls under the subjective interestingness methods of Section 7.3.1.

Additional methods can be used to prune patterns, methods that do not require the interest mapping criteria. Statistical tests such as the χ^2 test are used to prune in [BMS97, LHM99b, LHM01b]. These tests have associated parameters (significance, confidence, etc.) that need to be initialized.

A collection of instrumental pruning methods is detailed in [SLRS99]. These methods require the initialization, by a user, of several parameters (error, strength, etc.).

Another type of pruning is a constraint based approach as in [BJAG99]. [BJAG99] present a novel algorithm to overcome the exponential explosion of frequent (large) itemsets in dense datasets. The algorithm only mines rules that comply with the two usual constraints of minimum support and confidence thresholds, and two new constraints; the first constraint is that

all the mined rules have a user-specified consequent (subjective interestingness). The second, unprecedented, constraint is of a user specified minimum confidence improvement threshold. Only rules whose confidence is at least the minimum confidence improvement threshold greater than the confidence of any of their simplifications are outputted, where a simplification of a rule is formed by removing one or more attributes from its assumption. Note that the goal of [BJAG99] is to create a more concise list of rules as the output of the mining algorithm.

Summarization of Patterns

Several distinct methods fall under the summarization approach. [AY98b, AY98a] suggest a redundancy measure that summarizes all the rules at the *given* levels of support and confidence very compactly using the more "complex" rules. [AY98a]'s notion of redundancy only has to do with being able to summarize all the rules at the *given* levels of support and confidence in a very compact form. The preference to complex rules is formally defined as: given rules $A \rightarrow B$ and $C \rightarrow D$, $C \rightarrow D$ is redundant with respect to $A \rightarrow B$ if (1) $A \cup B = C \cup D$, and $A \subset C$, or (2) if $C \cup D \subset A \cup B$ and $A \subseteq C$. Under the assumption that the user has no preference to higher support and confidence levels than those specified, this summary includes a comparatively small number of rules that summarizes the entire list of rules without losing any information at all. The user can then review this relatively small list of rules to gain insight.

[LHM99b]'s summary of association rules with a single attributed consequent includes "direction-setting" rules, rules that represent the direction a group of non-direction-setting rules follows. The direction is calculated using the χ^2 test, which is also used to prune the original list of rules outputted by the mining algorithm prior to the discovery of direction-setting rules. This technique favors "less-complex" rules. An intuitive summary that simplifies the discovered rules by providing an overall picture of the relationships in the data and their exceptions is presented in [LHH00]. The novel approach of [Zak00], that only mines non-redundant association rules, also favors "less-complex" rules. [Zak00] defines a rule $r' = A' \rightarrow B'$ as redundant if there exists another rule $r = A \rightarrow B$ such that $A \subseteq A'$ and $B \subseteq B'$ and both rules have the same confidence. Note that [Zak00] generates a set of "generating rules" from which all the other rules can be inferred.

[TKR+95] introduce effective approaches to improve the understandabil-

ity of the mined rules. They suggest clustering rules *"that make statements about the same database rows [...]"*, using a simple distance measure. [TKR+95] also introduce an algorithm, the `Algorithm RuleCover`, to compute rule covers as short descriptions of large sets of rules. For this approach to work without losing any information, [TKR+95] makes a monotonicity assumption, or restriction, on the databases on which the algorithm can be applied: *"The data set should be monotonic in the sense that if there is a matching rule for Y* [the consequent] *with a high confidence, there should not exist a more special rule with a lower confidence. This is to ensure that we do not miss any contradictory information by limiting ourselves to the rules with high confidence only."*

A novel and powerful summarization approach through similarity based rule grouping is introduced in [AT01, AT99]. The similarity measure is specified though an attribute hierarchy, organized in a tree format by a human domain expert. The domain expert also specifies a level of rule aggregation, called a **cut**, which is a subset of the tree nodes *"such that for every path from a leaf node to the root, exactly one node on such path belongs to this subset"* [AT01]. Each association rule, r, is then translated into an **aggregated rule** by mapping the attributes in r to the corresponding nodes in the cut without repetitions of cut-nodes (when more than one attribute is mapped to the same cut-node) in a conjunction, the assumption or consequent. Rules mapped to the same aggregated rule are similar, and grouped together. The summary of the list of mined rules is then the set of all aggregated rules, providing a condensed, understandable summary of *all* the mined rules, without any loss of information, based on an expert defined attribute hierarchy and cut.

7.4 Dependencies of Interestingness Processing Criteria and the Independent Interestingness PreProcessing

> *The opposite of a correct statement is a false statement. But the opposite of a profound truth may well be another profound truth.*
>
> —Niels Bohr

To determine which patterns are interesting, users need to decide which interestingness criterion (or criteria) to use to determine interestingness.

The selection of which criterion is to be used shows a preference to a specific type rules whether the criterion uses domain knowledge explicitly or implicitly. However, this selection is often preformed without considering how it will affect the patterns chosen as interesting.

We examine an example of how a choice of which ranking criterion to use for interestingness ranking imposes a preference on the patterns ranked as interesting. The *lift* or *interest* ranking criterion, defined in [BMS97] as $\mathit{interest}(A \rightarrow B) = \mathit{confidence}(A \rightarrow B)/\mathbf{Pr}[B]$, ranks highly association rules with high confidence and low a priori probability of the consequent. This characteristic is not unique to the *interest* criterion. The *AddedValue* criterion, introduced in Definition 3 in Section 3.1.1 (originally in [SM99]), $\mathit{AddedValue}(A \rightarrow B) = \mathit{confidence}(A \rightarrow B) - \mathbf{Pr}[B]$, shares this property. And yet, there is a significant difference between the order these two criteria impose on the rules they rank. One difference is derived from the *interest* criterion's indiscrimination to the same differences in orders of magnitude in the values of the confidence and the a priori probability of the consequent, as in Example 7.4.1.

Example 7.4.1 *We examine two association rules $A \rightarrow B$ and $C \rightarrow D$ such that $\mathit{confidence}(A \rightarrow B) = 0.5$, $\boldsymbol{Pr}[B] = 0.2$, and $\mathit{confidence}(C \rightarrow D) = 0.05$, $\boldsymbol{Pr}[D] = 0.02$. For these rules, we have: $\mathit{interest}(A \rightarrow B) = 0.5/0.2 = 0.05/0.02 = \mathit{interest}(C \rightarrow D)$, that is, the interest criterion will rank these two rules as equally interesting. However, $\mathit{AddedValue}(A \rightarrow B) = 0.5 - 0.2 = 0.3 > 0.03 = 0.05 - 0.02 = \mathit{AddedValue}(C \rightarrow D)$, indicating that the AddedValue criterion will rank the same two rules as having substantially different interest levels.*

This indifference to orders of magnitude, portrayed in Example 7.4.1, may be exactly what some users are looking for. However, a user unaware of this property may fail to choose the subjectively beneficial objective interestingness criterion in his/her particular case. The same applies to other differences between the two criteria, such as *interest*'s symmetric treatment of the assumption and consequent.

User preferences, or choices, are implied by a user's selection of an interestingness criterion. When a user decides to use summarization (a subjective decision in itself), further consideration needs to be given to what kind of summary is preferred. Are the "more complex" rules suggested in [AY98a] to be favored over a summary such as the one suggested by [BJAG99] or

[LHM99b]?

Some interestingness processing approaches require initialization of thresholds (for example, improvement threshold, type-I error level for the χ^2 test, etc.). These thresholds are set by users, implicitly requiring the use of domain knowledge. Defaulting to commonly used threshold values, for example, 95% significance level for the χ^2 test, is still a decision that will influence which rules are outputted as interesting.

The χ^2 test is a popular pruning method [BMS97, LHM99b, LHM01b]. Selecting to use it, as opposed to another method, is a decision that will affect the rules outputted as interesting. As explained in [BMS97], the use of the χ^2 test requires conditions that do not always hold, and thus, cannot be used in all cases. There is also the acute problem of multiplicity: when performing a series of tests at a given significance level, the *overall* probability of making at least one type-I error can be much higher than the predetermined level. When using the χ^2 test, one should also be aware of the type-II errors since only a minute fraction of the rules is expected to be interesting. Remember that type-I error (normally denoted by α) occurs when a true null hypothesis is rejected erroneously. Type-II error (usually denoted by β) is said to occur when a false null hypothesis is accepted erroneously (in this case, when a rule is considered not-interesting when it is interesting).

User preferences are sometimes implicitly manifested in the selection of an interestingness criterion used—through an assumption made within the chosen approach. For example, [TKR+95] introduce a domain independent method of reducing the number of mined rules. However, to ensure *"that the pruning methods do not lose any information, one has to assume the following. The data set should be monotonic in the sense that if there is a matching rule for Y with high confidence, there should not exist a more special rule with lower confidence"* [TKR+95]. Hence, even though the method is domain independent, it is not task independent. Choosing such a method dictates an implicit user choice. The same applies to the pruning rules described in [SLRS99]; the diverse and highly effective pruning rules are domain independent, but not user independent. Thresholds need to be set by users before the pruning can be executed.

A diverse collection of interestingness processing criteria is available to users, with different criteria potentially yielding different results. For a given KDD process users are then confronted with the task of choosing the subjectively "right" interestingness criterion from this diverse collection. This task is far from trivial. As summarized in Table 7.1, this task can require

Interestingness Method Used	Output affected by		
	criterion selection	initial-ization	explicit user interests
Explicit methods (Section 7.3.1)	Yes	Yes	Yes
Implicit methods (Section 7.3.2)			
Ranking Patterns	Yes	No	No
Pruning & Constraints	Yes	Yes	No
Summarization	Yes	No	For analysis of results
Impartial (PreProcessing) methods	No	No	No

Table 7.1: **Dependency of Interestingness Methods on User Preferences.** *The right multicolumn lists the different aspects of a choice of an interestingness method (in the rows) that affect the output of that method: selection, initialization and explicit user preferences. of method.*

the implicit use of domain and prior knowledge to (1) select the interestingness criterion used, and, (2) initialize the relevant threshold of that selected criterion. In some cases, explicit use of domain knowledge is also required (note that it is required for the analysis of the results of the summarization approach). These interestingness processing methods therefore cannot be used indiscriminately by any user, in any domain, for any task.

We now define **Impartial Interestingness Criteria** to be domain-independent, user-independent and task-independent measures of interest that eliminate not-interesting rules (in accordance with the Intuitive Problem Statement of Interestingness presented in Section 7.2). This impartiality to all consideration lends the impartial criteria their name. The impartial criteria constitute the Interestingness PreProcessing Step.

The **Interestingness PreProcessing Step** preprocesses the mined patterns to eliminate patterns that can be determined to be not-interesting through an impartial interestingness criterion, that is, independently of the domain, user and task. No information is required by the Interestingness PreProcessing Step to perform this analysis, other than the Intuitive Problem Statement of Interestingness (Section 7.2). The Interestingness PreProcessing Step therefore precedes the Interestingness Processing Step (and hence its name), and follows directly after the data-mining process as depicted in

the interestingness framework outlined in Figure 7.1.

Techniques that fall under the Interestingness PreProcessing Step are called impartial interestingness criteria. The execution of all the impartial interestingness criteria constitutes the Interestingness PreProcessing Step. Similarly, objective and subjective interestingness criteria fall under the Interestingness Processing Step. The execution of all the objective and subjective interestingness criteria constitutes the Interestingness Processing Step.

By definition, the impartial interestingness criteria do not impose bias on the rules they output as potentially interesting, or on the nature of "interestingness" in accordance with the Intuitive Problem Statement of Interestingness (Section 7.2). Impartial criteria (and Interestingness PreProcessing) do not make any assumption on the data, domain or user, and do not require any user interaction.

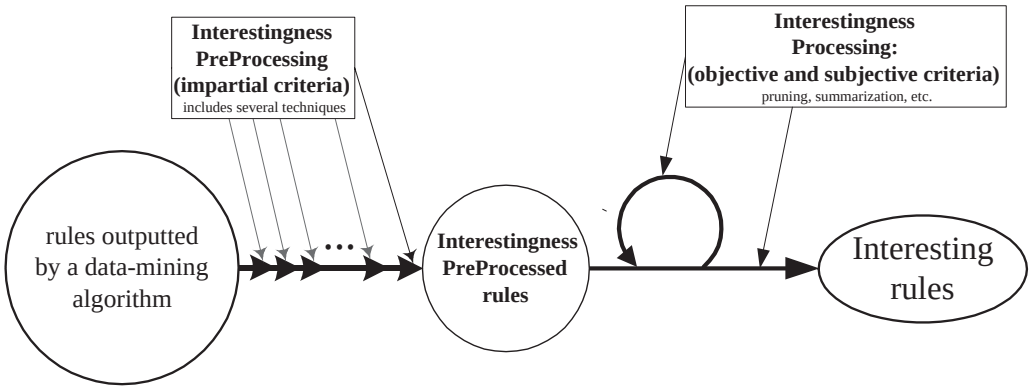

Figure 7.1: **Framework for Determining interestingness**. Impartial interestingness criteria, the criteria that constitute the Interestingness PreProcessing Step, do not require any domain, user or task information, and should always be used to decrease the number of rules that need to be subsequently processed for interestingness. Interestingness PreProcessing therefore precedes the Interestingness Processing that uses, implicitly and/or explicitly, domain or prior knowledge through the objective and subjective criteria.

The impartial interestingness criteria have several benefits:

1. First, and most pragmatically, users can apply all impartial interestingness criteria (as part of the Interestingness PreProcessing Step) *automatically*, directly after the mining process, to all problems conforming

to the Intuitive Problem Statement of Interestingness (Section 7.2). This enables a significant reduction in the size of every case of the interestingness analysis problem, without imposing any bias on the type of rules outputted.

2. Running the impartial interestingness criteria comes with zero cost to users since these methods do not depend on any user interaction or user-information. There is therefore no downside to applying them automatically following every mining process.

3. The output of the Interestingness PreProcessing Step—rules that were not filtered out by the impartial interestingness criteria—does not require appraisal by users, as it is in other approaches, for example, summarization.

4. The output of the impartial interestingness criteria can be used as the input for interestingness processing methods, such as ranking, summarization, etc.

5. The impartial interestingness criteria (the Interestingness PreProcessing Step) circumvent the pitfall of a precise definition of what is interesting.

7.5 Impartial Interestingness Criteria

In this section we introduce the first two impartial interestingness criteria, the first two techniques that fall under the Interestingness PreProcessing Step.

7.5.1 Impartial Interestingness Through Overfitting: Interestingness PreProcessing 1

The first impartial interestingness criterion we introduce filters out overfitting by capitalizing on the basic characteristic of association rules: the implication as captured by the *confidence* of the rules. The confidence value of an association rule is not the best predictor for the interestingness of the rule; *confidence* does not take into account the a priori probability of the consequent. A rule may have a high confidence level, but this confidence value may be equal to the a priori probability of the consequent, rendering the

association rule meaningless. Yet, the confidence criterion does capture an inherent property of association rules that may be exploited in some cases as detailed below.

Let the task to whose solution we are trying to approach be in accordance with the Intuitive Problem Statement of Interestingness in Section 7.2.

Impartial Interestingness Criterion 1 : Overfitting. *The application of the overfitting impartial interestingness criterion to* Ω *consists of the deletion of any rule* $r = A \cup C \rightarrow B \in \Omega$ *if there exists a rule* $\widehat{r} = A \rightarrow B \in \Omega$ *such that:* $confidence(\widehat{r}) \geq confidence(r)$. *Of course,* $support(\widehat{r}) > support(r)$.

Verification: To show that the criterion above is an impartial interestingness criterion we need to show that any rule that this criterion eliminates from Ω is a not-interesting rule, that is, according to the Intuitive Problem Statement of Interestingness of Section 7.2: a rule that does not provide any more information than rules already in Ω. This follows directly from the definition of when rules are deleted according to this criterion. We examine two rules, $r_1, r_2 \in \Omega$ such that $r_1 = C \rightarrow D$, $r_2 = A \rightarrow B$ where $A \subset C$ (that is, $A \neq C$), $B = D$, and $confidence(r_2) \geq confidence(r_1)$. In the general case, given two rules with the same consequent and assumptions that strictly contain each other, not much can be concluded without knowing what types of rules the KDD users are interested in. However, *when* $confidence(A \rightarrow B) \geq confidence(C \rightarrow D)$, the rule $C \rightarrow D$ does not provide more information (in accordance with the Intuitive Interestingness Problem Statement of Section 7.2) than $A \rightarrow B$ since we are adding more attributes to the assumption, A, to form a larger assumption, C, and diminishing the predictive power (confidence). In this case, given $A \rightarrow B$, we can delete $C \rightarrow D$, as we wanted to show.

Note that this becomes very clear in the "extreme" case (also the "best case" scenario for r) where $confidence(r) = confidence(\widehat{r})$. In this case r has more constraints in its assumption than $\widehat{r}$, but its predictability power of the consequent remains the *same* as $\widehat{r}$. Therefore to predict B we will have a bigger population segment by using the more general $\widehat{r}$ than by targeting the population segment defined by r. ■

Example 7.5.1 *Given two rules* $r = \langle \texttt{milk} \wedge \texttt{shoes} \rangle \rightarrow \langle \texttt{cereal} \rangle$ *and* $\hat{r} = \langle \texttt{milk} \rangle \rightarrow \langle \texttt{cereal} \rangle$ *such that* $\boldsymbol{Pr}[milk] = 0.8$, $\boldsymbol{Pr}[cereal] = 0.6$, $\boldsymbol{Pr}[shoes] = 0.5$, $\boldsymbol{Pr}[milk \wedge cereal] = 0.4$, $\boldsymbol{Pr}[milk \wedge shoes] = 0.4$, $\boldsymbol{Pr}[cereal \wedge shoes] = 0.3$, *and* $\boldsymbol{Pr}[milk \wedge cereal \wedge shoes] = 0.2$, *r will be deleted by the overfitting impartial*

interestingness criterion as it provides no more information (Section 7.2) than $\hat{r}$ (both rules have the same confidence level 0.5). This is an illustrative example.

Note that this impartial interestingness criterion does *not* eliminate a rule $r_1 = C \rightarrow D$, $r_2 = A \rightarrow B$ such that $A = C$ and $B \subset D$ (that is, $B \neq D$). In this case we have, $\mathit{confidence}(A \rightarrow B) = \mathbf{Pr}[A \cup B]/\mathbf{Pr}[A] > \mathbf{Pr}[A \cup D]/\mathbf{Pr}[A] = \mathit{confidence}(C \rightarrow D)$. The relationship between the respective confidence values and the a priori probabilities of the consequents can be measured in many ways. We examined two such ways in Section 7.4, *interest* and *AddedValue* (more ways are listed in [SM99, BJAG99, TKS02]). A possible scenario could be that the a priori probability of B is so large compared to that of D, that the rule $A \rightarrow B$ does not give significantly more information than $\rightarrow B$, while $A \rightarrow D = C \rightarrow D$ does. An impartial interestingness criterion needs to possess the property that it can be applied to all cases, without requiring any assumptions or information from users, such as a user definition for "significantly more information" or thresholds that would define formal improvement of information (as in [BJAG99, SLRS99]). That is why in this case, the overfitting impartial interestingness criterion does *not* eliminate either rule, r_1 or r_2. Following the same reasoning, for $r_1 = C \rightarrow D$, $r_2 = A \rightarrow B$ such that $A \subset C$ and $B \subset D$, no deletions are performed by this impartial interestingness criterion to preclude, in the general case, the elimination potentially interesting association rules.

Rules that are eliminated by this impartial interestingness criterion are, as we saw above, not-interesting rules as in Section 7.2: rules that do not provide any more information than rules that already exist in Ω. In effect, this preprocessing eliminates overfitting and hence its name.

The empirical results of the application of this preprocessing are very powerful: in most cases it eliminated an average of half the rules. See Section 7.6.1 for detailed results.

7.5.2 Impartial Interestingness Through Transition: Interestingness PreProcessing 2

In this section we introduce the second impartial interestingness criterion.

Impartial Interestingness Criterion 2 : Transition. *The application of the transition impartial interestingness criterion consists of the deletion*

of any rule r, $r = A \rightarrow C \in \Omega$, *for which* $\exists r_2, r_3 \in \Omega$ *such that*

1. $r_2 = A \rightarrow C \wedge D$, *and*
2. $r_3 = C \rightarrow D$ *where* $confidence(r_3) = 1$.

Verification: To show that this is an impartial interestingness criterion, we need to show that any rule that this criterion eliminates from Ω is a not-interesting rule (in accordance with the Intuitive Problem Statement of Interestingness in Section 7.2), a rule that does not provide any more information than rules that already exist in Ω. This follows directly from the definition of when rules are eliminated according to this criterion: from $r_3 = C \rightarrow D$ such that $confidence(r_3) = 1$ we have that $\mathbf{Pr}[C \wedge D] = \mathbf{Pr}[C]$. Hence, for $r_2 = A \rightarrow C \wedge D$ we have $confidence(A \rightarrow C) = confidence(r_2)$, and we have $support(A \rightarrow C) = support(r_2)$, indicating that the rule $r = A \rightarrow C$ does not contain any more information than the rule r_2. Since according to the transition impartial interestingness criterion we have that $r_2, r_3 \in \Omega$, we have easily verified our claim. ■

Example 7.5.2 *We examine three rules:* $r = \langle \texttt{raisins} \rangle \rightarrow \langle \texttt{cereal} \rangle$, $r_2 = \langle \texttt{raisins} \rangle \rightarrow \langle \texttt{milk} \wedge \texttt{cereal} \rangle$, *and* $r_3 = \langle \texttt{cereal} \rangle \rightarrow \langle \texttt{milk} \rangle$ *such that* $\mathbf{Pr}[\texttt{milk}] = 0.8$, $\mathbf{Pr}[\texttt{cereal}] = 0.6$, $\mathbf{Pr}[raisins] = 0.3$, $\mathbf{Pr}[milk \wedge \texttt{cereal}] = 0.6$, $\mathbf{Pr}[milk \wedge \texttt{raisins}] = 0.2$, $\mathbf{Pr}[cereal \wedge \texttt{raisins}] = 0.1$ *and* $\mathbf{Pr}[milk \wedge cereal \wedge \texttt{raisins}] = 0.1$, *that for the purpose of this illustrative example we assume were all outputted by the same mining process. We then have that* $confidence(r_3) = 1$, *indicating that this preprocessing criterion (Impartial Interestingness Criterion 2: Transition) would delete the rule* r. *We note that* $support(r) = support(r_2)$ *and that* $confidence(r) = confidence(r_2)$. *This impartial criterion does not delete* r_3, *so that if a user is interested in a relationship between milk and cereal, that information is still available. However, since* r *does not provide any more information than that already expressed by* r_2 *and* r_3, *we do not lose information, or an interesting rule, by deleting it.*

Note that given four rules, $r = A \rightarrow C \wedge D$, $r_2 = A \rightarrow C$, $r_3 = C \rightarrow D$ such that $confidence(r_3) = 1$, and, $r_4 = A \rightarrow D$, r_4 cannot be deleted in the general case without risking losing a potentially interesting rule. In Example 7.5.2, r_4 corresponds to the rule $\langle \texttt{milk} \rangle \rightarrow \langle \texttt{raisins} \rangle$. Now, $confidence(r_4) = 0.25$, and users interested in the relationship between customers who purchase milk and customers who purchase raisins may find the higher confidence level of r_4 interesting. Thus, in the general case, r_4 cannot be deleted.

7.6 Empirical Evaluation With Real Data

Aristotle maintained that women have fewer teeth than men; although he was twice married, it never occurred to him to verify this statement by examining his wives' mouths.

—Bertrand Russell

To determine the effectiveness of the impartial interestingness criteria we introduced in Section 7.5, we present the results of their application on the six databases described in Section A.1: the Grocery Database, the WWW Proxy Logs Database, the Nursery Database, the Chess Database, the Adult Database, and the Mushroom Database. We ran the association rule mining algorithm described in Section 2.2 on these databases with the following thresholds:

- 3.5% thresholds for the Grocery Database, yielding 3,046 mined rules,
- 6% thresholds for the WWW Proxy Logs Database, yielding 9,206 rules,
- 3% for the Nursery Database, yielding 8,314 mined rules,
- 10% for the Chess Database, yielding 8,292 mined rules,
- 20% for the Adult Database, yielding 13,906 mined rules, and,
- 35% for the Mushroom Database, yielding 6,356 mined rules.

7.6.1 Results of Application of the Overfitting Impartial Interestingness Criterion

As explained in Section 7.5.1, the overfitting impartial interestingness criterion relies on relative values of the confidence (the predictability value) of association rules to delete not-interesting rules. To demonstrate that this reliance does not translate into a functional dependence of the criterion on a particular choice of confidence mining thresholds, we applied this impartial criterion to the outputs of mining the databases using different confidence thresholds. That is, we mined each database using a single support threshold and several increasing confidence thresholds. The support thresholds we used for each database are listed in the beginning of Section 7.6. The results

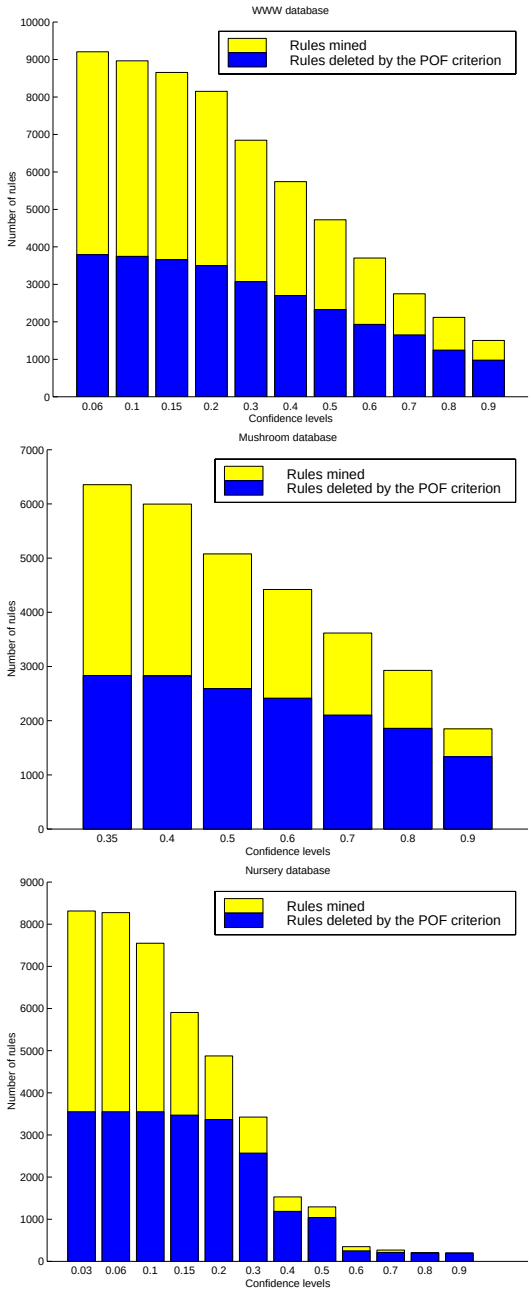

Figure 7.2: **Results of the application of the overfitting impartial interestingness criterion to the WWW Proxy Logs Database, Mushroom Database and the Nursery Database.**

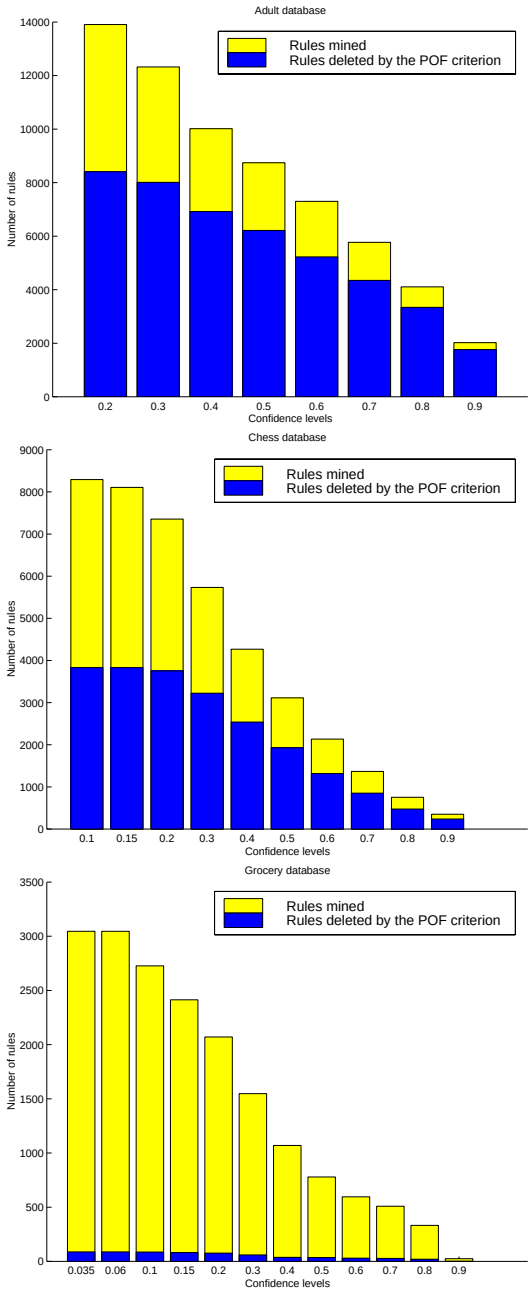

Figure 7.3: **Results of the application of the overfitting impartial interestingness criterion to the Adult Database, Chess Database and the Grocery Database.**

of the application of the first impartial interestingness criterion using several confidence thresholds for each database are summarized in Figures 7.2 and 7.3.

For example, the WWW Proxy Logs Database was mined with confidence thresholds of 0.06, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, and 0.9 using a 0.06 support threshold in all cases. The results of the application of this impartial interestingness criterion on the association rules outputted by these ten mining processes are summarized in the top histogram of Figure 7.2. Each light-colored bar indicates the number of rules mined using a specific confidence threshold. For example, the first (leftmost) light-colored bar in the WWW Proxy Logs Database histogram depicts the 9,206 association rules that were mined with 0.06 confidence and support thresholds. The dark-colored bar superimposed on it indicates the number of rules deleted as result of the application of the overfitting impartial criterion on the 9,206 mined rules, 3,793 rules deleted in this case. That is, over 41% of the rules mined by from the WWW Proxy Logs Database using 0.06 support and confidence thresholds were deleted by the overfitting impartial interestingness criterion. This deletion is very significant considering that no user interaction was required to perform it. This impartial interestingness criterion performed even better on the output of mining the WWW Proxy Logs Database using increasingly higher confidence levels, deleting as much as 65% of the rules mined with 0.9 confidence levels, as indicated by the rightmost bar in the WWW Proxy Logs Database histogram: out of the 1,503 rules mined (the rightmost light-colored bar in the top histogram of Figure 7.2), 977 rules (the superimposed rightmost dark-colored bar) rules were deleted.

The other five histograms in Figures 7.2, and 7.3 similarly summarize the results of applying the overfitting impartial interestingness criterion to the other five databases. The average deletions brought about by the application of this impartial interestingness criterion are summarized in Table 7.2.

These results are extremely encouraging: a significant number, in most cases, an average of over *half* the rules outputted by the mining algorithm are deleted by the application of the overfitting impartial interestingness criterion. This is a *very* significant result not only because of the large number of rules that is deleted by this preprocessing technique, but also because this deletion requires *no* information of any kind from a user of the KDD process.

Remember that the output of this impartial criterion, rules that were *not* filtered out by it, is not necessarily comprised exclusively of interesting rules.

Database	Impartial criterion overfitting average deletion
WWW Proxy Logs Database	$49.3\% \pm 7.9\%$
Chess Database	$57.7\% \pm 7.3\%$
Nursery Database	$69.4\% \pm 18.0\%$
Adult Database	$72.6\% \pm 8.6\%$
Grocery Database	$3.7\% \pm 1.5\%$
Mushroom Database	$55.2\% \pm 9.8\%$

Table 7.2: **Average percent of rules deleted by the application of the the overfitting impartial interestingness criterion** *(impartial interestingness criterion 1)*

The rules that were filtered out by the this impartial interestingness criterion are overfitting of the data that are not useful or interesting. The output of this preprocessing needs to be further evaluated and filtered by other criteria to extract only the subjectively interesting rules from it.

7.6.2 Results of Application of the Transition Impartial Interestingness Criterion

The results of the application of the transition impartial interestingness criterion (Impartial Interestingness Criterion 2, Section 7.5.2) to the six databases were more modest than those of the application of the overfitting impartial criterion. Even these modest results are very significant: a reduction even of a small percentage of the rules is both effective and important when these results are achieved automatically, without any interaction with any users for any purpose. The average elimination brought about by the transition impartial criterion was as follows: $3.3\% \pm 3.4\%$ for the WWW Proxy Logs Database, $17.4\% \pm 5\%$ for the Mushroom Database, $2.5\% \pm 2.7\%$ for the Nursery Database, $3.1\% \pm 1.8\%$ for the Chess Database, $1\% \pm 1.9\%$ for the Adult Database. We present the detailed results in a graphical representation in Figure 7.4 for the first three databases.

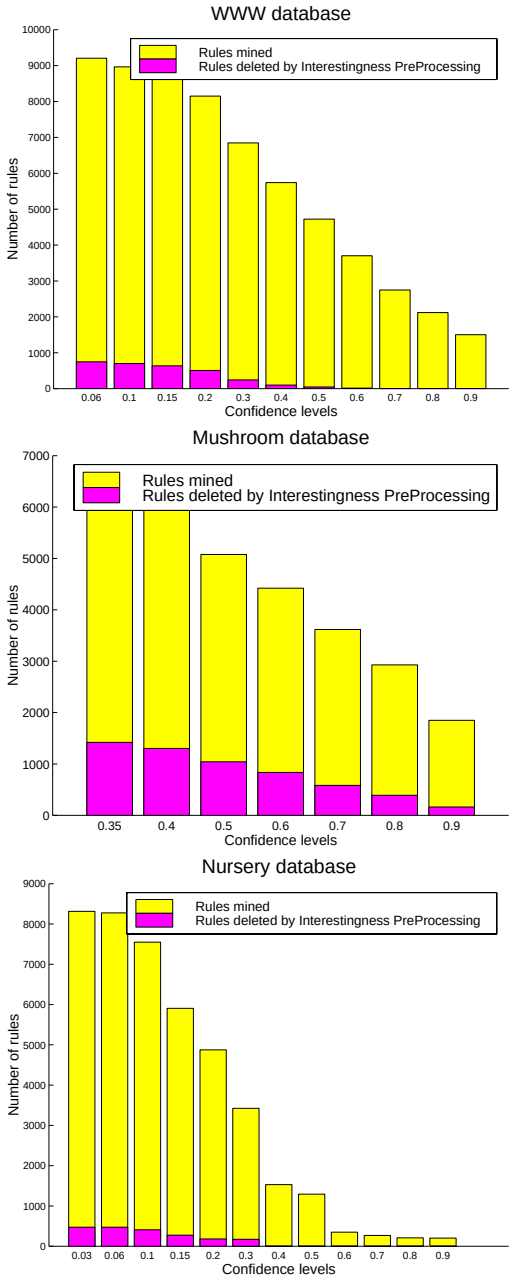

Figure 7.4: **Results of the application of the transition impartial interestingness criterion**

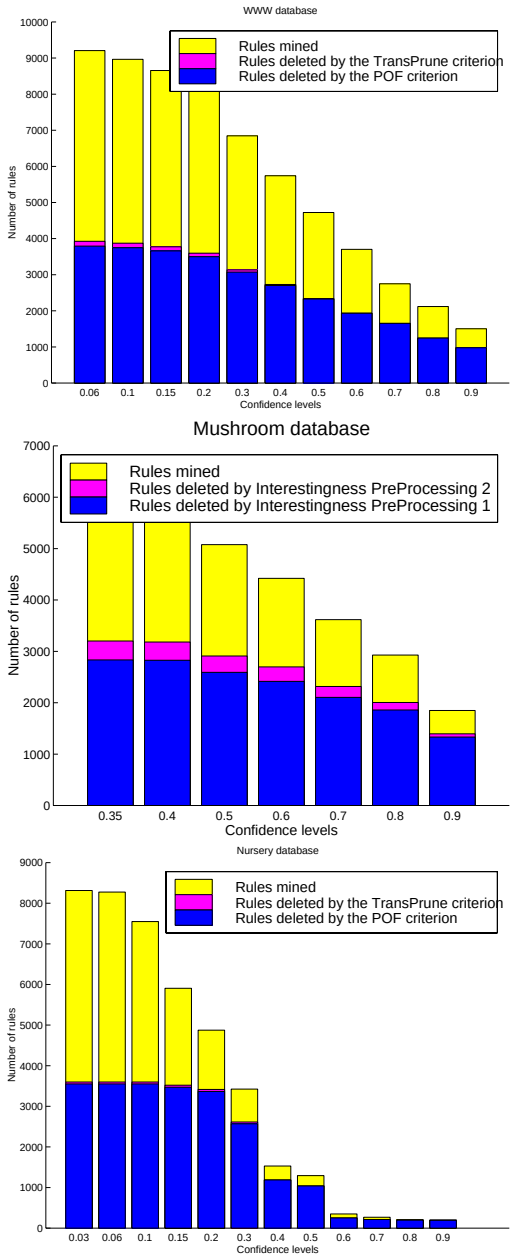

Figure 7.5: **Results of the sequential application of the impartial interestingness criteria to the WWW Proxy Logs Database, the Mushroom Database and the Nursery Database**

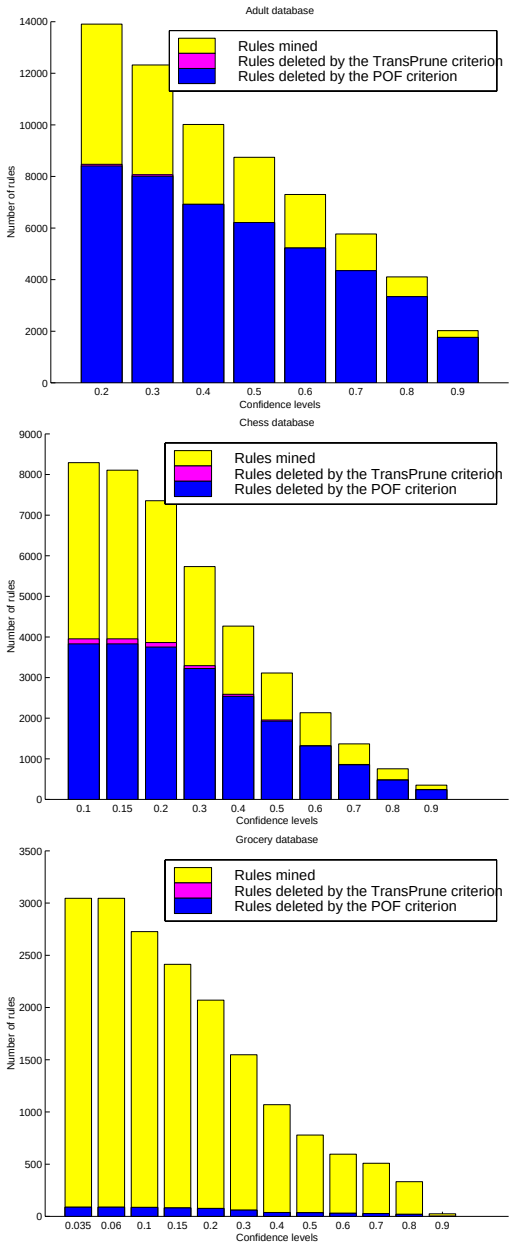

Figure 7.6: **Results of the sequential application of the impartial interestingness criteria to the Adult Database, the Chess Database and the Grocery Database.**

7.6.3 Results of the Sequential Application of the Impartial Interestingness Criteria

As depicted in Figure 7.1, the impartial interestingness criteria can and *should* be applied sequentially. We proceeded to do so, applying the two impartial interestingness criteria we introduced in this chapter sequentially. The results of this sequential application are depicted in Figure 7.5 for the WWW Proxy Logs Database, Mushroom Database and Nursery Database and in Figure 7.6 for the Adult Database, Chess Database and Grocery Database.

7.7 Summary

In this chapter we introduced a new type of interestingness criteria, the impartial interestingness criteria. The impartial criteria eliminate association rules that can be determined to be not interesting in a user-, domain-, and task-independent manner. This elimination process does not require any information from the KDD users, such as rule-preference, domain knowledge, etc. The generic nature of the impartial criteria differentiates it from the existing interestingness criteria, both subjective and objective (that fall under the Interestingness Processing Step), that do impose a bias on the type of rules outputted as interesting. The impartial criteria form the Interestingness PreProcessing Step, the first step in the interestingness analysis framework we introduced. The Interestingness PreProcessing Step is a new step in the interestingness analysis that is always applied directly after the mining, in any domain, and without user interaction.

In this work we also introduced the first two impartial interestingness criteria. The deletion of not-interesting rules by these criteria is very significant since it is achieved without requiring a user to provide *any* information. Our empirical results, of the application of these impartial criteria, indicate that these are powerful criteria: in most cases, more than half the rules were eliminated by their application. It is important to note that any filtering power, however weak, of any impartial criterion is significant since *all* the impartial criteria are activated sequentially immediately following the mining process and these results add up.

Chapter 8

Concluding Remarks and Future Work

Now this is not the end. It is not even the beginning of the end. But it is, perhaps, the end of the beginning.

—Winston Churchill

In this dissertation we studied one of the central problems in KDD, that of interestingness. The problem of interestingness is becoming more and more important in KDD as the size of the mined databases increases, producing an increasingly larger number of patterns.

Data mining algorithms output patterns. The human users, or the methods to do so imbued in the algorithms by a human counterpart, determine whether or not these patterns are interesting: understandable, meaningful, actionable, new, etc. We studied the two types of interestingness available in the literature, objective interestingness and subjective interestingness, and introduced a third type: impartial interestingness criteria. We investigated objective interestingness ranking criteria in Chapter 3 (based on [SM99]) and found that they are useful tools, but not sufficient to resolve the problem of interestingness. Interestingness is ultimately subjective, requiring the application of subjective interestingness criteria. In Chapter 4 (based on [Sah99]) we introduced a new approach to subjective interestingness, one that requires very little domain knowledge, that even naive users can provide, in order to quickly eliminate the majority of rules that are not interesting. We investigated how this approach can be incorporated into the mining process and the benefits and disadvantages of doing so in Chapter 5 (based on [Sah02b]). In Chapter 6 (based on [Sah02a]) we introduced a framework for association

rule clustering. This framework enables the automation of much of the tedious, menial work normally involved in the exploration and understanding of interestingness. Finally, in Chapter 7 (based on [Sah01]) we introduced a completely new class of interestingness criteria, criteria that are impartial to the domain, user, and task of the KDD process and can therefore be automatically applied immediately following any association rule mining process to eliminate not-interesting rules. These are the impartial interestingness criteria. They form the Interestingness PreProcessing Step which is the first step in the interestingness framework we outline.

In each of the areas we discussed there are several potential directions for future research. In the Interestingness Via What Is Not Interesting approach we restricted the algorithm to collect a very limited amount of domain knowledge. Extending the domain knowledge gathered to allow the elimination of more not-interesting rules while maintaining the user-interaction to be low-intensity (low-volume and low-level) is therefore one direction of future research. Another area of future research is investigating ways to limit the amount of questions users are asked about what they are not interested in (which they may find tedious). In the area of incorporating subjective interestingness into the mining process future research includes the examination and formalization of the types of domain knowledge that can be used to decrease the size of the search space—which can be critical when mining dense databases—without restricting the output of the mining process to meet the needs of only a narrow audience base. An important direction of future research is combining the post-processing of the Interestingness Via What Is Not interesting approach with the clustering framework introduced in this work, as well as with the other interestingness approaches. There are several other areas open for future research around the clustering framework we introduced: continued analysis of interestingness of outlier rules in different environments and its possible use in noisy data, the investigation of the benefits of incorporating fuzzy clustering methodologies to enable a single rule to have varying degrees of memberships to different clusters, and clustering of association rules that supports incremental databases. In the objective interestingness criteria space future research includes incorporating objective interestingness into the mining process as well as theoretical analysis of their behavior. Future research in Interestingness PreProcessing branches in two direction. First, it includes the development of other interestingness PreProcessing methodologies to enable the automatic elimination of more not-interesting rules and approach the subset of rules that

only includes interesting rules. A second venue of future research is how to incorporate the PreProcessing into the mining process instead of applying these criteria as post-processing methods.

[Pyl98] says that *"In over 20 years in the field now known as data mining, never once have I used a magic wand. Nonetheless, business managers seem to expect magic from applying data mining tools to their data. I've even heard someone say, 'Data mining is like worms crawling through your data, discovering nuggets of gold!'"* KDD and interestingness are not magic wands. They are tools to discover potentially interesting patterns in data. As we said in the introduction in Chapter 1, these patterns need to be acted upon to be useful (but that is outside the scope of the KDD process and this dissertation). However, KDD and interestingness are essential tools in the extremely data-rich environment in which we live. It is likely that our environment will continue to inundate us with data, making these tools critical for our success.

Prediction is very difficult, especially about the future.

—Niels Bohr

Bibliography

Employ your time in improving yourself by other men's writings, so that you shall gain easily what others have labored hard for.

—Socrates

[AHKV98] Helena Ahonen, Oskari Heinonen, Mika Klemettinen, and A. Inkeri Verkamo. Applying data mining techniques for descriptive phrase extraction in digital document collections. In *Proceedings of the IEEE Forum on Research and Technology Advances in Digital Libraries (IEEE ADL)*, pages 2–11, Santa Barbara, CA, USA, April 1998.

[AHS+96] Rakesh Agrawal, Mannila Heikki, Ramakrishnan Srikant, Hannu Toivonen, and A. Inkeri Verkamo. *Advances in Knowledge Discovery and Data Mining*, chapter 12: *Fast Discovery of Association Rules*, pages 307–328. AAAI Press/The MIT Press, Menlo Park, California, 1996.

[AIS93] Rakesh Agrawal, Tomasz Imielinski, and Arun Swami. Mining association rules between sets of items in large databases. In Peter Buneman and Sushil Jajodia, editors, *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, pages 207–216, Washington DC, USA, May 1993. ACM Press.

[AT97] Gediminas Adomavicius and Alexander Tuzhilin. Discovery of actionable patterns in databases: The action hierarchy approach. In David Heckerman, Heikki Mannila, Daryl Pregibon, and Ramasamy Uthurusamy, editors, *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining*, pages 111–114, Newport Beach, CA, USA, August 1997. AAAI Press.

[AT99] 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*, pages 377–381, San Diego, CA, USA, August 1999.

[AT01] 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, January–April 2001.

[AY98a] Charu C. Aggarwal and Philip S. Yu. A new approach to online generation of association rules. Technical Report Research Report RC 20899, IBM T J Watson Research Center, 1998.

[AY98b] Charu C. Aggarwal and Philip S. Yu. Online generation of association rules. In *Proceedings of the Fourteenth IEEE International Conference on Data Engineering (ICDE)*, pages 402–411, Orlando, FL, USA, February 1998.

[BFOS84] Leo Breiman, Jerome H. Friedman, Richard A. Olshen, and Charles J. Stone. *Classification and Regression Trees*. 1984.

[BH03] Brock Barber and Howard J. Hamilton. Extracting share frequent itemsets with infrequent subsets. *Data Mining and Knowledge Discovery*, 7(2):153–185, April 2003.

[BJA99] Roberto J. Bayardo Jr. and Rakesh Agrawal. Mining the most interesting rules. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 145–154, San Diego, CA, August 1999.

[BJAG99] Robert J. Bayardo Jr., Rakesh Agrawal, and Dimitrios Gunopulos. Constraint-based rule mining in large, dense databases. In *Proceedings of the Fifteenth IEEE ICDE International Conference on Data Engineering*, pages 188–197, Sydney, Australia, March 1999.

[BKM] C. Blake, E. Keogh, and C.J. Merz. UCI repository of machine learning databases `http://www.ics.uci.edu/~mlearn/MLRepository.html`.

[Bla96] Tony Blair. Computers and children: Partnerships for the future. *The Computer Bulletin: Special Supplement: IT in Schools*, April 1996.

[BMPG01] Sugato Basu, Raymond J. Mooney, Krupakar V. Pasupuleti, and Joydeep Ghosh. Evaluating the novelty of text-mined rules using lexical knowledge. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 233–238, San Francisco, CA, USA, August 2001.

[BMS97] Sergey Brin, Rajeev Motwani, and Craig Silverstein. Beyond market baskets: Generalizing association rules to correlations. In Joan Peckham, editor, *Proceedings of ACM SIGMOD International Conference on Management of Data*, pages 265–276, Tucson, AZ, USA, May 1997.

[BMUT97] Sergey Brin, Rajeev Motwani, Jeffrey D. Ullman, and Shalom Tsur. Dynamic itemset counting and implication rules for market basket data. In Joan Peckham, editor, *Proceedings ACM SIGMOD International Conference on Management of Data*, pages 255–264, Tucson, AZ, USA, May 1997.

[Boo83] Daniel J. Boorstin. *The Discoverers*. Random House, 1983.

[CDF+01] Edith Cohen, Mayur Datar, Shinji Fujiwara, Aristides Gionis, Piotr Indyk, Rajeev Motwani, Jeffrey D. Ullman, and Cheng Yang. Finding interesting associations without support pruning. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 13(1):64–78, 2001.

[Chu58] Winston S. Churchill. *A History of the English Speaking People*, volume I: The Birth of Britain. Dodd, Mead & Company, New York, 1958.

[Cli98] William J. Clinton. Remarks by the president to the 150th anniversary of the American Association for the Advancement of Science. The White House Office of the Press Secretary, `http://www.aaas.org/meetings/1998/clinton98.htm`, February 1998.

[CT91] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. John Wiley & Sons, Inc., 1991.

[CTS00] Robert Cooley, Pang-Ning Tan, and Jaideep Srivastava. Discovery of interesting usage patterns from web data. *Web Usage Analysis and User Profiling (Proceedings of WEBKDD Workshop)*, 1836 of Lecture Notes in Computer Science:163–182, 2000.

[DB98] Steven K. Donoho and Scott W. Bennett. Fraud detection and discovery. In *Tutorial T4: Fourth International Conference on Knowledge Discovery and Data Mining*, New York, NY, USA, August 1998.

[DH73] Richard O. Duda and Peter E. Hart. *Pattern Classification and Scene Analysis*. John Wiley & Sons, 1973.

[Dic92] *The American Heritage Dictionary of the English Language*. Houghton Mifflin Co., third edition, 1992.

[DL98] Guozhu Dong and Jinyan Li. Interestingness of discovered association rules in terms of neighborhood based unexpectedness. In Xindong Wu, Ramamohanarao Kotagiri, and Kevin B. Korb, editors, *Proceedings of the Second Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*, volume 1394 of *Lecture Notes in Computer Sciences*, pages 72–86, Melbourne, Australia, April 1998. Springer.

[dSP75] Derek de Solla Price. *Science Since Babylon*. Yale University Press, New Haven and London, enlarged edition, 1975.

[Fik65] Gregory Mikhailovich Fikhtengol'ts. *The Fundamentals of Mathematical Analysis*, volume 1. Pergamon Press, 1965.

[FPSS96a] Usama M. Fayyad, Gregory Piatetsky-Shapiro, and Padhraic Smyth. *Advances in Knowledge Discovery and Data Mining*, chapter 1: *From Data Mining to Knowledge Discovery: An Overview*, pages 1–34. AAAI Press, 1996.

[FPSS96b] Usama M. Fayyad, Gregory Piatetsky-Shapiro, and Padhraic Smyth. The kdd process for extracting useful knowledge from volumes of data. *Communications of the ACM*, 39(11):27–34, 1996.

[FPSS96c] Usama M. Fayyad, Gregory Piatetsky-Shapiro, and Padhraic Smyth. Knowledge discovery and data mining: Towards a unifying framework. In Evangelos Simoudis, Jiawei Han, and Usama M. Fayyad, editors, *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, pages 82–88, Portland, OR, USA, August 1996.

[FPSSU96] Usama M. Fayyad, Gregory Piatetsky-Shapiro, Padhraic Smyth, and Ramasamy Uthurusamy. *Advances in Knowledge Discovery and Data Mining*, chapter Preface, pages xiii–xiv. AAAI Press, 1996.

[Fre98] Alex A. Freitas. A multi-criteria approach for the evaluation of rule interestingness. In *Proceedings of the International Conference on Data Mining*, pages 7–20, Rio de Janeiro, Brazil, September 1998.

[GB98] Pedro Gago and Carlos Bento. A metric for selection of the most promising rules. In Jan M. Zytkow and Mohamed Quafaou, editors, *Proceedings of the Second European Symposium on Principles of Data Mining and Knowledge Discovery (PKDD)*, volume 1510 of *Lecture Notes in Computer Science*, pages 19–27, Nantes, France, September 1998. Springer.

[GPSP00] Steve Gallant, Gregory Piatetsky-Shapiro, and Dorian Pyle. Data mining for successful customer relationship management (crm). In *Tutorial PM-1: Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Boston, MA, USA, August 2000.

[Han95] Jiawei Han. Mining knowledge at multiple concept levels. In *Proceedings of the ACM CIKM International Conference on Information and Knowledge Management, Invited Talk*, pages 19–24, Baltimore, MD, USA, 1995.

[Han97] Rudolf Hanka. Information overload and the need for 'just-in-time' knowledge. In *Proceedings of the Fourth Hong Kong (Asia-Pacific) Medical Informatics Conference*, Hong Kong, 1997.

[Har75] John A. Hartigan. *Clustering Algorithms*. John Wiley & Sons, 99th edition, 1975.

[HG02] Jochen Hipp and Ulrich Günter. Is pushing constraints deeply into the mining algorithms really what we want? *SIGKDD Explorations*, 4(1):50–55, June 2002.

[HGN00] Jochen Hipp, Ulrich Güntzer, and Gholamreza Nakhaeizadeh. Algorithms for association rule mining – a general survey and comparison. *SIGKDD Explorations*, 2(1):58–64, July 2000.

[HH00] Robert J. Hilderman and Howard J. Hamilton. Principles for mining summaries using objective measures of interestingness. In *Proceedings of the Twelfth IEEE International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 72–81, Vancouver, Canada, November 2000.

[HH01a] R. J. Hilderman and H. J. Hamilton. *Knowledge Discovery and Measures of Interest*. Kluwer Academic Publishers, 2001.

[HH01b] Robert J. Hilderman and Howard J. Hamilton. Evaluation of interestingness measures for ranking discovered knowledge. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD-01)*, pages 247–259, 2001.

[Hid99] Christian Hidber. Online association rule mining. In Alex Delis, Christos Faloutsos, and Shahram Ghandeharizadeh, editors, *Proceedings ACM SIGMOD International Conference on Management of Data*, pages 145–156, Philadelphia, PA, USA, June 1999. ACM Press.

[HK01a] Jiawei Han and Micheline Kamber. *Data Mining Concepts and Techniques*, chapter 3: *Data Preprocessing*, pages 105–143. Morgan Kaufmann Publishers, Academic Press, 2001.

[HK01b] Jiawei Han and Micheline Kamber. *Data Mining Concepts and Techniques*, chapter 8: *Cluster Analysis*, pages 335–393. Morgan Kaufmann Publishers, Academic Press, 2001.

[HKK00] Eui-Hong (Sam) Han, George Karypis, and Vipin Kumar. Scalable parallel data mining for association rules. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 12(3):337–352, May/June 2000.

[JMF99] Anil. K. Jain, M. Narasimha Murty, and Patrick. J. Flynn. Data clustering: a review. *ACM Computing Surveys*, 31(3):264–323, 1999.

[KF70] Andrei Nikolaevich Kolmogorov and Sergei V. Fomin. *Introductory Real Analysis*. Dover Publication, 1970.

[KHK99] George Karypis, Eui-Hong (Sam) Han, and Vipin Kumar. Chameleon: A hierarchical clustering algorithm using dynamic modeling. *IEEE Computer: Special Issue on Data Analysis and Mining*, 32(8):68–75, 1999.

[Kle99] Mika Klemettinen. *A Knowledge Discovery Methodology for Telecommunication Network Alarm Databases*. PhD thesis, University Of Helsinki, 1999.

[Klo92] Willi Klosgen. Problems for knowledge discovery in databases and their treatment in the statistics interpreter explora. *International Journal of Intelligent Systems*, 7(7):649–673, 1992.

[Klo96] Willi Klosgen. *Advances in Knowledge Discovery and Data Mining*, chapter 10: *Explora: a Multipattern and Multistrategy Discovery Assistant*, pages 249–271. AAAI Press, 1996.

[KMR+94] Mika Klemettinen, Heikki Mannila, Pirjo Ronkainen, Hannu Toivonen, and A. Inkeri Verkam. Finding interesting rules from large sets of discovered association rules. In *Proceedings of the Third ACM CIKM International Conference on Information and Knowledge Management*, pages 401–407, Orlando, FL, USA, November 29–December 2 1994. ACM Press.

[KMS97] Ali Kamal, Stefanos Manganaris, and Ramakrishnan Srikant. Partial classification using association rules. In David Heckerman, Heikki Mannila, Daryl Pregibon, and Ramasamy Uthurusamy, editors, *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining*, pages 115–118. AAAI Press, 1997.

[Koh01] Ronny Kohavi. Mining e-commerce data: the good, the bad and the ugly. In *Industry Track, Invited Talk*, pages 8–13, San Francisco, CA, USA, August 2001.

[KS96] Micheline Kamber and Rajjan Shinghal. Evaluating the interestingness of characteristic rules. In Evangelos Simoudis, Jiawei Han, and Usama M. Fayyad, editors, *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, pages 263–266, Portland, OR, USA, August 1996.

[Lan83] Serge Lang. *Real Analysis*. Addison-Wesley Publishing Company, 2 edition, 1983.

[Les97] Michael Lesk. How much information is there in the world? `http://www.lesk.com/mlesk/ksg97/ksg.html`. Technical report, `lesk.com`, 1997.

[LH96] Bing Liu and Wynne Hsu. Post-analysis of learned rules. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI)*, pages 828–834, Portland, OR, USA, August 1996.

[LHC97] Bing Liu, Wynne Hsu, and Shu Chen. Using general impressions to analyze discovered classification rules. In David Heckerman, Heikki Mannila, Daryl Pregibon, and Ramasamy Uthurusamy, editors, *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining*, pages 31–36, Newport Beach, CA, USA, August 1997. AAAI Press.

[LHH00] Bing Liu, Minqing Hu, and Wynne Hsu. Multi-level organization and summarization of the discovered rules. In *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 208–217, Boston, MA, USA, August 2000.

[LHM99a] Bing Liu, Wynne Hsu, and Yiming Ma. Mining association rules with multiple minimum supports. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 337–341, San Diego, CA, USA, August 1999.

[LHM99b] Bing Liu, Wynne Hsu, and Yiming Ma. Pruning and summarizing the discovered associations. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 125–134, San Diego, CA, USA, August 1999.

[LHM01a] Bing Liu, Wynne Hsu, and Yiming Ma. Discovery the set of fundamental rule changes. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 335–340, San Francisco, CA, USA, August 2001.

[LHM01b] Bing Liu, Wynne Hsu, and Yiming Ma. Identifying non-actionable association rules. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 329–334, San Francisco, CA, USA, August 2001.

[LMY01] Bing Liu, Yiming Ma, and Philip S. Yu. Discovering unexpected information from your competitors' web sites. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 144–153, San Francisco, CA, USA, August 2001.

[LSM98] Wenke Lee, Salvatore Stolfo, and Kui W. Mok. Data mining approaches for intrusion detection. In *Proceedings of the Seventh (USENIX) Security Symposium*, San Antonio, TX, USA, 1998.

[LSW97] Brian Lent, Arun N. Swami, and Jennifer Widom. Clustering association rules. In Alex Gray and Per-Åke Larson, editors, *Proceedings of the Thirteenth IEEE ICDE International Conference on Data Engineering*, pages 220–231, Birmingham, UK, April 1997.

[MDLN01] Bamshad Mobasher, Honghua Dai, Tao Luo, and Miki Nakagawa. Effective personalization based on association rule discovery from web usage data. In *Proceedings of the Third International Workshop on Web Information and Data Management (WIDM), in Conjunction with ACM CIKM*, pages 9–15, Atlanta, GA, USA, November 2001.

[MLC] `SGI MLC++`: Machine learning library in c++ `http://www.sgi.com/Technology/mlc`.

[MM95] John A. Major and John J. Mangano. Selecting among rules induced from a hurricane databases. *Journal of Intelligent Information Systems*, 4:39–52, 1995.

[MPSM96] Christopher J. Matheus, Gregory Piatetsky-Shapiro, and Dwight McNeill. *Advances in Knowledge Discovery and Data Mining*, chapter 20: *Selecting and Reporting What Is Interesting: The kefir application to healthcare data*, pages 495–515. AAAI Press, 1996.

[MTV94] Heikki Mannila, Hannu Toivonen, and A. Inkeri Verkamo. Efficient algorithms for discovering association rules. In *Knowledge Discovery in Databases (KDD'1994): Workshop on Knowledge Discovery in Databases*, pages 181–192, Seattle, Washington, July 1994. AAAI Press.

[MY97] Renée J. Miller and Yuping Yang. Association rules over interval data. In Joan Peckham, editor, *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 452–461, Tucson, AZ, USA, May 1997.

[NLHP98] Raymond T. Ng, Laks V. S. Lakshmanan, Jiawei Han, and Alex Pang. Exploratory mining and pruning optimizations of constrained association rules. In Laura M. Haas and Ashutosh Tiwary, editors, *Proceedings of ACM SIGMOD International Conference on Management of Data*, pages 13–24, 1998.

[NSKF00] Joel Larocca Neto, Alexandre D. Santos, Celso A. A. Kaestner, and Alex A. Freitas. The integrated data mining tool minekit and a case study of its application on video shop data. In *Proceedings of the Second International ICSC Symposium on Engineering of Intelligent Systems (EIS-2000)*, July 2000.

[PCY95] Jong Soo Park, Ming-Syan Chen, and Philip S. Yu. An effective hash-based algorithm for mining association rules. In *Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data*, pages 175–186, San Jose, CA, USA, May 1995.

[PH00] J. Pei and J. Han. Can we push more constraints into frequent pattern mining? In *Proc. of the International Conference on Knowledge Discovery and Data Mining*, pages 350–254, 2000.

[PH01] Vikram Pudi and Jayant R. Haritsa. On the optimality of association-rule mining algorithms. Technical Report TR-2001-01.

Appeared as a poster in ICDE-2001, Database Systems Lab, SERC, Indian Institute of Science, 2001.

[PH02] Vikram Pudi and Jayant R. Haritsa. On the efficiency of association-rule mining algorithms. In Ming-Shan Cheng, Philip S. Yu, and Bing Liu, editors, *Advances in Knowledge Discovery and Data Mining, 6th Pacific-Asia Conference, PAKDD*, volume 2336 of *Lecture Notes in Computer Science*, pages 80–91, Taipei, Taiwan, May 2002. Springer.

[PJ98] Foster Provost and David Jensen. Evaluating knowledge discovery and data mining. In *Tutorial T7: Fourth International Conference on Knowledge Discovery and Data Mining*, New York, NY, USA, August 1998.

[Pou99] Chris Pound. In cyberspace no one can hear you scream. In *Industrial Session 1: Data Management in a Networked World, part of 25th International Conference on Very Large DataBases*, Edinburgh, Scotland, UK, September 1999.

[PS91] Gregory Piatetsky-Shapiro. *Knowledge Discovery in Databases*, chapter 13: *Discovery, Analysis, and Presentation of Strong Rules*, pages 292–248. AAAI/MIT Press, 1991.

[PSM99] Gregory Piatetsky-Shapiro and Brij Masand. Estimating campaign benefits and modeling lift. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 185–193, San Diego, CA, USA, August 1999.

[PT98] Balaji Padmanabhan and Alexander Tuzhilin. A belief-driven method for discovering unexpected patterns. In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*, pages 94–100, New York City, NY, USA, 1998.

[PT00] Balaji Padmanabhan and Alexander Tuzhilin. Small is beautiful: Discovering the minimal set of unexpected patterns. In *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 54–63, Boston, MA, USA, August 2000.

[Pyl98] Dorian Pyle. Knowledge discovery and data mining: The expectation of magic. *PC AI Magazine*, 12(5):14–16, September/October 1998.

[Pyl99] Dorian Pyle. *Data Preparation for Data Mining*. Morgan Kaufmann, 1999.

[RP01] John F. Roddick and Sally Price. What's interesting about cricket? — on thresholds and anticipation in discovered rules. *SIGKDD Explorations*, 3(1):1–5, July 2001.

[Sah99] Sigal Sahar. Interestingness via what is not interesting. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 332–336, San Diego, CA, USA, August 1999.

[Sah01] Sigal Sahar. Interestingness preprocessing. In *Proceedings of the IEEE ICDM International Conference on Data Mining*, pages 489–496, San Jose, CA, USA, November 29–December 2 2001.

[Sah02a] Sigal Sahar. Exploring interestingness through clustering: A framework. In *Proceedings of the IEEE ICDM International Conference on Data Mining*, pages 677–680, Maebashi City, Japan, December 2002.

[Sah02b] Sigal Sahar. On incorporating subjective interestingness into the mining process. In *Proceedings of the IEEE ICDM International Conference on Data Mining*, pages 681–684, Maebashi City, Japan, December 2002.

[SK98] Einoshin Suzuki and Yves Kodratoff. Discovery of surprising exception rules based on intensity of implication. In Jan M. Zytkow and Mohamed Quafaou, editors, *Proceedings of the Second European Symposium on Principles of Data Mining and Knowledge Discovery (PKDD)*, volume 1510 of *Lecture Notes in Computer Science*, pages 10–18, Nantes, France, September 1998. Springer.

[SLRS99] Devavrat Shah, Laks V. S. Lakshmanan, Krithi Ramamritham, and S. Sudarshan. Interestingness and pruning of mined patterns. In *Proceedings of the ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery (DMKD)*, Philadelphia, PA, USA, May 1999.

[SM99] Sigal Sahar and Yishay Mansour. An empirical evaluation of objective interestingness criteria. In Belur V. Dasarathy, editor, *SPIE*

Conference on Data Mining and Knowledge Discovery, pages 63–74, Orlando, FL, USA, April 1999.

[SON95] Ashok Savasere, Edward Omiecinski, and Shamkant B. Navathe. An efficient algorithm for mining association rules in large databases. In *Proceedings of the 21st International Conference on Very Large Databases (VLDB)*, pages 432–444, Zurich, Switzerland, 1995.

[SR00] Myra Spiliopoulou and John F. Roddick. Higher order mining: Modeling and mining the results of knowledge discovery. In *Proceedings of the Second Conference on Data Mining Methods and Databases*, pages 309–320, Cambridge, UK, 2000. WIT Press.

[ST95] Avi Silberschatz and Alexander Tuzhilin. On subjective measures of interestingness in knowledge discovery. In Usama M. Fayyad and Ramasamy Uthurusamy, editors, *Proceedings of the First International Conference on Knowledge Discovery and Data Mining*, pages 275–281, Montréal, Québec, Canada, August 1995.

[ST96] Avi Silberschatz and Alexander Tuzhilin. What makes patterns interesting in knowledge discovery systems. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 8(6):970–974, 1996.

[Ste66] Burton Stevenson. *The MacMillan Book of Proverbs, Maxims and Famous Phrases*, page 1887. The MacMillan Company, New York, sixth edition, 1966.

[Sub98] Ramesh Subramonian. Defining diff as a data mining primitive. In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*, pages 334–338, New York City, NY, USA, August 1998. AAAI Press.

[SVA97] Ramakrishnan Srikant, Quoc Vu, and Rakesh Agrawal. Mining association rules with item constraints. In David Heckerman, Heikki Mannila, Daryl Pregibon, and Ramasamy Uthurusamy, editors, *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining*, pages 67–73, Newport Beach, CA, USA, August 1997. AAAI Press.

[TA02] Alexander Tuzhilin and Gediminas Adomavicius. Handling very large numbers of association rules in the analysis of microarray data.

In *Proceedings of the Eight ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 396–404, Edmonton, Alberta, Canada, July 2002.

[TK00] Pang-Ning Tan and Vipin Kumar. Interestingness measures for association patterns: A perspective. In *Proceedings of the Workshop on Post-Processing in Machine Learning and Data Mining: Interpretation, visualization, Integration and Related Topics, part of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 56–64, Boston, MA, USA, August 2000.

[TKR+95] Hannu Toivonen, Mika Klemettinen, Pirjo Ronkainen, Kimmo Hätönen, and Heikki Mannila. Pruning and grouping discovered association rules. In *Proceedings of the MLnet Familiarization Workshop on Statistics, Machine Learning and Knowledge Discovery in Databases*, pages 47–52, Heraklion, Crete, Greece, April 1995.

[TKS02] Pang-Ning Tan, Vipin Kumar, and Jaideep Srivastava. Selecting the right interestingness measure for association patterns. In *Proceedings of the Eight ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 32–41, Edmonton, Alberta, Canada, July 2002.

[Toi96] Hannu Toivonen. Sampling large databases for association rules. In *Proceedings of the 22nd International Conference on Very Large Databases (VLDB)*, pages 134–145, Mumbay, India, September 1996. Morgan Kaufmann.

[TS96] Alexander Tuzhilin and Avi Silberschatz. A belief-driven discovery framework based on data monitoring and triggering. Technical Report IS-96-26, Center for Research on Information Systems, Department of Information Systems, New York University Leonard N. Stern School of Business, December 1996.

[Web00] Geoffrey I. Webb. Efficient search for association rules. In *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 99–107, Boston, MA, USA, August 2000.

[WS92] Larry Wall and Randal L. Schwartz. *Programming perl*. O'Reilly & Associates, Inc. MacMillan Company, 1992.

[Zak00] Mohammed Javeed Zaki. Generating non-redundant association rules. In *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 34–43, Boston, MA, USA, August 2000.

Appendix A

Data Description

> *When Kepler found his long-cherished belief did not agree with the most precise observation, he accepted the uncomfortable fact. He preferred the hard truth to his dearest illusions, that is the heart of science.*
>
> —Carl Sagan (in Cosmos)

Throughout the discussion in this dissertation we use empirical studies and analyses to support our hypotheses. The various databases used for these studies are briefly described in this appendix.

A.1 Database Description

A.1.1 Grocery Database

We obtained real-life datasets from an Israeli commercial company that sells groceries via telephone, fax, and the Internet. The company has a list of approximately 95,000 items, not all of which are available at all time. The transaction dataset we received contained a full, detailed description of all purchases made. In order to make the dataset more manageable, we chose to work on the 1,728 items that cover 80% of the gross income of the store. We translated each transaction in the original dataset to one containing 1,757 boolean attributes: 1,728 attributes describe the items purchased, another 19 attributes are used to indicate different distribution centers in the country, and 10 attributes to indicate the range of money spent on a single shopping basket. Thus, the database we compiled contains 67,470 transactions, each describing a single shopping basket: its contents, how much money was spent

on it and where in the country the order was made. Note the very large number of attributes compared to the number of transactions in this database. This is a very sparse database with only a 1.4% average density.

A.1.2 WWW Proxy Logs Database

This database was complied from the logs of a World Wide Web (WWW) proxy server. Each transaction in the database specifies the categories of sites (for example, NEWS, PORNOGRAPHIC, etc.) a particular user browsed. The database represents the accesses of the 2,336 heaviest users to 15 site categories.

A.1.3 Four Datasets from the UCI Repository

We use four datasets from the UCI repository [BKM]:

Adult Database

The [BKM] dataset contains 15 attributes, 6 continuous and 9 nominal. One of the nominal attributes is a boolean attribute detailing whether the adult's income exceeds $50,000 a year. We discretized the 15 attributes into 171 boolean attributes. We used the 45,222 entries in the original dataset from the repository that had no missing attributes as the basis for the compiled database Adult Database.

Mushroom Database

The mushroom dataset [BKM] contains 23 nominal attributes. One of these attributes was an indicator of whether or not the mushroom is poisonous. We converted the 23 attributes to 113 boolean variables. We discarded the eleventh nominal attribute that contained missing values. Thus, we were able to use all 8,124 instances included in the original dataset.

Nursery Database

The nursery dataset was derived from a hierarchical decision model originally developed to rank applications for nursery schools. It contains 8 nominal attributes which we turned into 32 boolean ones. There were no missing

values for any of the attributes and we were able to use all 12,960 instances of the dataset.

Chess Database

This dataset, King and Rook (white) versus King and Pawn (where the pawn is one square away from queening on a7), has a total of 3,196 instances and 37 nominal attributes. One of these nominal attributes is an indicator of whether the white can or cannot win. Each instance in the dataset describes a single board position. We refer to this database as the Chess Database or as the KR vs. KP Database (King Rook versus King Pawn database).

A.2 The Implementations

We have used two implementation of the association rule mining algorithm described in Section 2.2: one using `MLC++` [MLC], and one using Perl 5 (though we referenced [WS92]). We ran both those implementations on the databases described in Section A.1, and used the mined association rules for our analyses. We modified the Perl version for the algorithm implementation in Chapter 5. The algorithms in Chapters 4 and 6 as well as the Interestingness PreProcessing techniques in Chapter 7 were all implemented in Perl.

Appendix B

Acronym Dictionary

There is much pleasure to be gained from useless knowledge.

—Bertrand Russell

B, etc.: Storage sizes Table B below summarizes the commonly used storage sizes.

Name	*Abbreviation*	*Size*	*Size in bytes*
Byte	B	8 bits	1
Kilobyte	KB	1,024 B	1,024
Megabyte	MB	1,024 KB	1,048,576 (or a million)
Gigabyte	GB	1,024 MB	1,073,741,824 (or a billion)
Terabyte	TB	1,024 GB	1,099,511,627,776 (or a trillion)
Petabyte	PB	1,024 TB	1,125,899,906,842,624
Exabyte	EB	1,024 PB	1,152,921,504,606,846,976
Zettabyte	ZB	1,024 EB	1,180,591,620,717,411,303,424
Yottabyte	YB	1,024 ZB	1,208,925,819,614,629,174,706,176

Table B.1: Common Storage Sizes

DB Database.

DM Data Mining, part of the KDD process, Section 1.2. Popularly used to refer to the entire KDD process.

ID Identification

KDD Knowledge Discovery in Databases, see Definition 1 in Section 1.2.

TID Transaction identifier, a unique number that identifies a transaction in a database.

Appendix C

Subjective Interestingness: Interestingness Via What Is Not Interesting Example

The farther backward you can look, the farther forward you are likely to see.

—Sir Winston Churchill

In Chapter 4 we introduced the Interestingness Via What Is Not Interesting approach. In this appendix we give an example of how users can expect to interact with this minimalistic approach. In Figure C.1 we have a representation of the first user classification process over the rules mined from Grocery Database with 3.5% thresholds. The first ancestor rule brought for user classification is ⟨`tomatoes`⟩→⟨`cucumbers`⟩. As discussed in Section 4.4.2, the $RSCL$ value of the ancestor rule, along with some more information on the ancestor rule is presented to the user together with the ancestor rule to be classified. Users can choose to classify the rule, stop classifying ancestor rules (exit the algorithm), or request additional information. In this first iteration the $RSCL$ value of the ancestor rule presented for classification is over 10%. Later iterations will present for classification ancestor rules with equal or smaller $RSCL$ values (see Section 4.4.1). At every point in the algorithm users can opt to stop or exit it. The guideline and expectation is that users will stop classifying rules once the $RSCL$ value of the next rule to be classified drops under a threshold beyond which it no longer pays for users to classify rules, rather than for users to continue to classify many rules, which they may find tedious. There is also the option to examine additional information

such as the top n (user defined) attributes in the assumption and consequent that have the highest support, sorted in descending order according to their support, as well as the ancestor rules sorted in descending order according to their $RSCL$ values. Note that other than the attributes `cucumbers` and `tomatoes` other attributes have been masked and are referred to by coded numbers.

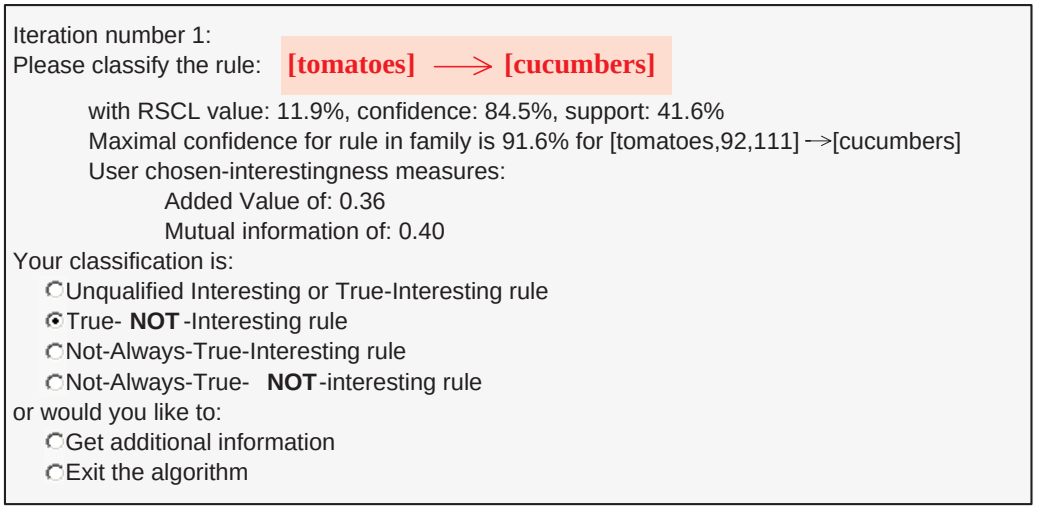

Figure C.1: **Representation of the first user classification in the Interestingness Via What Is Not Interesting algorithm on the Grocery Database.**

The ancestor rule ⟨`tomatoes`⟩→⟨`cucumbers`⟩ was classified as TRUE-NOT-INTERESTING by our users. Our users indicated that they were familiar with the relationship between the purchase of tomatoes and cucumbers; tomatoes and cucumbers are the two main ingredients of salads in Israel. ⟨`Tomatoes`⟩→⟨`cucumbers`⟩, the rule presented for classification is simple in the sense that it only contains a single attribute in its assumption and a single attribute in its consequent: it is an ancestor rule. Making this classification was therefore natural. The TRUE-NOT-INTERESTING classification allows the algorithm to eliminate the entire family of ⟨`tomatoes`⟩→⟨`cucumbers`⟩, as explained in Section 4.4.3. The results of this classification are represented in Figure C.2. These results show that 363 rules were eliminated as a result of the classification of ⟨`tomatoes`⟩→⟨`cucumbers`⟩ as TRUE-NOT-INTERESTING.

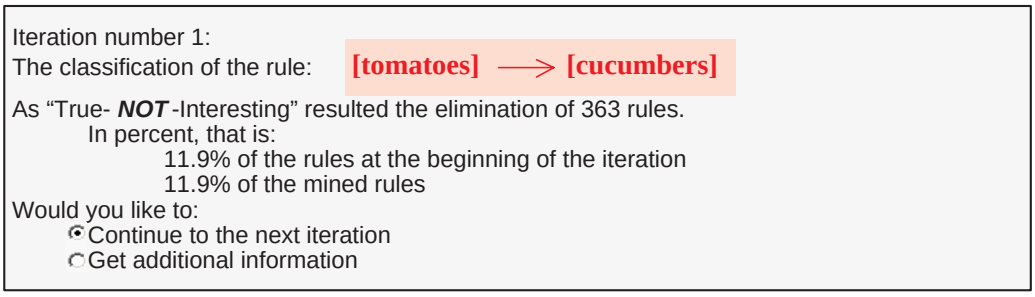

Figure C.2: **Representation of results of first user classification in the Interestingness Via What Is Not Interesting algorithm on the Grocery Database.**

363 make up 11.9% of the 3,046 rules mined with 3.5% thresholds. Since this is the first iteration, the number of rules we started with in this iteration is the total number of mined rules. In following iterations, the number of rules the iteration starts with will be smaller than the number of mined rules, which will make the percent of rules eliminated out of the rules the iteration started out with larger than the percent of rules eliminated out of the total number of rules.