ABSTRACT

ENHANCING WEB MARKETING BY USING AN ONTOLOGY

by
Xuan Zhou

The existence of the Web has a major impact on people's life styles. Online shopping, online banking, email, instant messenger services, search engines and bulletin boards have gradually become parts of our daily life. All kinds of information can be found on the Web. Web marketing is one of the ways to make use of online information. By extracting demographic information and interest information from the Web, marketing knowledge can be augmented by applying data mining algorithms. Therefore, this knowledge which connects customers to products can be used for marketing purposes and for targeting existing and potential customers. The Web Marketing Project with Ontology Support has the purpose to find and improve marketing knowledge.

In the Web Marketing Project, association rules about marketing knowledge have been derived by applying data mining algorithms to existing Web users' data. An ontology was used as a knowledge backbone to enhance data mining for marketing. The Raising Method was developed by taking advantage of the ontology. Data are preprocessed by Raising before being fed into data mining algorithms. Raising improves the quality of the set of mined association rules by increasing the average support value. Also, new rules have been discovered after applying Raising. This dissertation thoroughly describes the development and analysis of the Raising method. Moreover, a new structure, called Intersection Ontology, is introduced to represent customer groups on demand. Only needed customer nodes are created. Such an ontology is used to simplify the marketing knowledge representation. Finally, some additional ontology usages are mentioned. By integrating an ontology into Web marketing, the marketing process support has been greatly improved.

ENHANCING WEB MARKETING BY USING AN ONTOLOGY

by
Xuan Zhou

A Dissertation
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy in Computer Science

Department of Computer Science

May 2006

APPROVAL PAGE

ENHANCING WEB MARKETING BY USING AN ONTOLOGY

Xuan Zhou

Dr. James Geller, Dissertation Advisor — Date
Professor, New Jersey Institute of Technology

Dr. Yehoshua Perl, Dissertation Co-Advisor — Date
Professor, New Jersey Institute of Technology

Dr. Michael Halper, Committee Member — Date
Professor, Kean University

Dr. Yugyung Lee, Committee Member — Date
Associate Professor, University of Missouri-Kansas City

Dr. David Mendonça , Committee Member — Date
Assistant Professor, New Jersey Institute of Technology

BIOGRAPHICAL SKETCH

Author: Xuan Zhou

Degree: Doctor of Philosophy

Date: May 2006

Undergraduate and Graduate Education:

- Doctor of Philosophy in Computer Science,
  New Jersey Institute of Technology, Newark, NJ, 2006
- Master of Architecture,
  Hunan University, Changsha, Hunan, P.R.China, 2000
- Bachelor of Architecture,
  Hunan University, Changsha, Hunan, P.R.China, 1997

Major: Computer Science

Presentations and Publications:

X. Chen, X. Zhou, R. Scherl and J. Geller, "Using an Interest Ontology for Improved Support in Rule Mining," in *Proceedings of the 5th International Conference, DaWaK 2003*, pp. 320-329, 2003.

J. Geller, X. Zhou, K. Prathipati, S. Kanigiluppai and X. Chen, "Raising data for improved support in rule mining: How to raise and how far to raise," *Intelligent Data Analysis*, vol. 9, no. 4, pp. 397-415, 2005.

X. Zhou, J. Geller, Y. Perl and M. Halper, "An Application Intersection Marketing Ontology," in *Theoretical Computer Science: Essays in Memory of Shimon Even; Lecture Notes in Computer Science , Vol. 3895*, pp. 143-153, 2006.

X. Zhou and J. Geller, "Raising, to Enhance Rule Mining Using an Ontology," in *Data Mining with Ontologies: Implementations, Findings and Frameworks*, 2007. In Preparation; Chapter Proposal was Accepted.

To my beloved wife, Wenzhen
and daughter, Allison.

ACKNOWLEDGMENT

First and foremost, I would like to express my deep and sincere gratitude to Dr. James Geller and Dr. Yehoshua Perl for all the advice and instructions about my research work and about how to become a qualified researcher. Thanks for their guidance and support throughout these years. I am also deeply grateful to Dr. Michael Halper, Dr. Yugyung Lee, and Dr. David Mendonça for their detailed and constructive comments.

A big thanks to Dr. Huangying Gu and Dr. Edward Sarian for their help and support during my PhD studies.

Many thanks to my fellow students in the Web Marketing Project who worked in my group and on other modules. All the discussions and interaction with them made my studies more enjoyable and effective. Special thanks to Xiaoming Chen, Kalpana Prathipati, and Sripriya Kanigiluppai for their work on "Raising," and to Edwin Portscher for his work on "Glossary."

I also would like to express my gratitude to my parents and my in-laws for having faith in me and for their help in taking care of my daughter.

Most of all, I would like to thank my wife, Wenzhen Cai, for her unconditional love, encouragement, support, and for making it all possible. Thanks also to my daughter, Allison Cai Zhou, who brings me so much happiness.

TABLE OF CONTENTS

TABLE OF CONTENTS
(Continued)

TABLE OF CONTENTS
(Continued)

Chapter **Page**

LIST OF TABLES

LIST OF FIGURES

LIST OF FIGURES
(Continued)

Figure **Page**

CHAPTER 1

INTRODUCTION

1.1 Problem Description

Marketing has faced new challenges over the past decade. The days of the mass market are definitely over. In the sixties it was sufficient to place an advertisement on one of the major TV channels, and people would be exposed to them. The population also appeared to be more homogenous, in that products tended to be attractive to large groups of people. Today, both these factors have changed.

Consumers now are exposed to numerous cable channels and satellite channels. Few people are "married to a TV network." Many people do not get their information from TV at all, but use Web sites. The population has also developed. Minorities have grown and asserted their own tastes and needs. A product that is attractive to the average white Anglo-Saxon or Italian citizen might be completely uninteresting to a first generation South American immigrant. Similarly, the market has split up by preferences. Sushi can be found at every corner. Chinese and Indian food have made major inroads and many consumers would like to cook the same food in their homes. In short, the mass market is dead, and marketers today face the problem to advertise to many disjoint niche markets.

With the increase in available, cheap data storage, it is understood that companies are keeping terabytes of information about their customers. Today it is not outrageous anymore to talk about one-to-one marketing. However, marketers face two other problems. At the best of times, they may have information about previous customers. But how could they get personal information about potential customers? Secondly, if information about individuals is truly not accessible, how could they classify such individuals into small categories and then market effectively to these small categories?

This dissertation presents a study of these two problems, finding information about individuals and about preferences of small groups of individuals by combining Web extraction with Ontologies and Data Mining algorithms.

There are millions of home pages on the Web. People freely express their likes and dislikes on their home pages. By posting these home pages, they make themselves available online and create their own identity on the Web.

These pages are a valuable source of data for marketing purposes. One approach is to use the contact information for direct (email) marketing. For example, if someone is interested in music, then he/she might want to buy CDs. Thus the marketing can be directed towards a very narrow niche. If someone is interested in Jennifer Lopez, then he/she can be targeted for a particular CD containing recordings of the singer or for a biography that may be of interest to a fan.

A second important use of this data is for research relevant to marketing. The data may be mined for useful correlations between interests and also between demographic categories. If someone is interested in Jennifer Lopez, what is the likelihood that he/she is interested in the music of another singer? What age groups are interested in particular types of music? The available data can be used for such investigations. The results may again be useful for marketing. For example, it may be determined that people interested in Jennifer Lopez are generally interested in certain other singers. But additionally, the results may be of general sociological interest.

1.2 Web Marketing Project Review

Originating from and returning to the Web, the Web Marketing system consists of six modules as shown in Figure 1.1, two of which are of special interests in this dissertation. A quick review of the whole system is presented as follows.

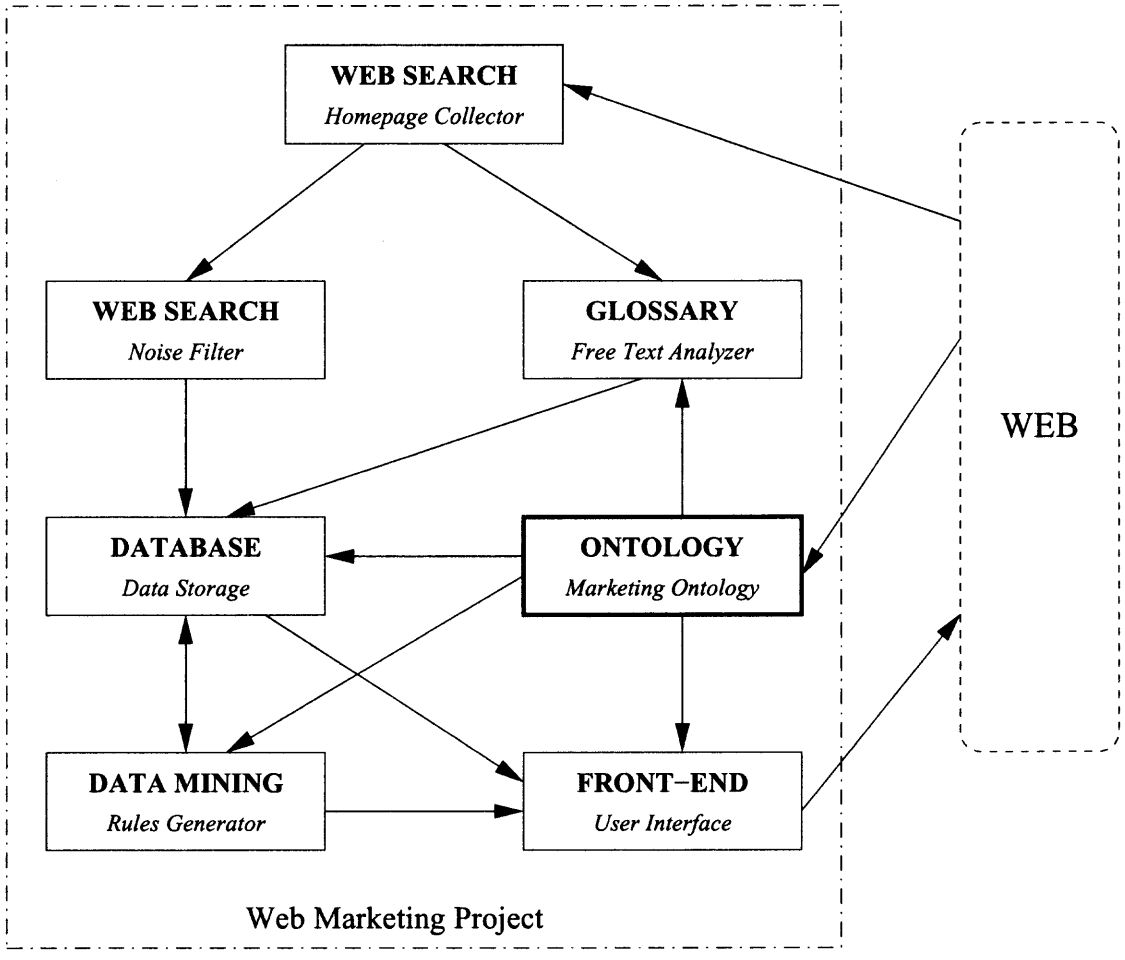


Figure 1.1 Overall Web Marketing Project architecture.

Web Search Module The Web search module extracts home pages of users from several portal sites. The following portal sites have been used: LiveJournal, ICQ, and Yahoo, as well as a few major universities, as listed in Table 1.1. Due to the relative sizes of these portal sites and the content qualities, most of the useful data are from Yahoo. The data had to be cleaned, which is a labor intensive process, as home page users often try to be funny and creative instead of being factual and informative. For instance, many home page owners list their names as "Ask Me" or a variation of this. The programming language used to extract Web pages is Compaq's (now HP's) Web Language (WebL) [2]. It is an JAVA-based imperative, interpreted language that has built-in support for common Web protocols like HTTP and FTP, and popular data types like HTML and XML. Useful

information on a Web page (home page) were automatically extracted and stored. In a typical Yahoo profile page, like in Figure 1.2, the page owner has listed demographic information and interest information. The name, email address, gender, age, marital status, location, occupation, interests, and hobbies can be collected from such a Web page and be used in Web marketing.

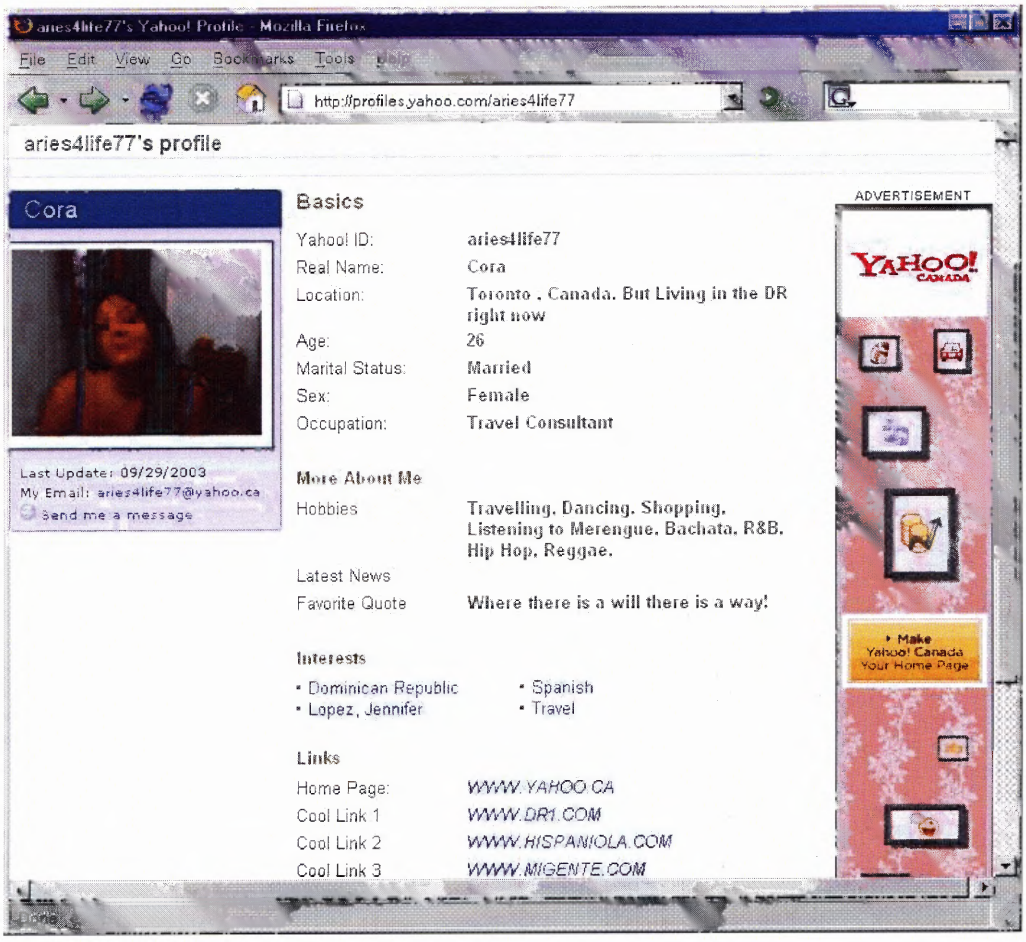


Figure 1.2 One Yahoo profile Web page.

Database Module The Object-Relational database stores the results of this search and cleaning process. Oracle 9.0 is used for this purpose. The raw data is stored in traditional tables. Data cleaning was preformed both before and after entering data to deal with the real

Table 1.1 Universities searched for home pages

1	Boston University
2	California State University, Sacramento
3	California Institute of Technology
4	Cornell University
5	Duke University
6	Harvard University
7	Massachusetts Institute of Technology
8	New Jersey Institute of Technology
9	Pennsylvania State University
10	Rutgers, The State University of New Jersey
11	Stanford University
12	University of Arizona
13	University of Arkansas
14	University of Illinois at Urbana-Champaign
15	University of Mississippi
16	University of Southern California
17	University of van Amsterdam
18	Virginia Polytechnic Institute and State University

world dirty data. However, the same database is also used to store the results of the Web mining process in the form of rules. The insertion of rules into the tables makes use of the object-oriented features of the Object-Relational database system. Lastly, the ontologies are also stored in tables. The storage of the Yahoo interest ontology will be described in detail in Chapter 2.

Data Mining Module The data mining module uses well-known data mining algorithms to extract association rules from given data. The WEKA [3] package was used at the beginning of the project. From the WEKA package, the Apriori algorithm [4] for data mining was used. However, the large sizes of the datasets in this research and the syntactic rules of WEKA forced the performance of an additional cleaning step (e.g., eliminating commas from real data). Thus, even this straight forward task was not trivial. The real world data about real people tends to produce rules with unsatisfactory support values. Thus, in this research a method has been developed for improving the support values of rules by using the ontology. This method is called "*Raising* " and will be discussed in depth in Chapter 3. Moreover, due to the limitations of WEKA found during the project, the FP-Growth algorithm [5, 6] was used in the second stage to correct some errors and improve the results. More details about this will be covered in Chapter 4.

Ontology Module The term "ontology" refers to an engineering artifact, constituted by a specific vocabulary, used to describe a certain reality, plus a set of explicit assumptions regarding the intended meaning of the vocabulary words [7, 8]. The ontology is the main knowledge representation mechanism of this project. It consists of two taxonomies, one of which describes different customer classifications, while the other one contains a large hierarchy of interests. Interest hierarchies have been built based on Yahoo, ICQ and LiveJournal. Only the larger Yahoo interest hierarchy is being used for mining. However, in Chapter 8, the problem of integrating these hierarchies will also be discussed. For the

customer classification, an intersection ontology [9] was developed in Chapter 6 to build the customer hierarchy on demand.

Glossary Module The advanced extraction component processes unstructured Web pages which do not follow structure rules. While Web pages accessible through Web portals typically have very well defined structures that make it easy to locate age and interest information (if given at all), this is not the case for many home pages at universities or from Internet Service Providers (ISP). For such home pages, interest information needs to be extracted by using an additional knowledge base. The details of this module which uses Web-based glossaries are described in [10]. Glossaries, which are freely available on the Internet, are used to determine interests from home pages. Processing of these glossaries can be automated and requires little human effort and time. Once the terms have been extracted from these glossaries, they can be used to infer interests from the home pages of Web users.

There are internet glossaries on every imaginable topic. They are also very easy to find, a simple Google search reveals many results. Also glossaries tend to have regular structures. As can be seen from the sample glossary in Figure 1.3, the terms that are defined are usually in bold or highlighted in some way. This makes it easy to automate the extraction of the glossary terms. To build a glossary for a topic, for example "golf," Google was used to search for the term "golf glossary." The first 30 hits of this search returned distinct golf glossaries. Naturally, there was a good degree of overlap between those glossaries, but some of them contained words rarely found in any of the other glossaries. In this research, a program was written which combines the results into a single glossary. A set of 30 glossaries has been generated.

After glossaries are created, a classifier is used to analyze those unstructured home pages containing free text. The classifier loads all glossary files. Each glossary file is hashed into a different hash table. The Web crawler then takes over and visits every

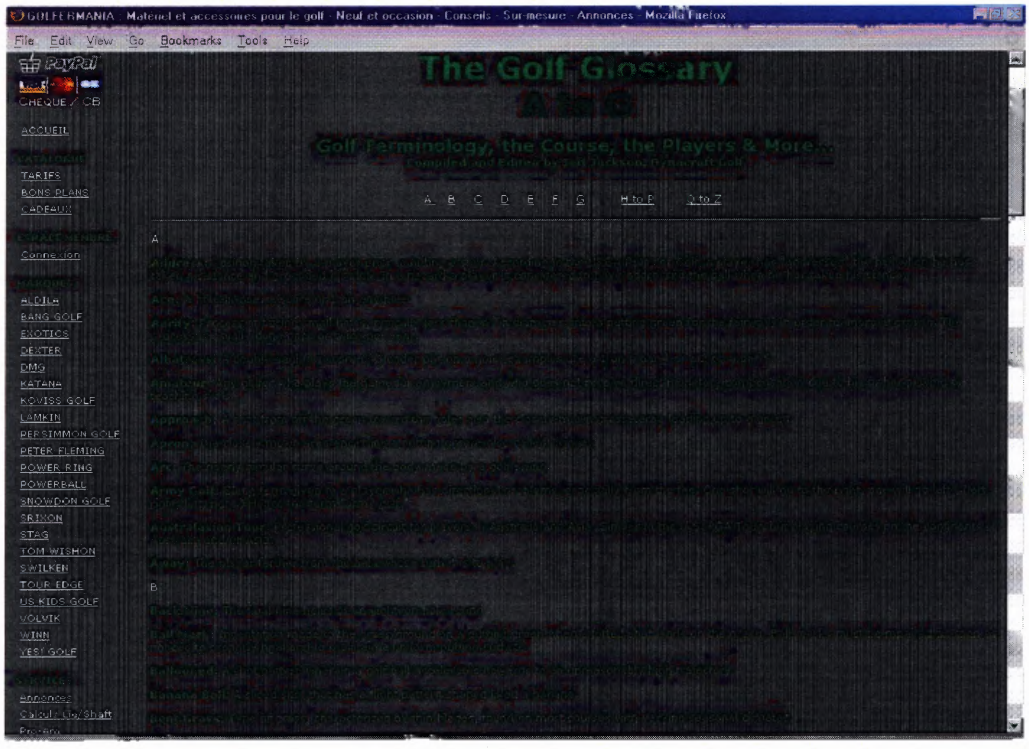


Figure 1.3 A golf glossary on the Web.

home pages. It extracts the words from the HTML page, including words from the Meta tags. These words are then compared against the glossary hash tables in a sliding window sequence from one word to seven words in length. When matches between a word in the Web page and a word in the glossary hash table occur, the word or words and the glossary that they occurred in are recorded. At the end of the page the results are tallied and written to a final output file.

The result of the classifier for a given home page consists of a list of pairs ((glossary topic 1, number of word matches 1), (glossary topic 2, number of word matches 2) ....). Ideally, the glossary topic with the largest number of word matches should be identical to the topic of the home page that is being classified.

Front-End Module The front-end is a user-friendly, Web-based GUI that allows users with no knowledge of SQL to query both the raw data and the rules stored in the tables.

By visiting the Web Marketing Project home page [11], as shown in Figure 1.4, users have access to interest searches and rule searches. In the "Interests of Individual Web Users" section as in Figure 1.5, users may select data by choosing a desired age range from a menu. They may also select a gender, and one or several of many interests from a system of exploding menus. More options can also be selected for customization, which include Occupation, Language, Marital Status, Query Term Combination Methods and Query Match methods. Due to the size of the screenshot, not all options are shown. However, all the options are listed below.

- Most Popular interests: List the most popular interests that have been queried recently.
- Select Interest(s): Choose interests from a pop-up menu which was created according to the ontology hierarchy.
- Enter the Occupation to search: Manually input an occupation.
- Select AND/OR of interests: When several interests are chosen, how those query terms are combined logically.
- Select a Gender: Choose a gender.
- Select an Age: Choose an age range. Several disconnected ranges may be chosen.
- Select Language: Choose a language.
- Select Match: Choose whether the query uses String match or Exact match. Details will be described in Chapter 2.
- Select Marital Status: Choose a Marital Status.

The results from the query are shown in the Web browser as in Figure 1.8. The information of people who are interested in "LAW" are displayed in several pages. For example, Eddie is an 18 years old man who lives in CA and is interested in "LAW." Demographic information of people is shown in the table. However, user can choose what information to display in the previous Web page. The Figure 1.8 displays most of

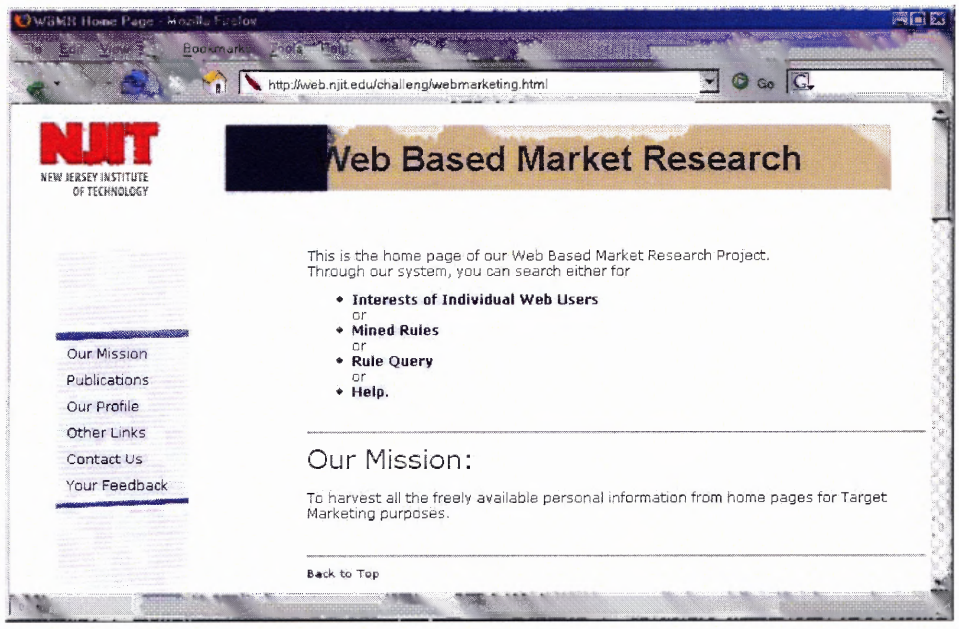


Figure 1.4 Web Marketing Project home page.

the options: Name, Email, Language, Age, Gender, Occupation, Marital Status, Address, Interest (but not URL). In the "Mined Rules" pages, one of which is shown in Figure 1.6, all mined rules are listed by interest categories and by level numbers. Rules are listed in descending order of their confidence values. In the "Rule Query" page, as shown in Figure 1.7, users are able to query rules retrieved by support and confidence value thresholds or by specific interests selected. Those rules which satisfy the conditions will be shown in response to the query.

1.3 What are Ontologies?

Ontology is defined as the branch of philosophy concerned with the study of the nature of being. However, when computer scientists refer to "an ontology" they mean a computer implementation of human-like knowledge.

Ontologies are descendants of the semantic networks in Artificial Intelligence. Quillian's first semantic network in 1968 was a computer implementation of a dictionary [12]. Terms

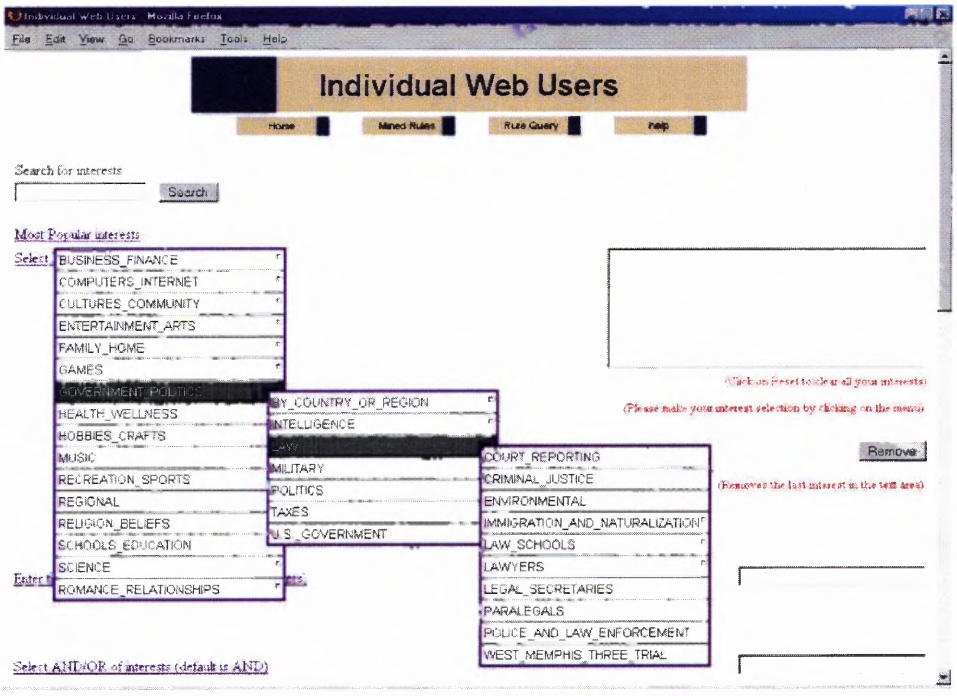


Figure 1.5 Web Marketing Project personal interest query page.

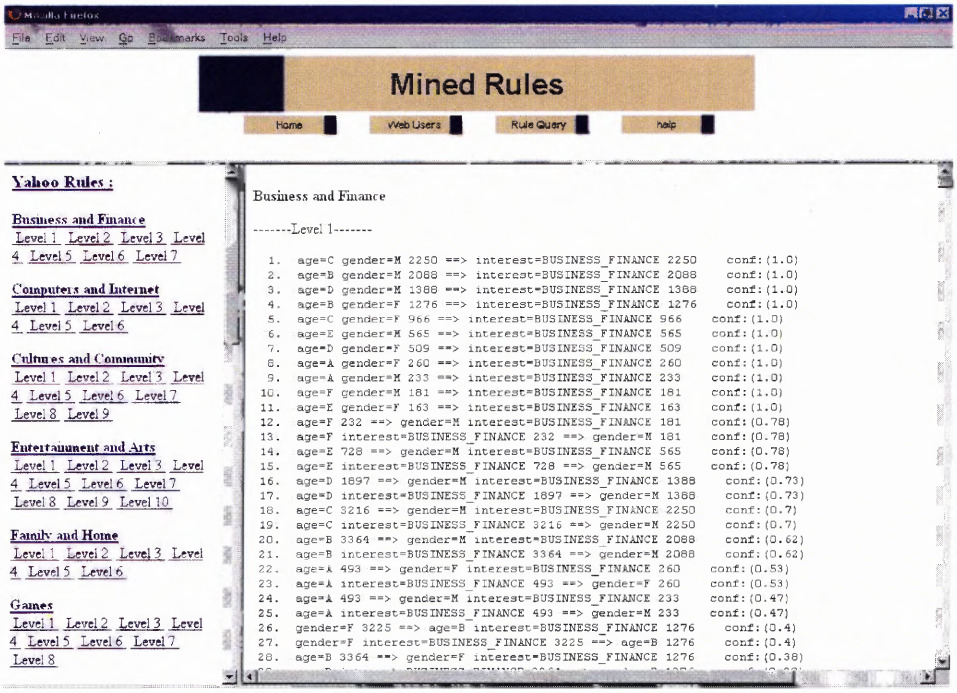


Figure 1.6 Web Marketing Project mined rules lists page.

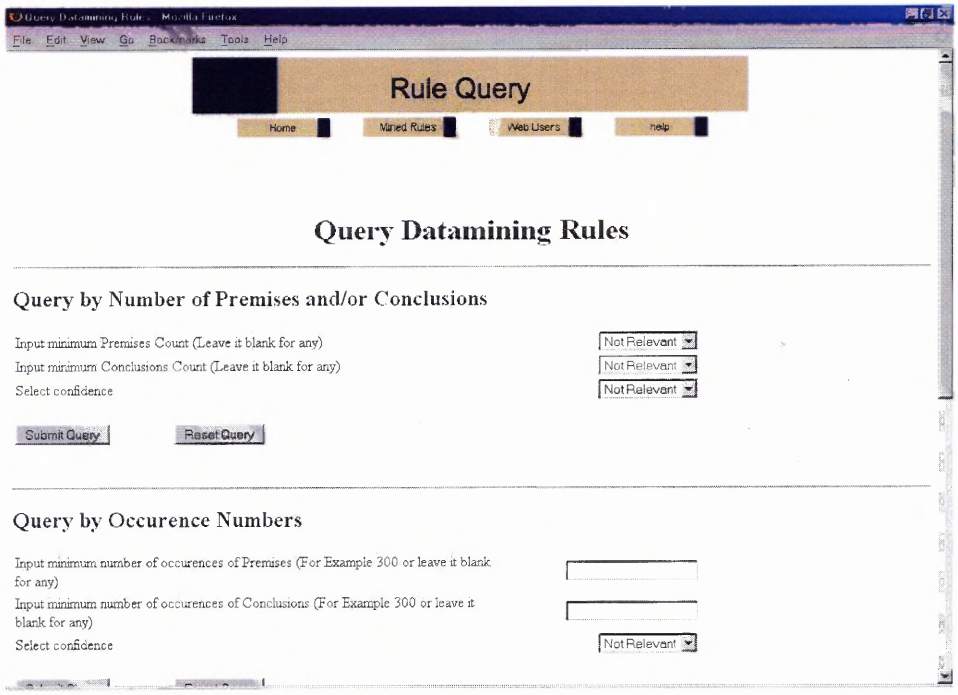
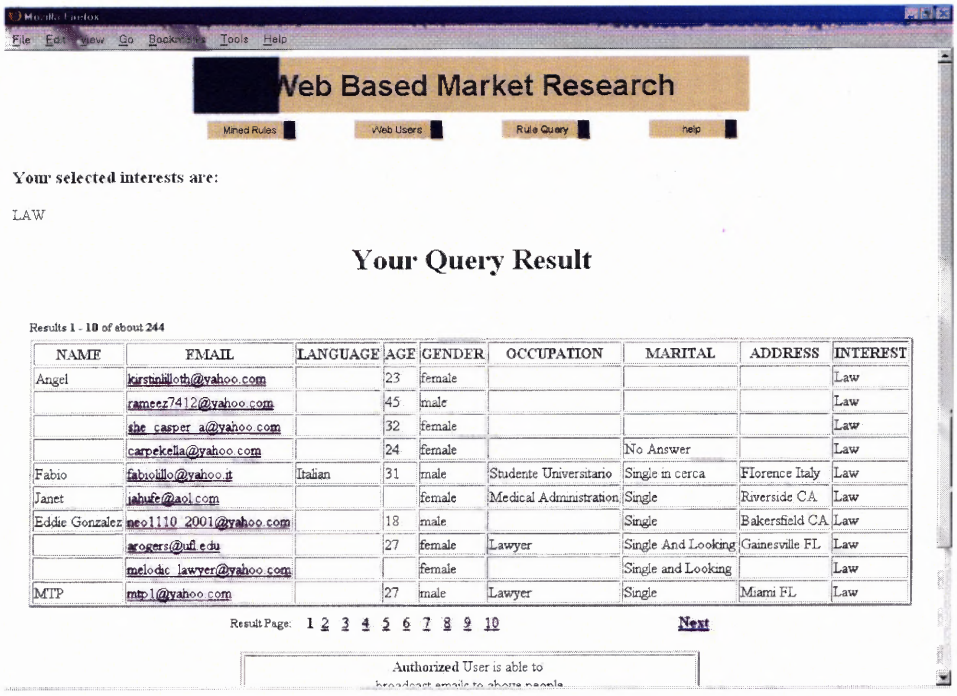


Figure 1.7 Web Marketing Project mined rules query page.



NAME	EMAIL	LANGUAGE	AGE	GENDER	OCCUPATION	MARITAL	ADDRESS	INTEREST
Angel	kirstinilloth@yahoo.com		23	female				Law
	rameez7412@yahoo.com		45	male				Law
	the_casper_a@yahoo.com		32	female				Law
	carpekella@yahoo.com		24	female		No Answer		Law
Fabio	fabiolillo@yahoo.it	Italian	31	male	Studente Universitario	Single in cerca	Florence Italy	Law
Janet	jabufe@aol.com			female	Medical Administration	Single	Riverside CA	Law
Eddie Gonzalez	neo1110_2001@yahoo.com		18	male		Single	Bakersfield CA	Law
	arogers@ufl.edu		27	female	Lawyer	Single And Looking	Gainesville FL	Law
	melodic_lawyer@yahoo.com			female		Single and Looking		Law
MTP	mtp1@yahoo.com		27	male	Lawyer	Single	Miami FL	Law

Figure 1.8 Web Marketing Project interest query results by exact match.

in dictionaries refer to other terms, and Quillian implemented these references by pointers. However, as one term could have different meanings, a distinction is made between terms and concepts. Concepts are the fundamental building blocks of all semantic networks and ontologies.

A concept is a basic unit of knowledge, and, as opposed to a term, a concept is unambiguous. Quillian used only a small number of kinds of links which have been extensively studied and greatly refined since then. The most fundamental of these links, describes a generalization/specialization relationship between two concepts. This relationship satisfies transitivity. It has been variously called IS-A, sub-concept, subclass, a-kind-of, *etc.* It allows property inheritance, as described below.

Humans have additional "local" information about concepts. For example, solid objects have color, size, *etc.* This kind of local information is called "attributes," "properties" or "slots". If a general concept has an attribute (vehicles have a weight), then a specific sub-concept will have the same property (cars have a weight). One can imagine that inheritance is the propagation of a property from the general concept to the more specific concept against the direction of the IS-A link. Besides the IS-A links, ontologies contain other links, e.g., *likes*, *owns*, *connected-to*, *etc.* Most of these additional links have no "built-in behavior." These links are variously called associative relationships, roles, semantic relationships, *etc.* and are labeled by their names. Relationships are inherited down along IS-A links.

Because a concept cannot be more general than itself, and because of the transitivity of the IS-A links, there cannot be any cycles of IS-A links in a semantic network. Furthermore, it is practical to have one concept (often called THING) that is a generalization of every concept in an ontology. Thus, the concepts and IS-A links in an ontology form a hierarchy with a root. In other words, the hierarchy of an ontology is a rooted Directed Acyclic Graph (DAG), where the nodes represent the concepts and the links represent IS-A relationships. Furthermore, the concepts and the IS-A links together form a weakly connected component.

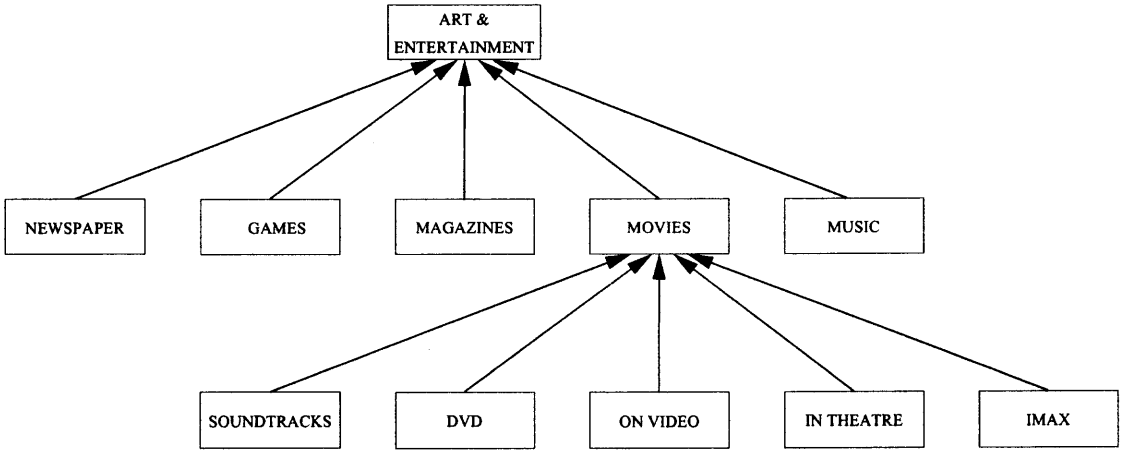


Figure 1.9 A partial interest hierarchy (tree).

The definition of an ontology as a graph results in a natural diagram representation for ontologies. Figure 1.9 shows an example of an ontology. This example is adapted from [13] by eliminating other relationships such as part-of. In this and later figures, every box stands for a concept. Bold arrows (typically pointing upwards) stand for IS-A relationships. Thin arrows stand for other relationships. The IS-A relationships in this example form a tree. However, most of the time, as in examples seen later, DAGs are used. Family terms, such as *child*, *ancestor* and *descendant* are used to describe the DAG. A number of other extensions exist for ontologies, e.g. rules or axioms. However, these are not used in our model of an ontology and will be omitted.

Other definitions of Ontology as found in Dictionary.com [14] are:

1. <philosophy> A systematic account of Existence.

2. <artificial intelligence> (From philosophy) An explicit formal specification of how to represent the objects, concepts and other entities that are assumed to exist in some area of interest and the relationships that hold among them.

   For AI systems, what "exists" is that which can be represented. When the knowledge about a domain is represented in a declarative language, the set of objects that can be represented is called the universe of discourse. The ontology of a program can

be described by defining a set of representational terms. Definitions associate the names of entities in the universe of discourse (e.g. classes, relations, functions or other objects) with human-readable text describing what the names mean, and formal axioms that constrain the interpretation and well-formed use of these terms. Formally, an ontology is the statement of a logical theory.

A set of agents that share the same ontology will be able to communicate about a domain of discourse without necessarily operating on a globally shared theory. An agent commits to an ontology if its observable actions are consistent with the definitions in the ontology. The idea of ontological commitment is based on the Knowledge-Level perspective.

3. <information science> The hierarchical structuring of knowledge about things by subcategorizing them according to their essential (or at least relevant and/or cognitive) qualities. This is an extension of the previous senses of "ontology" (above) which has become common in discussions about the difficulty of maintaining subject indices.

In this research, "ontology" is used as in the third definition. Concepts are structured in hierarchies of IS-A relationships. Thus, the definition of an ontology is presented as follows: An ontology is a directed graph of nodes, which represent concepts, and edges, which represent IS-A and semantic relationships between pairs of nodes. Concepts are labeled by unique terms. Concepts have additional (name, value) pairs, called attributes, where the attribute name needs to be unique for each concept. The set of all concepts together with the set of all IS-A links form a rooted, connected, Directed Acyclic Subgraph of the ontology. This subgraph is called the *taxonomy* of the ontology. Both attributes and semantic relationships may be inherited downwards, against the direction of the IS-A links, from more general concepts to more specific concepts.

Modern ontologies are attributed to Thomas Gruber [15] who built on a rich history which is briefly reviewed in [16]. Ontology building deals with modeling the world with

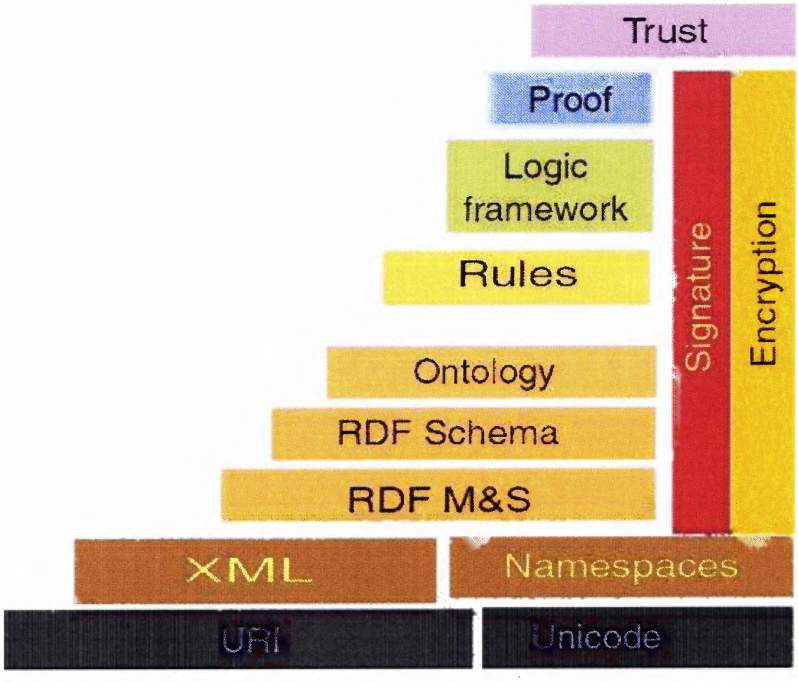


Figure 1.10 Semantic Web layer cake [1].

shareable knowledge structures. With the emergence of the Semantic Web, the development of ontologies and ontology integration have become very important [17, 18, 19, 20]. The Semantic Web is a vision, for a next generation Web, of Tim Berners-Lee, the inventor of the original Web, and colleagues. This vision is described in a figure called the "layer cake" of the Semantic Web [17, 1]. This figure, shown as Figure 1.10, consists of nine functional layers of increasing technical complexity and abstraction. Each layer supports all the layers above it. Ontologies are flush in the middle of the layer cake. All the layers below Ontology, such as XML and RDF Schema are well developed. All the layers above Ontology, such as Rules and Proofs are well established within Artificial Intelligence (AI), but do not exist in widely applicable form outside of AI.

Ontologies will be used in the Semantic Web as follows. The current Web has shown that string matching by itself is often not sufficient for finding specific concepts. Rather, special programs are needed that search the Web for the concepts specified by a user. Such

programs, which are activated once and traverse the Web without further supervision, are called agent programs.

Successful agent programs will search for concepts as opposed to words. Due to the well known homonym and synonym problems, it is difficult to select between different concepts expressed by the same word (e.g., Jaguar the animal, or Jaguar the car). However, having additional information about a concept, such as which concepts are related to it, makes it easier to solve this matching problem. For example, if that Jaguar that IS-A car is desired, then the agent knows which of the meanings to look for.

Ontologies provide a repository of this kind of relationship information. To make the creation of the Semantic Web easier, Web page authors will need to derive the terms of their pages from existing ontologies, or develop new ontologies for the Semantic Web.

Many technical problems remain for ontology developers, e.g. scalability. Yet, it is obvious that the Semantic Web will never become a reality if ontologies cannot be developed to the point of functionality, availability and reliability comparable to the existing components of the Web.

Some ontologies are used to represent general world or word knowledge. Other ontologies have been used in a number of specialized areas. An overview of ontologies and their usages and properties can be found in [21, 22]. Also, [23] focuses on the ontology usage in the bioinformatics field while [24] describes ontology work in the ecommerce field. For a comprehensive review of established ontologies see [25]. Two special issues on ontologies are [26, 20].

1.4 Definitions

Level of a Concept All concepts in an ontology with their IS-A relationships form a tree or a DAG (Directed Acyclic Graph) with a single ROOT. The root node is defined as being at level 0. It is simple to find the level of each node in a tree structure by BFS (Breadth First Search). However, to assign levels in a DAG, a tree that spans the DAG structure is

used and level values are assigned to each node. Thus, a node's level value will be the largest number of edges of any path from the root to the node. The detailed algorithm will be introduced in Section 2.3.

Association Rules *Association Rules* are statements in the form

$$\{X_1, X_2, \cdots, X_n\}\ Occur_{ante} \rightarrow \{Y_1, Y_2, \cdots, Y_n\}\ Occur_{ante\&con}$$

meaning that if all of $\{X_1, X_2, \cdots, X_n\}$ are found in a data tuple, then there is a chance, as decided by the confidence value, of finding $\{Y_1, Y_2, \cdots, Y_n\}$. In this representation, $\{X_1, X_2, \cdots, X_n\}$, located before the arrow, is called *antecedent*. $\{Y_1, Y_2, \cdots, Y_n\}$, located after the arrow, is called *consequent*. $Occur_{ante}$ and $Occur_{ante\&con}$ are two natural numbers. $Occur_{ante}$ stands for the number of occurrences of the antecedent among the whole dataset while $Occur_{ante\&con}$ stands for the number of occurrences of the antecedent and the consequent together among the whole dataset. In other words, $Occur_{ante}$ is the number of tuples which include all of $\{X_1, X_2, \cdots, X_n\}$ and $Occur_{ante\&con}$ is the number of tuples which include all of $\{X_1, X_2, \cdots, X_n, Y_1, Y_2, \cdots, Y_n\}$.

Support The *support*, also called *coverage*, is the number of instances for which the rule predicts correctly. The value can be presented in two ways.

1. As absolute value of the number of occurrences of both the antecedent and consequent. Thus,

$$supp = Occur_{ante\&con}$$

2. As relative value that shows the proportion of the occurrences of both the antecedent and consequent among the whole dataset. Thus,

$$supp_{rel} = Occur_{ante\&con}/Total$$

in which $Total$ is the number of all tuples in the dataset.

In this dissertation, the absolute support is used to describe an association rule and to analyze results. However, the relative support is also used in the implementation, as will be described in Chapter 7.

Confidence The *confidence* value of the rule is the probability of finding the consequent of this rule in all instances (tuples) including the antecedent. The *confidence*, also called *accuracy*, stands for the correct prediction based on all instances it applies to [3]. The *confidence* value is always represented as a fraction:

$$conf = Occur_{ante\&con}/Occur_{ante}$$

Because of the definitions of $Occur_{ante\&con}$ and $Occur_{ante}$, $Occur_{ante\&con}$ is always not greater than $Occur_{ante}$. Thus, the *confidence* value of a rule is always

$$1 \geq conf \geq 0$$

Both the support value and the confidence value are two key factors when a rule is judged whether it is useful or not. On one hand, a rule with a high support value but a low confidence value cannot reliably predict the consequent from the antecedent. For example, among 10,000 people who bought rice, two of them also bought toothbrushes. A rule indicating a relationship between rice and toothbrush purchases would not help a lot with predicting future purchases. One the other hand, a rule with a high confidence value but a low support value is not representative. Assume now that among 10,000 people, only 10 people bought rice and those 10 also bought toothbrushes. This will generate a rule with a perfect confidence value about the relationship between rice and toothbrush purchases.

However, the 10 people do not represent those 10,000 persons well. Normally, only the rules that have confidence and support above certain thresholds are useful for research or marketing practice.

Raising *Raising* is a method invented in this research, which is used before data mining to increase the support of the resulting rules. Details of Raising will be described in Chapter 3 and Chapter 4.

Interest Score *Interest Score* is used as a measure to show how close two concepts are in an ontology hierarchy. As will be described in Chapter 8, interest scores can be used for ranking retrieved results.

CHAPTER 2

ONTOLOGY MODULE IN WEB MARKETING PROJECT

2.1 Problem Description

It was decided that the best way to represent the necessary knowledge for the Web Marketing Project would be to use an ontology as the backbone. However, before using an ontology, the ontology needs to be constructed from scratch and be stored somewhere for further accesses. Moreover, since real-world data is dealt with, some cleaning is necessary.

The problems arise from the very beginning. First of all, the contents to build the ontology need to be found. In this research, people's interests are used for marketing purposes. Since an interest is likely to imply a purchase, it is well suited for the purpose of predicting future purchases. How to get the interest information to construct the ontology is the first problem to be solved.

Secondly, when interests for an ontology have been derived, they need to be stored locally for further usage. All other modules in the Web Marketing Project need to communicate with the ontology which is stored in a database. Database related problems had to be taken care of, such as database design.

Last but not least, the interest information and related demographic information are from the real world. It is not surprising to find improper uses and errors in it. Thus, the problem of cleaning data, both before and after storage, had to be dealt with.

2.2 Ontology Web Extraction

The Yahoo Web portal [27] has been developed by experts. The DMOZ Open Directory Project (ODP) [28] is a human edited directory of the Web, compiled by a vast global community of volunteer editors. Personal information is one of the key ingredients for the Web marketing project. There was a need for a collection of people's demographic

information and interest information. Thus, the Yahoo member directory was chosen as primary information source.

The Yahoo! Member Directory (http://members.yahoo.com/) has 16 top level interest categories:

1. Business & Finance
2. Computers & Internet
3. Cultures & Community
4. Entertainment & Arts
5. Family & Home
6. Games
7. Government & Politics
8. Health & Wellness
9. Hobbies & Crafts
10. Music
11. Recreation & Sports
12. Regional
13. Religion & Beliefs
14. Romance & Relationships
15. Schools & Education
16. Science

Under each category, interests belonging to that category are listed. For most of the interests, there are unique IDs, which are 10-digit numbers starting with "16". These IDs are called Yahoo IDs, and they are used to identify different interests. However, some interests do not have a Yahoo ID, such as the 16 interests at level 1. Those interests are general interests. In this case, a Simulated Yahoo ID was created in this project and assigned to those interests. For example, the level 1 interest "Business & Finance" was assigned an ID 2000000001. Why Yahoo chose such an inconsistent representation is unknown. However, one assumption is that only those interests, which has been chosen by people as their personal interests, are assigned a Yahoo ID. That explains why some general interests do not have IDs since people prefer to choose more detailed interests such as "CLASSICAL MUSIC" instead of general interests such as "MUSIC."

An interest hierarchy was constructed by following the links in the Yahoo! Member Directory. The hierarchy starts at level 0 with the root "INTEREST." All the 16 interest categories are located at the level 1. In other words, there are 16 nodes at level 1 which include all interest categories from the Yahoo! Member Directory. Following the links, the whole hierarchy was constructed level by level. The whole retrieval process was done automatically by program. The extraction program was written in WebL, a Web Language developed by Compaq, which was bought out by HP [2]. WebL is implemented in Java. It is a scripting language for automating tasks on the Web. It has nice support for Web protocols such as HTTP and provides utilities for conveniently analyzing HTML pages.

A recursive WebL program was written to traverse through the Yahoo! Member Directory from the starting page and collecting all the interests and their associated Yahoo IDs. ICQ.com and LiveJournal.com were also used for Web retrieval. Extraction programs in WebL were written to collect the interests from both sites. However, since the interest hierarchy in ICQ.com has only two levels and the interests hierarchy in LiveJournal.com only has one level, the extraction process is comparably simpler than that of Yahoo.com.

2.3 Levels in an Ontology

A *level* number in an ontology refers to the location identification of a concept. All concepts in the interest ontology hierarchy constructed above form a tree or a DAG (Directed Acyclic Graph) with a single ROOT. However, depending on different research approaches, the assignment of level numbers to concepts in an ontology can be different. Not mentioning the additional complications of a DAG, the root node can be assigned either level 0 or level 1. In this research, the root node is defined as being at level 0. To assign levels in a DAG, a tree that spans the DAG structure is used and level values are assigned to each node. Thus, a node's level value will be the largest number of edges of any path from the root to the node.

A revised BFS algorithm was thus developed by the author and used to assign suitable level numbers as described. The following is the LEVEL-BFS algorithm, revised from the BFS in [29], to assign a level number to each node in a DAG.

Graph G = {V, E} is a Rooted DAG with the Root *r*

```
LEVEL-BFS(G, r)
        // Initialize the level of each node.
        for each vertex u ∈ V[G]
            do level[u] ← ∞
        // Start the algorithm from the Root.
        level[r] ← 0
        //A first-in, first-out queue Q is used for temporary storage
        ENQUEUE(Q, r)
        while !EMPTY[Q]
              do u ← head[Q]
                 for each v in edge (v, u) ∈ E[G]
                     //Assign a level value only if all its parents
                     //have already been assigned level values
                     do if level[v] = ∞ and
                                    ∀w in edge (v, w) ∈ E[G], level[w] ≠ ∞
                        then level[v] ← level[u] + 1
                             ENQUEUE(Q, v)
                 //Remove the head node from the queue.
                 //The node has been assigned a level value and
                 //all its adjacent nodes have been traversed.
```

DEQUEUE(Q)

Figure 2.1 is a simple example of such an assignment. The node E is visited twice during the algorithm and is assigned a level number 3 according to the longer path through the parent G. When comparing levels of different nodes, the nodes that are closer to the ROOT are at higher levels in the hierarchy. Thus, in Figure 2.1, node F has a higher level than node E. Note that higher levels mean smaller level numbers.

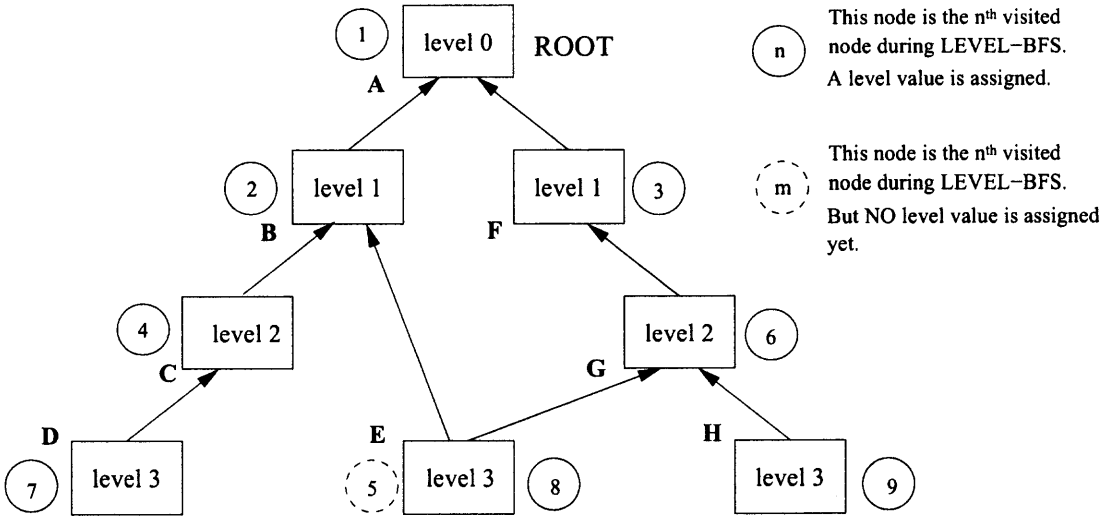


Figure 2.1 An example of assigning levels by LEVEL-BFS.

2.4 Ontology Storage

After the ontology had been extracted from the Web, the interest information had to be stored locally to be used by the programs. However, since the ontology hierarchy is a DAG, multi-parent relations are needed to be reflected in the storage, which complicated the representation.

Due to the large size of the hierarchy and the availability of Oracle it was decided to store the interest hierarchy in a database. Several designs were considered and it was decided to use an approach where paths to interests are materialized. This minimizes the need for database traversal.

In the Oracle database, one relational table is used to store the entire ontology hierarchy. The table structure is shown in Table 2.1. 13 columns include a sequence number, a Yahoo ID and columns for interests at each level, if there is one. At the time this work was done, the deepest concept in the Yahoo interest hierarchy was eleven levels deep. The construction of all paths was done as follows. The DAG hierarchy was transferred into a tree. As in Figure 2.2, two trees are represented for a transformation. The original DAG in Figure 2.2(a) needs to be transformed into a tree. There are two choices as shown in Figure 2.2(b) and Figure 2.2(c).

The tree in Figure 2.2(b) repeats all the subtrees for a node with multiple parents for each of the parents. Thus, for node "H" in Figure 2.2(a), the subtree with 4 nodes "H," "I,""J" and "K" is reproduced twice and is placed under "E" and "F" respectively. For the same reason, since the node "I" has a parent "H" and a parent "G," the subtree consisting of "I" and "K" is also repeated under "G." In this kind of transformation, a node with multiple parents is repeated, together with all its descendants in several copies, one copy for each of its parents.

One the other hand, the tree in Figure 2.2(c) does not contain a lot of repetitions. For each multi-parent node, the subtree of its descendants only appears once in the Ontology hierarchy. However, the multi-parent node itself is repeated and placed under each of its parents. For example, the node "H" in Figure 2.2(a) appears twice in Figure 2.2(c) as "H" and "H@." The subtree of descendants only appears under the node "H." This representation fully simulates the hierarchy of the Yahoo Member Directory at the time of retrieval. While an interest may appear at different locations in the hierarchy, its descendants only appear at one location. For the rest of the locations, a symbol of "@" is used to mark the missing descendants. Thus, for an interest "METRICOM INC," two paths are the following:

1. BUSINESS FINANCE→COMPANIES→COMMUNICATIONS AND NETWORKING →TELECOMMUNICATIONS→WIRELESS→CELLULAR PROVIDERS →METRICOM INC@

2. BUSINESS FINANCE→COMPANIES→COMPUTERS→HARDWARE →PERIPHERALS →MODEMS→WIRELESS→METRICOM INC

All people's information can be found by reaching the "METRICOM INC" interest in path #2. Moreover, the "METRICOM INC@" in path #1 also links to the "METRICOM INC" link and same information can be retrieved.

As for the hierarchy in Figure 2.2, the transformation from (a) to (c) was used. Thus, for all the leaves in Figure 2.2(c), the following tuples will be created:

1. SEQ1/ID_of_I/A/B/D/G/I

2. SEQ2/ID_of_K/A/B/E/H/I/K

3. SEQ3/ID_of_J/A/B/E/H/J

4. SEQ4/ID_of_H/A/C/F/H

Every tuple in the table represents an interest-path in the tree. The interest-path is a set of ordered interests starting from a first-level interest to an interest in the tree. Thus, in every tuple, the first-level interest is located in the "Level1" column, the second-level interest is located in the "Level2" interest column, *etc.* All the interest columns below the end of the path are assigned the value "NULL." Every interest in the ontology has an unique Sequence Number and a Yahoo ID.

Table 2.1 Ontology table in database

Sequence	Yahoo ID	Level1	Level2	Level3	...	Level10	Level 11

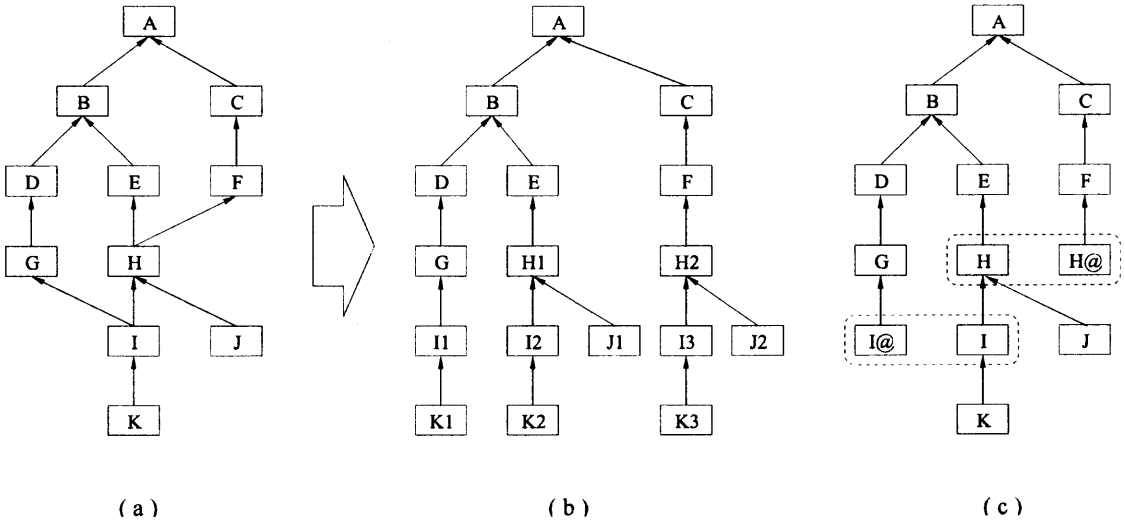


Figure 2.2 Transform an ontology DAG to a tree.

2.5 Data Cleaning

The data used in this research comes from the real world and there is no control at the source. Data has to be cleaned for system functioning. There are two major parts of cleaning, format cleaning and content cleaning.

As for format cleaning, the extracted data has to be translated into the format of the database. Two major actions of cleaning were performed.

The first action was the combining of words whenever necessary. An interest name in the ontology can be one single word, such as "JAZZ," or one term composed from several words, such as "ACID JAZZ." Since one interest is represented by one node in the hierarchy, it was decided that a single word would be made up by simply connecting the words together by underscores. Thus, the interest "ACID JAZZ" was transformed into "ACID_JAZZ" and was stored in the database as a single word.

The other action is about special symbols. Special symbols exist in the interest names, such as ampersand ("&"), hyphen("-") and comma (","), as in "ROCK & POP" and "LOPEZ, JENNIFER." It was decided that special symbols were to be removed as often as possible from the database. There are two main reasons. One reason is for the convenience of database storage. For instance, since symbols such as comma (",") are not allowed to

appear in the column, all the commas in any interest names are removed. The other reason is for the consistency of database querying. For example, ampersand ("&") and the word "and" both appear in the interest names. When a user inputs a query of "ROCK AND POP," the system would return an empty result since the interest name is "ROCK & POP." Thus, all the ampersands ("&") were replaced by the words "and" to avoid such inconsistency.

Data cleaning is not only limited to the ontology module. Though cleaning is also needed in the ontology, some other modules in the Web marketing project also need more cleaning work. However, while cleaning in the ontology is mainly format cleaning, the cleaning of demographic information focuses on content cleaning. Such a situation is caused by the inputting of unedited user data and the fact that many people act irrationally. The information about users was input by themselves when they signed up for their Yahoo service and updated later in their profiles. The sign-up and updating processes are accomplished on Yahoo Web pages. When users input their interests, they need to browse to find the interest category that they are interested in and click on the "join the list" link, as shown in Figure 2.3, to add that specific interest to their profile and also let their ID be included in the list of who is interested in this topic. Thus, users are not able to change the name of each interest and the cleaning work mainly involves formatting. However, the input for demographic information is not like choosing interests. In a profile updating Web page as shown in Figure 2.4, only *Gender* and *Marital Status* are input by choosing from drop-down boxes with multiple options. Most other information is input in textboxes or textareas using free text, such as *Age*, *Location*, *Occupation*, and *Real Name*. People then have the choice to make up nonsensical or offensive information and input it in their profiles. As a result, many data items cannot be true and have to be cleaned.

The *Age* section is the comparably easiest to clean. Since only decimal numbers are involved, it was decided that only ages which are greater than 5 and less than 100 will be accepted as valid ages. The assumption is that those whose ages are out of this range (if alive) will not list their profile online. It is hard to prove the correctness of this assumption

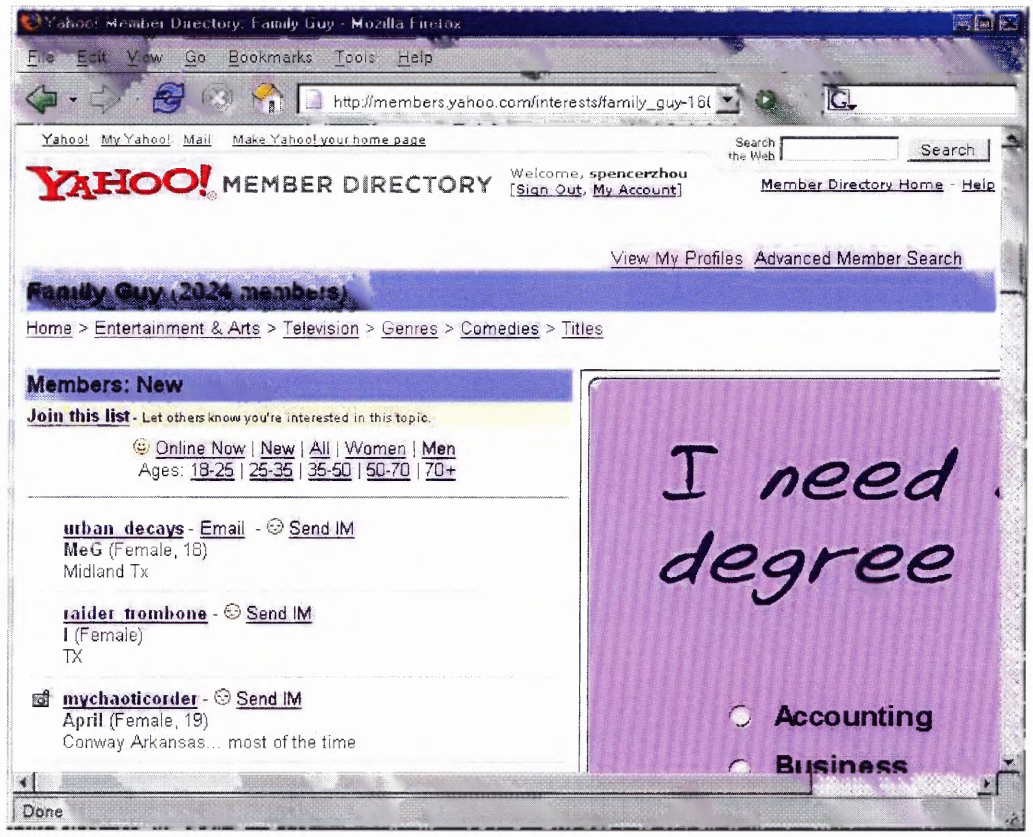


Figure 2.3 To add an interest to a profile by browsing corresponding category.

but the existing exceptions should be very few, if it not none. Moreover, Yahoo itself has improved its mechanism trying to make it more accurate. Instead of asking people's ages while they are signing up for Yahoo services, Yahoo currently asks for the date of birth on the Web page. Thus, the ages can be automatically calculated. Another advantage is that the age will be always accurate without users updating them since the dates of birth will never change. However, as shown in Figure 2.4, Yahoo still keeps the choice for members to update their ages manually. In other words, though Yahoo knows the real age of a member, from his date of birth, a member can change his age freely to be shown on the Web.

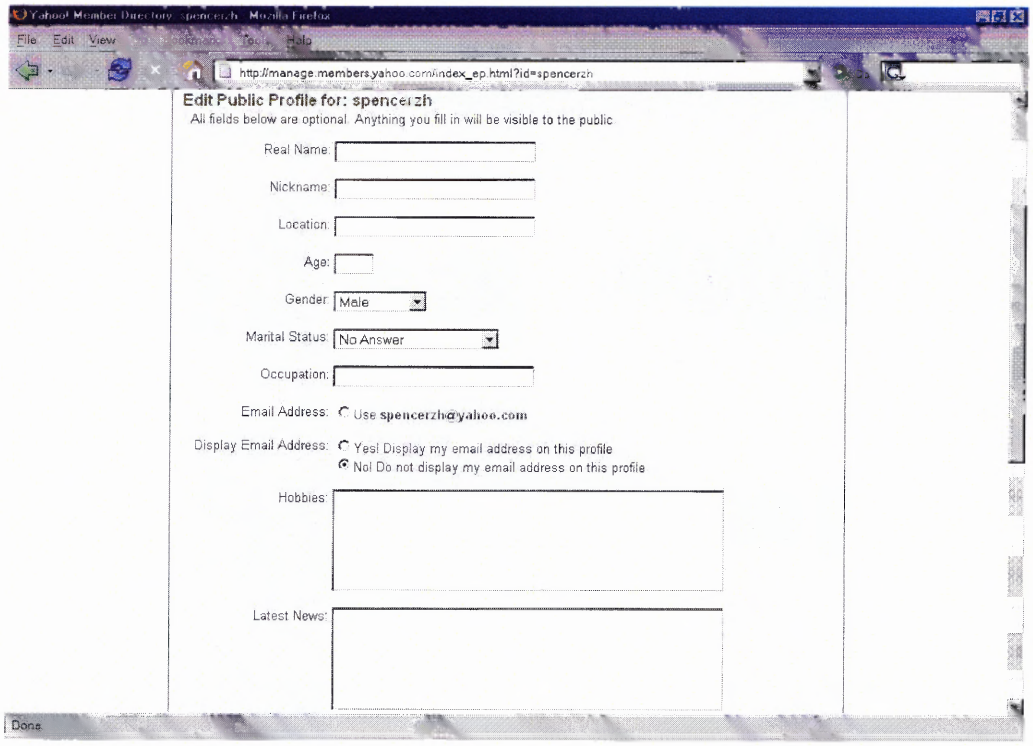


Figure 2.4 To update profile in Yahoo.

The cleaning of other sections is more complicated. The information input by members can be obviously wrong. For example, exactly 74 people list their name as "ask" and 215 people list their name as "ask me," as shown below.

```
SQL> select count(name) from geller.demographic
  2  where name='ask';

  COUNT(NAME)
  -----------
           74

SQL> select count(name) from geller.demographic
  2  where name='ask me';

  COUNT(NAME)
  -----------
          215
```

Table 2.2 Some cleaned terms in database

Section	Location	Occupation	Real Name
Number of cleaned terms	1205	2280	1972
Examples	everywhere	dream chaser	forgot
	nude pic	driving people crazy	sexy
	breast	guy hunter	killer
	dirty	that is irrelevant	creature
	single girl	ruler of universe	devil
	stuck in the mud	sex donation	nameless
	around	God	mystery
	at my computer	secret	me

As the data contains free text and is not structured, it is practically impossible to clean the false data thoroughly programmatically. Thus, some dirty jobs had been done to clean those false data. Columns in the database were reviewed to find obviously wrong data. Once a false term was found, the database was searched to find all occurrences of that term and then clean them. In total, 1205 terms were cleaned for *Location*, 2280 terms were cleaned for *Occupation*, and 1972 terms were cleaned for *Real Name*. Some of the false terms are listed in Table 2.2. Many four letter words were also cleaned from the data, but no examples will be given for those in this dissertation.

CHAPTER 3

DATA MINING AND USING RAISING TO IMPROVE QUALITY OF MINED RULES

3.1 Problem Description

Data mining has become an important research tool for the purpose of marketing. It makes it possible to draw far-reaching conclusions from existing customer databases about connections between different products commonly purchased. If demographic data are available, data mining also allows the generation of rules that connect them with products. However, companies are not only interested in the behavior of their existing customers, they would also like to find out about potential future customers. Typically, there is no information about potential customers available in a company database, that can be used for data mining.

It is possible to perform data mining on potential customers, if one makes the following two adjustments: (1) Many people express their interests freely and explicitly on their Web home pages. (2) Instead of dealing with products already purchased, this research focuses on the interests of customers. There is often a close relationship between specific interests and products, as will be shown below. The process of mining data on potential customers becomes a process of Web Mining.

Applying well-known data mining algorithms to the data extracted from the Web and stored in the database, association rules representing marketing knowledge are derived in this research for marketing purposes. However, when mining this data for association rules, what is available is often too sparse to produce rules with reasonable support values. Thus, some interesting rules were created by data mining, but with fairly low support values. In other words, those rules are not representative enough to predict future purchases. A problem arises about how to improve the quality of those association rules using the existing

data. Since an interest ontology was created in the project, as described in Chapter 2, taking advantage of the ontology hierarchy provides a path to solve this problem.

3.2 Data Mining

Data mining is also known as Knowledge Discovery in Databases (KDD)[1]. In [30], it has been defined as "The nontrivial extraction of implicit, previously unknown, and potentially useful information from data." There are other definitions:

- "Data mining is the analysis of (often large) observational data sets to find unsuspected relationships and to summarize the data in novel ways that are both understandable and useful to the data owner" [31].

- "Data mining is an interdisciplinary field bringing together techniques from machine learning, pattern recognition, statistics, databases, and visualization to address the issue of information extraction from large databases" [32].

- "Data Mining is the task of discovering interesting patterns from large amounts of data where the data can be stored in databases, data warehouses, or other information repositories. It is a young interdisciplinary field, drawing from areas such as database systems, data warehousing, statistics, machine learning, data visualization, information retrieval, and high-performance computing. Other contributing areas include neural networks, pattern recognition, spatial data analysis, image databases, signal processing, and many application fields, such as business, economics, and bioinformatics" [33].

A simple but widely used (though hypothetical) example is that of a very large North American chain of supermarkets. Through intensive analysis of the transactions and the goods bought over a period of time, analysts found that beer and diapers were often bought together. Though explaining this interrelation might be difficult, taking advantage of it,

[1]Some authors consider it one step of KDD only.

on the other hand, should not be hard (e.g. placing the high-profit diapers next to the high-profit beer). This technique is often referred to as *Market Basket Analysis*. Thus, the data mining system is used to discover the hidden relationships between products purchased from a large amount of unstructured existing data.

Knowledge discovery as a process consists of an iterative sequence of the following steps [33]:

1. Data Cleaning (to remove noise and inconsistent data.)
2. Data integration (where multiple data sources may be combined.)
3. Data selection (where data relevant to the analysis task are retrieved from the database.)
4. Data transformation (where data are transformed or consolidated into forms appropriate for mining by performing summary or aggregation operations, for instance.)
5. Data mining (an essential process where intelligent methods are applied in order to extract data patterns.)
6. Pattern evaluation (to identify the truly interesting patterns representing knowledge based on some interestingness measures.)
7. Knowledge presentation (where visualization and knowledge representation techniques are used to present the mined knowledge to the user.)

The most common tasks that data mining is usually used to accomplish are listed as follows [34]:

- Description (to find ways to describe patterns and trends lying within data.)
- Classification (to examine records containing information on a target categorial variable, which could be partitioned into several classes or categories, and create the training set. Based on the classifications in the training set, classifications are assigned to the new records.)

- Estimation (which is similar to classification except that the target variable is numerical rather that categorical.)
- Prediction (which is similar to classification and estimation except that for prediction, the results lie in the future.)
- Clustering (to group records, observations, or cases into clusters, which are collections of records that are similar to one another and dissimilar to records in other clusters.)
- Association (to find which attributes "go together." Association rules are of the form "If antecedent, then consequent," together with a measure of the support and confidence associated with the rule, as described in 1.4.)

This research focuses on the association task. The marketing knowledge in this Web Marketing Project is mainly represented in the form of association rules. The relationship between antecedent and consequent implies connections from people in one group to people in another group. One group can be either a demographic group or an interest group.

3.3 Description of Data and Mining

The data used for data mining consist of records of real personal data that contain demographic information and expressed interests of each person. In most cases, triples of age, gender and one interest are used as input for data mining.

The interests are stored in an ontology and organized as a DAG (Directed Acyclic Graph) as described in Chapter 2. Figure 3.1 shows a tiny part of this DAG. Every line in this diagram stands for an IS-A link. An IS-A link connects two interest concepts. The interest concept that is higher up is more general than the interest concept that is lower in the diagram. Thus, for example, the IS-A link from INTERNET to COMPUTERS_INTERNET indicates that INTERNET is a more specific interest than COMPUTERS_INTERNET.

Interests are derived from one of sixteen top level interest categories of Yahoo. These interest categories are called interests at level 1. Examples of level 1 interests include

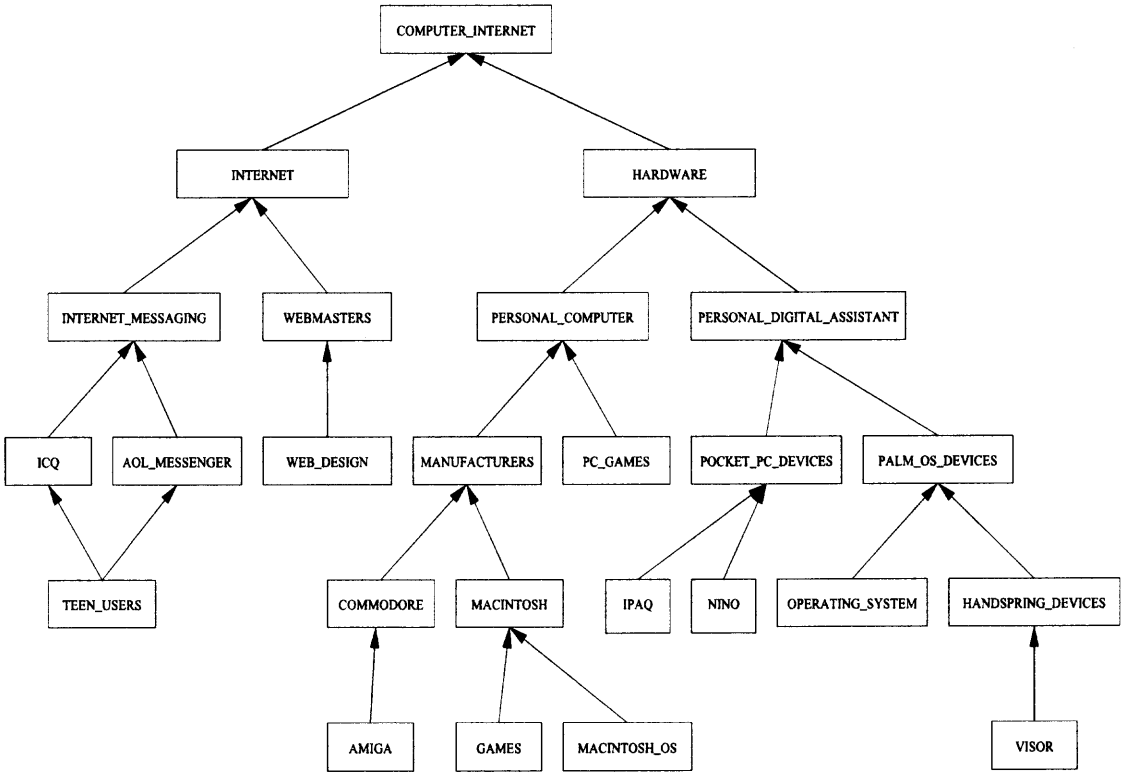


Figure 3.1 Subnetwork of the COMPUTER_INTERNET interest hierarchy.

COMPUTERS_INTERNET (Figure 3.1), RECREATION_SPORTS, HEALTH_WELLNESS, GOVERNMENT_POLITICS, FAMILY_HOME, *etc.* Other interests are descendants of these interest categories. Interests are often very specific. Thus, there are interests such as LOPEZ_JENNIFER, GRANT_AMY and MARTIN_RICKY in the hierarchy, who are all singers. The implemented version of the interest hierarchy contains over 31,000 interests. The current version of the hierarchy at Yahoo is an order of magnitude larger.

The database in the project contains about 300,000 real people and well over one million individual interests associated with these people. As a result of the large size of the database, the available data go beyond the capacity of the data mining program. Thus, the data had to be broken into smaller data sets. A convenient way to do this is to perform data mining on tuples containing the interest categories at level 1 (top level) or on the descendants of level 1 interests.

The innovation in this research is the use of a combined raising–mining operation. Thus, tuples of interest data and demographic data from the database are first raised to higher levels in the ontology. That means that generalizations of each tuple are created at multiple levels in the ontology. Then WEKA, which generates association rules [35] using the Apriori algorithm, first presented by [4], is applied separately at every level of the ontology. This research refers to the process of raising data to all possible higher levels as *exhaustive raising*. In Section 3.5 the theory of raising is discussed. Exhaustive raising followed by the repeated application of WEKA is very time intensive. Thus, in Section 3.8, possibilities are explored to avoid exhaustive raising by predicting the optimal level to which to raise tuples.

3.4 Previous Work on Combining Ontologies with Rule Mining

A concept hierarchy is present in many databases either explicitly or implicitly. Some previous work utilizes a hierarchy for data mining. Han [36] discusses data mining at multiple concept levels. His approach is to use associations at one level (e.g., milk $\rightarrow$ bread) to direct the search for associations at a different level (e.g., milk of brand X $\rightarrow$ bread of brand Y). As most of the data mining involves only one interest, the problem setting in this research is quite different. Han *et al.* [37] introduce a top-down progressive deepening method for mining multiple-level association rules. They utilize the hierarchy to collect large item sets at different concept levels. The approach in this research utilizes an interest ontology to improve support in rule mining by means of concept raising. To the best of our knowledge, the combination of ontologies with association rule mining for the purpose of finding generalized rules with high support from sparse data has not appeared in the literature before [38].

Fortin *et al.* [39] use an object-oriented representation for data mining. Their interest is in deriving multi-level association rules. As only one data item in each tuple is typically used for raising, the possibility of multi-level rules does not arise in our problem setting.

Srikant *et al.* [40] present Cumulative and EstMerge algorithms to find associations between items at any level by adding all ancestors of each item to the transaction. In this research, items of different levels do not coexist in any step of mining. Psaila *et al.* [41] describe a method how to improve association rule mining by using a generalization hierarchy. Páircéir *et al.* [42] also differ from the work of this research in that they are mining multi-level rules that associate items spanning several levels of a concept hierarchy. Data mining has been viewed as an operation with a query language in [43,44]. Joshi *et al.* [45] are interested in situations where rare instances are really the most interesting ones, e.g., in intrusion detection. They present a two-phase data mining method with a good balance of precision and recall. For this research, rare instances are not by themselves a focus, they are only important because they contribute to the low support values that this research is attacking.

Zaki and Hsiao [46] present a method that greatly reduces the number of redundant rules generated by previous rule miners. They define closed frequent item sets, which are sufficient for rule generation, to replace traditional frequent item sets. They show that this may lead to a reduction of the frequent item sets by two orders of magnitude, for a given support value. The concern is not with the efficiency of generating association rules, but with the total support of the resulting rules. However, any rule mining algorithm may be plugged into the Web marketing Project, as mining and raising are performed in a modular way. Thus, the Web Marketing Project would benefit from the improved efficiency of a data mining algorithm such as CHARM [46]. Mannila *et al.* [47] worked on improving algorithms for finding associations rules, by eliminating unnecessary candidate rules.

Lu *et al.* [48] divide their data sets into partitions that correspond to different time intervals. This helps them construct association rules that take "fashionable" items more strongly into account. The influence of outdated items on the generated association rules is reduced, resulting in rules with better predictive power. The focus of this research is

not on the age of transactions but on the position of data items in a specificity – generality dimension defined by an ontology.

Bruha and Tkadlec [49] attach rule quality values to rules and define methods for combining rule qualities. When several rules fire on a given input, rule quality can be used to decide which rule should have precedence. As opposed to their approach, our work does not concentrate on selecting one of several existing rules, but rather on generating high support rules.

Berzal *et al.* [50] have worked on a problem that is in some sense the diametrical opposite of the problem in this research. They are trying to eliminate misleading rules which are the result of too high support values. The problem of generating association rules when the available support is too low to derive practically useful rules is being addressed here.

This work is similar to [51] in that it incorporates prior knowledge into the rule mining process. Like Zhou *et al.* a directed acyclic graph structure is used to present such additional knowledge. However, this research is not using the numeric (probabilistic) dependencies of [51].

[52] have combined ontologies with association rules, but in a completely different way than what in this research. Their purpose is to semi-automatically construct ontologies. They are using an association rule miner in the service of this activity.

3.5 Using Raising for Improved Support

3.5.1 Formal Definition of Raising

A formal definition of **raising a tuple to its parents** is now presented. Below an alternative called **raising a tuple to a level** is shown.

Definition 1: An **operation** R, called raising a tuple to its parents. Given is a data tuple $T =< N, D >$ where N is derived from a rooted ontology O. In O, N has a uniquely

determined, ordered sequence of parents $< P_1, P_2, .., P_m >$. Then R is defined as the operation that takes T as input and returns the sequence of tuples

$$T^R = R(T) =<< P_1, D >, < P_2, D >, .. < P_m, D >> \tag{3.1}$$

as output for every T in O, except for the tuple <Root(O), D >. For the latter $R(T)$ is undefined.

In our case, N is an interest from an interest ontology O. D stands for one or several items of demographic information. For example, if the given tuple T says that one male (M) in the age range 20-24 is interested in Jennifer Lopez:

$$T =< \text{LOPEZ\_JENNIFER}, \text{M}, 20-24 > \tag{3.2}$$

and the interest LOPEZ_JENNIFER has two parents, ACTRESS and SINGER, then the result of raising is:

$$R(T) =<< \text{ACTRESS}, \text{M}, 20-24 >, < \text{SINGER}, \text{M}, 20-24 >> \tag{3.3}$$

meaning that one person of male gender in the age group 20-24 is interested in an actress and singer. One issue arising at this point is that the specific person might be interested in Jennifer Lopez *only* as an actress or *only* as a singer. Thus, this generalization step introduces a certain loss of information. However, this is not a critical issue, for three reasons: (1) Many interests have a single parent. (2) Even if there is an initial up-branching, there will be a join point higher up in the hierarchy. For example, when moving up in the interest hierarchy, ACTRESS and SINGER will be found to be descendants of ARTIST. (3) From a practical marketing perspective, which is the context of this work, a company that is marketing Jennifer Lopez is most likely to market her as actress and as singer, trying to

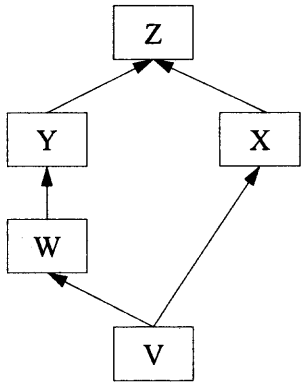


Figure 3.2 Example of parents at different levels.

sell both movie DVDs and music CDs. Thus, even somebody interested only in the singer Jennifer Lopez would be a target for marketing.

After having raised the data item $< N, D >$, any traditional association rule mining algorithm may be applied to the result of raising

$$<< P_1, D >, < P_2, D >, .. < P_m, D >> \tag{3.4}$$

instead of applying it to the original data item $< N, D >$. Thus, raising replaces a data item by "parent items" before performing rule mining.

For convenience, it is assumed that every interest in the hierarchy is assigned a level L by a breadth-first search. Then the level function L(T) is defined to return this level as a number. Interests nearer to the root have lower level numbers. The root is by definition at level 0 (see Section 2.3 for a BFS algorithm.)

There is one complication with the above definition of raising. In a DAG, a concept often has parents at several different levels. For example, a problematic DAG is shown in Figure 3.2.

If level numbers are assigned to this DAG in a top-down, breadth-first fashion, then Z will be at level 0. Z's children Y and X will be at level 1. W is a child of X and will therefore be at level 2. V is a child of W and therefore will be at level 3. However, V

has two parents, X and W. As X is at level 1 and W is at level 2, V has two parents at different levels. When raising only to parents in such a DAG, the root would be reached after two steps when following the path through X and after three steps when following the path through W. This is not a clean approach to raising. (Note that by doing a bottom-up numbering of levels, X would end up at level 2 instead of level 1. But now Z has children at different levels, which is not an improvement.)

This problem was overcome by doing the following. Instead of raising a tuple to all its parents, it is preferable to raise a tuple *to a specific level.* The advantage of this approach is that it cleanly generalizes to ancestors of a tuple.

Thus, the formal definition of **raising a tuple to a level** k (as opposed to raising it to its parents) is presented now.

Definition 2: An **operation** R^k, called raising a tuple to the level k. Given is a data tuple $T =< N, D >$ where N is derived from a rooted ontology O. In O, N has a uniquely determined ordered sequence of ancestors $< A_1^k, A_2^k, .., A_m^k >$, all at level k (counted from the root). Then R^k is defined as the operation that takes T as input and returns the sequence of tuples

$$T^k = R^k(T) =<< A_1^k, D >, < A_2^k, D >, .. < A_m^k, D >> \tag{3.5}$$

as output for every T in O, such that L(T) $> k$. N in T is at a level with a number greater than k.

The last condition in the definition above says that you can only raise a tuple to a higher level in the DAG diagram if it is initially below that level. By the definition of level numbers, the root is at level 0. Lower levels in the diagram correspond to higher level numbers.

Because N has m ancestors at level k, the result of raising T, namely $R^k(T)$, is a sequence of m new tuples, one for each ancestor at level k. In the previous example

(Figure 3.2),

$R^2(< V, D >) =<< W, D >>$, but $R^1(< V, D >) =<< Y, D >, < X, D >>$.

After raising, association rule mining is applied to all tuples at level k. This includes the raised tuples and the tuples with interests that were originally at level k.

It is important to note that even when raising a tuple from a level k to a level l, the raising operation is *not* being applied $k - l$ times. It is *not* raising to $k - 1$ first, $k - 2$ next, *etc.* Rather, it is directly raising to the target level l. The significance of this distinction will become clear in Section 3.6.

3.5.2 Applications of Raising

In the version of the Yahoo interest hierarchy used for this research, there are 11 levels of interests. For example, FAMILY_HOME is an interest at level 1. PARENTING is an interest at level 2. PARENTING is a child of FAMILY_HOME in the hierarchy. If a person expressed an interest in PARENTING, it is common sense that he or she is interested in FAMILY_HOME. Therefore, at level 1, when those people who have expressed an interest in FAMILY_HOME are counted, it is reasonable to count those who are interested in PARENTING also. This idea applies in the same way to lower levels.

A big problem in the derivation of association rules is that available data are sometimes very sparse and biased as a result of the interest hierarchy. For example, among over a million of interest records in the database of this research only 11 people expressed an interest in RECREATION_SPORTS, and nobody expressed an interest in SCIENCE. The fact that people did not express interests with those general terms does not mean they are really not interested in them. The data file of RECREATION_SPORTS has 62,734 data items. In other words, 62,734 interest expressions of individuals are in the category of (descendants of) RECREATION_SPORTS. Instead of saying "I'm interested in Recreation and Sports," people prefer saying "I'm interested in basketball and fishing." They tend to be more specific with their interests. After analyzing the 16 top level categories of the interest

hierarchy, it was found that users expressed interests at the top level only in two categories, MUSIC and RECREATION_SPORTS. When mining data at higher levels, it is important to include data from lower levels, in order to gain accurate rules and higher support.

The fact that there is an interest hierarchy at our disposal makes it easy to add to sparse data, involving interests near the root of the interest hierarchy, additional interests from lower down in the hierarchy. A greatly simplified example is now shown to clarify this process:

In the following examples, the < and > symbols of tuples are dropped. Tuples are shown as left-justified lines. In each such line an interest appears first. The first letter after the interest stands for an age range. The age range from 10 to 19 is represented by A, 20 to 29 is B, 30 to 39 is C, 40 to 49 is D, *etc.* This encoding was used, because WEKA did not allow number ranges to be represented directly. The second letter stands for the gender, Male (M) or Female (F). Text after a double slash (//) is not part of the data. It contains explanatory remarks.

INPUT:
Original Data File:
COUNTRY,D,F //level=3
JAZZ,D,F //level=3
GRANT_AMY,A,M //level=6
MARTIN_RICKY,B,F //level=5
PIANO,A,F //level=4
COUNTRY_LYRICS,A,M //level=4

The levels below 6 do not have any data in this example. Raising processes the data level-by-level starting at level 1. It is easiest to see what happens if the processing of level 3 is studied.

First the result is initialized with the data at level 3 contained in the source file. With the data shown above, that means that the result at level 3 is initialized with the following lines:

RESULT (initial):
COUNTRY,D,F
JAZZ,D,F

In order to perform raising to level 3, there is a need to find ancestors at level 3 of the interests located at levels 4, 5, *etc.* Table 3.1 shows all ancestors of the interests from levels 4, 5, 6, such that the ancestors are at level 3. Thus, during raising, the tuple <GRANT_AMY, A, M> at level 6 is replaced by the tuple <CHRISTIAN, A, M> at level 3. The tuple <MARTIN_RICKY, B, F> at level 6 is replaced by <LATIN, B, F> at level 3. Similarly, PIANO is mapped into KEYBOARD_INSTRUMENTS and COUNTRY_LYRICS into COUNTRY. Thus, the following lines are now added to the initial result at level 3.

CHRISTIAN,A,M // raised from level=6
LATIN,B,F // raised from level=5
KEYBOARD_INSTRUMENTS,A,F // raised from level=4
COUNTRY,A,M // raised from level=4

Thus the final result is given:

RESULT (final):
COUNTRY,D,F
JAZZ,D,F
CHRISTIAN,A,M
LATIN,B,F
KEYBOARD_INSTRUMENTS,A,F
COUNTRY,A,M

The initial data at level 3 contained tuples with JAZZ and COUNTRY. Raising creates four new tuples at level 3. However, the interest COUNTRY already occurred at level 3. Thus, there are now two tuples with COUNTRY at level 3. That means, after raising there are tuples with the following interests at level 3:

COUNTRY: 2 tuples
CHRISTIAN: 1 tuple
LATIN: 1 tuple
KEYBOARD_INSTRUMENTS: 1 tuple
JAZZ: 1 tuple

Before raising, there were only two items at level 3. Now, there are six items at level 3. That means that there are now more data as input for data mining than those before raising. In general, after raising, the results of data mining will have better support for many rules.

Table 3.1 Ancestor table for a Raising example

Interest Name	Its ancestor(s)
GRANT_AMY	CHRISTIAN
MARTIN_RICKY	LATIN
PIANO	KEYBOARD_INSTRUMENTS
COUNTRY_LYRICS	COUNTRY

3.6 Correctness Issues when Raising to a Level

Due to the existence of multiple parents and common ancestors, the details of the method of raising are very important. As pointed out previously, the method directly raises to a target level, without intermediate steps. Otherwise spurious tuples could be introduced, as the following example shows.

Figure 3.1 demonstrates a situation with two parents and a common ancestor. TEEN_USERS is an interest at level 5. It has two parents at level 4: ICQ and AOL_MESSENGER. If somebody is interested in TEEN_USERS, it is reasonable to say that he or she is also interested in ICQ and AOL_MESSENGER. The lowest common ancestor of ICQ and AOL_MESSENGER is INSTANTMESSAGING at level 3.

If the tuple is raised

$$T =< \text{TEEN\_USERS}, \text{M}, \text{A} > \tag{3.6}$$

in two steps to the ancestor INTERNET_MESSAGING, the following would happen. It would be first raised to the parent level resulting in

$$R(T) =<< \text{ICQ}, \text{M}, \text{A} >, < \text{AOL\_MESSENGER}, \text{M}, \text{A} >> \tag{3.7}$$

If it is then raised again to the (next) parent level, the result is:

$$T =<< \text{INTERNET\_MESSAGING}, \text{M}, \text{A} >< \text{INTERNET\_MESSAGING}, \text{M}, \text{A} >> \tag{3.8}$$

Thus, a spurious tuple has been introduced. Instead of only one person who is interested in INTERNET_MESSAGING, it appears that there are two. However, if the tuple T is directly raised to level 3, i.e., to the target ancestor, then it would appear only once. This method solves the problem of spurious tuples caused by multiple parents and common ancestors. Note that it would *not* be a solution to simply eliminate duplicates at the target, because in most cases there will be legitimate duplicates. The whole purpose of raising is to provide large numbers of identical tuples at higher levels for the purpose of mining them.

3.7 Results of Exhaustive Raising

The results of an experiment with *exhaustive raising* is now analyzed. Exhaustive raising raises every tuple to every level above its initial location. In Section 3.8, a method will be discussed a method of avoiding exhaustive raising, by raising to an "optimal" level.

The quality of association rules is normally measured by specifying support and confidence. Support may be given in two different ways [3], as absolute support and as relative support, as described in Section 1.4. Witten *et al.* write:

> The coverage of an association rule is the number of instances for which it predicts correctly – this is often called its support. ... It may also be convenient to specify coverage as a percentage of the total number of instances instead. (p. 64)

For this purposes, this research is most interested in the total number of tuples that can be used for deriving association rules, thus only the absolute version of support is used. The data support is substantially improved by means of raising. Below, rule (1) is from RECREATION_SPORTS at level 2 without raising:

age=C interest=OUTDOORS 370 $\Rightarrow$ gender=M 228 conf:(0.62) (1)

Among 370 people in age group C who are interested in OUTDOORS, 228 are male, resulting in an absolute support of 228 and a confidence of 228/370 = 0.62. Following, rule (2) is from RECREATION_SPORTS at level 2 with raising.

age=C interest=OUTDOORS 8284 ⇒ gender=M 5598 conf:(0.68) (2)

Rule (1) and Rule (2) have the same attributes and rule structure. Without raising, the absolute support is 228, while with raising it becomes 5598. The improvement of the absolute support of this rule is 2355%.

Another rule obtained before raising is:

age=B interest=AVIATION 70 ⇒ gender=M 55 conf:(0.79) (3)

In the data, 70 people are of age category B and express AVIATION as their interest. Among them, 55 are male. This results in rule (3). The confidence for rule (3) is 0.79. Raising affects the results of the WEKA algorithm. For example, rule (3) did not appear in the rules returned by WEKA after raising.

There is a combination of two factors why rules may disappear after raising. WEKA ranks the rules according to confidence and support, and discards rules with lower confidence even though the support may be higher. By default, WEKA limits the number of returned rules by a cut-off parameter (e.g. 100). After raising, WEKA returns no rule about AVIATION, because its support is too small compared with other interests such as SPORTS and OUTDOORS. The rule about AVIATION gets bumped out of the top 100 rules. In other words, one effect of raising is that rules that appear in the result of WEKA before raising might not appear after raising and vice versa. In this concrete example, the following rule (4) that was not returned before raising appeared among the top 100 rules after raising:

age=A gender=F 13773 ⇒ interest=SPORTS 10834 conf:(0.79) (4)

It is of interest to understand why raising causes such dramatic relative shifts in support values. This can be analyzed for the example of rule (3) disappearing from the

result set, and rule (4) appearing in its stead. Although the Yahoo ontology ranks both AVIATION and SPORTS as level 2 interests, the hierarchy structure underneath them is not balanced. According to the hierarchy, AVIATION has 21 descendents, while SPORTS has 2120 descendents, which is about 100 times more. The longest path to a leaf from level 1 that goes through AVIATION has only 5 nodes. The longest comparable path of SPORTS has 11 nodes. Following are the longest paths through AVIATION and SPORTS:

1. RECREATION_SPORTS (Level=1) ⟶
AVIATION (Level=2) ⟶
MILITARY (Level=3) ⟶
PLANES (Level=4) ⟶
B-52 STRATOFORTRESS (Level=5)

2. RECREATION_SPORTS (Level=1) ⟶
SPORTS (Level=2) ⟶
SOCCER (Level=3) ⟶
BY REGION (Level=4) ⟶
COUNTRIES (Level=5) ⟶
UNITED KINGDOM (Level=6) ⟶
LEAGUES (Level=7) ⟶
FA CARLING PREMIERSHIP (Level=8) ⟶
CLUBS (Level=9) ⟶
LIVERPOOL FC (Level=10) ⟶
OWEN, MICHAEL (Level=11)

Raising tuples to level 2 means that all interests below level 2 are replaced by their ancestors at level 2. There are many tuples with interests from the subtree under SPORTS. Even if it is naively assumed that there is only one tuple per node in the two subtrees under SPORTS and AVIATION, there would be 100 times more tuples about SPORTS than about AVIATION. As a result, SPORTS becomes an interest with overwhelmingly high support, whereas the improvement caused by raising to AVIATION is so small that rule (3) disappears from the returned rules.

The appearance of rule (4) is considered as an additional benefit of raising. After raising, 13773 people are in the age category A and gender category F. Among them, 10834 are interested in SPORTS. The confidence is 0.79. These data allow a high confidence and high support rule to be generated. However, there was no rule about SPORTS in the result set before raising. Intuitively, one would prefer getting a rule about SPORTS over getting a

rule about AVIATION. In other words, a rule has been uncovered which is with very strong support that also agrees well with the intuition. Thus, this example shows that raising can give preference to rules that agree well with the intuition.

To evaluate this method, the support and confidence of raised and unraised rules have been compared. The improvement of absolute support is substantial. Table 3.2 compares support and confidence for the same rules before and after raising for RECREATION_SPORTS at level 2. There are 58 3-attribute rules without raising, and 55 3-attribute rules with raising. 18 rules are the same in both results. Their support and confidence are compared in the table. The average support is 170 before raising, and 4,527 after raising. The average improvement is 2898%. Thus, there is a substantial improvement in absolute support.

The picture looks different for confidence. After raising, the lower average confidence in Table 3.2 is a result of expanded data for the following reasons. Raising effects not only the data that contribute to a rule, but all other data as well. Thus, relevant and irrelevant tuples are added to the input data for mining, and the confidence of a rule in the output might either drop or increase. Even in the cases where confidence becomes lower, the improvement in support by far outpaces the drop in confidence.

Table 3.3 shows the comparison of all rules that are the same before and after raising. The average improvement of support is calculated at level 2, level 3, level 4 and level 5 for each of the 16 categories. As explained in Section 3.5, few people expressed an interest at level 1, because these interest names are too general. Before raising, there are only 11 level 1 tuples with the interest RECREATION_SPORTS and 278 tuples with the interest MUSIC. In the other 14 categories, there are no tuples at level 1 at all. However, after raising, there are 6,119 to 174,916 tuples at level 1, because each valid interest in the original data can be represented by its ancestor at level 1, no matter how low the interest is in the hierarchy.

All the 16 categories have data down to level 6. However, COMPUTERS_INTERNET, FAMILY_HOME and HEALTH_WELLNESS have no data at level 7. In general, data

Table 3.2 Support and confidence before and after Raising

Rule (int = interest, gen = gender)	Supp. w/o rais.	Supp. w/ rais.	Improv. of supp.	Conf. w/o rais. %	Conf. w/ rais. %	Improv. of Conf.
age=C int=AUTOMOTIVE ⇒ gen=M	57	3183	5484%	80	73	-7%
age=B int=AUTOMOTIVE ⇒ gen=M	124	4140	3238%	73	65	-8%
age=C int=OUTDOORS ⇒ gen=M	228	5598	2355%	62	68	6%
age=D int=OUTDOORS ⇒ gen=M	100	3274	3174%	58	67	9%
age=B int=OUTDOORS ⇒ gen=M	242	5792	2293%	54	61	7%
age=C gen=M ⇒ int=OUTDOORS	228	5598	2355%	51	23	-28%
gen=M int=AUTOMOTIVE ⇒ age=B	124	4140	3238%	47	37	-10%
age=D gen=M ⇒ int=OUTDOORS	100	3274	3174%	46	27	-19%
age=B int=OUTDOORS ⇒ gen=F	205	3660	1685%	46	39	-7%
age=B gen=M ⇒ int=OUTDOORS	242	5792	2293%	44	18	-26%
gen=F int=OUTDOORS ⇒ age=B	205	3660	1685%	42	39	-3%
gen=M int=OUTDOORS ⇒ age=B	242	5792	2293%	38	34	-4%
int=AUTOMOTIVE ⇒ age=B gen=M	124	4140	3238%	35	25	-10%
gen=M int=OUTDOORS ⇒ age=C	228	5598	2355%	35	33	-2%
age=D ⇒ gen=M int=OUTDOORS	100	3274	3174%	29	19	-10%
gen=M int=AUTOMOTIVE ⇒ age=C	57	3183	5484%	22	28	6%
int=OUTDOORS ⇒ age=B gen=M	242	5792	2293%	21	22	1%
int=OUTDOORS ⇒ age=C gen=M	228	5598	2355%	20	21	1%

below level 6 are very sparse and do not contribute a great deal to the results. Therefore, in Table 3.3, it is presented the comparison of rules above level 5 only.

Sometimes the same rules are generated by WEKA with and without raising. In other cases, there is not a single rule in common between the rule sets with and without raising. The comparison is therefore not applicable. Those conditions are denoted by "N/A" in Table 3.3. For all other cases the *improvement rate* r *for a single rule* is computed as follows. s_b is the support of a rule before raising. s_a is the support of a rule after raising.

$$r = \frac{s_a - s_b}{s_b} * 100[percent] \tag{3.9}$$

Because there are no tuples at all at level 1 for most categories, there would be a 0 in the denominator. Thus, the improvement rate at level 1 is not computed.

Table 3.4 shows the average improvement of support of all rules after raising to level 2, level 3, level 4 and level 5 within the 16 interest categories. This is computed as follows.

Table 3.3 Support improvement rate of common rules

Category	Level 2	Level 3	Level 4	Level 5
BUSINESS_FINANCE	122%	284%	0%	409%
COMPUTERS_INTERNET	363%	121%	11%	0%
CULTURES_COMMUNITY	N/A	439%	N/A	435%
ENTERTAINMENT_ARTS	N/A	N/A	N/A	N/A
FAMILY_HOME	148%	33%	0%	0%
GAMES	488%	N/A	108%	0%
GOVERNMENT_POLITICS	333%	586%	0%	N/A
HEALTH_WELLNESS	472%	275%	100%	277%
HOBBIES_CRAFTS	N/A	0%	0%	0%
MUSIC	N/A	2852%	N/A	0%
RECREATION_SPORTS	2898%	N/A	76%	N/A
REGIONAL	6196%	123%	N/A	0%
RELIGION_BELIEFS	270%	88%	634%	0%
ROMANCE_RELATIONSHIPS	224%	246%	N/A	17%
SCHOOLS_EDUCATION	295%	578%	N/A	297%
SCIENCE	1231%	0%	111%	284%
Average Improvement	1086%	432%	104%	132%

The support values for all rules before raising are summed and then the sum is divided by the number of rules, i.e., the average support of a rule before raising, S_b, is computed. This computation is done for the given level and interest category. Similarly, the average support of all the rules after raising is computed. Then the *improvement rate ρ for a set of rules* is computed like before:

$$\rho = \frac{S_a - S_b}{S_b} * 100[percent] \tag{3.10}$$

The average improvement rate over all interest categories for level 2 through level 5 is, respectively, 279%, 152%, 68% and 20%. In Table 3.4 there are three values where the improvement rate ρ is negative. This may happen if the average support becomes lower after raising. That in turn can happen, because, as mentioned before, the rules before and after raising may be *different* rules.

Generality of Raising: Note that even though this study is based on one specific interest hierarchy and a specific limited set of demographic features, there is nothing in the method that is domain specific. The only formal condition was that exactly one item of each tuple is

Table 3.4 Support improvement rate of all rules

Category	Level 2	Level 3	Level 4	Level 5
BUSINESS_FINANCE	231%	574%	-26%	228%
COMPUTERS_INTERNET	361%	195%	74%	-59%
CULTURES_COMMUNITY	1751%	444%	254%	798%
ENTERTAINMENT_ARTS	4471%	2438%	1101%	332%
FAMILY_HOME	77%	26%	56%	57%
GAMES	551%	1057%	188%	208%
GOVERNMENT_POLITICS	622%	495%	167%	1400%
HEALTH_WELLNESS	526%	383%	515%	229%
HOBBIES_CRAFTS	13266%	2%	7%	60%
MUSIC	13576%	3514%	97%	62%
RECREATION_SPORTS	6717%	314%	85%	222%
REGIONAL	7484%	170%	242%	-50%
RELIGION_BELIEFS	285%	86%	627%	383%
ROMANCE_RELATIONSHIPS	173%	145%	2861%	87%
SCHOOLS_EDUCATION	225%	550%	1925%	156%
SCIENCE	890%	925%	302%	317%
Average Improvement	279%	152%	68%	20%

derived from a DAG-shaped hierarchy. Tree-shaped hierarchies, as special cases of DAGs, are also acceptable. By using only one item from the hierarchy in each tuple, it becomes possible to assign a position to each tuple in the hierarchy. The hierarchy itself defines levels and the direction of raising.

Thus, in marketing, it is easy to switch from an interest hierarchy to a product hierarchy. Many additional demographic features may be added for mining purposes, such as zip codes, household income, number of children, marital status, *etc.*

As an example area outside of marketing where the Raising method could be used, it is suggested to use the well-known animal taxonomy [53]. Animals have been organized into kingdoms, phyla, subphyla, superclasses, classes, infraclasses, orders and families. The bottom two levels in this hierarchy are called genus and species. Tuples could include species names, geographical areas of occurrence and annual reductions in head-counts. This would be especially interesting for endangered species. Raising would allow people to generate rules where locations and reductions in head-counts are associated with families and orders of animals.

3.8 Avoiding Exhaustive Raising: What Level to Raise To

In Section 3.7, the effects of raising every tuple to every level and generating rules at each level were considered. This is called exhaustive raising. Exhaustive raising is computationally very expensive. Thus, instead of raising all data to every level, it would be better if how far up tuples should be raised is known ahead of time to produce good association rules. On one hand, the farther up tuples are raised, the better the support will be. Thus, it appears that one should go all the way up to level 1. On the other hand, at level 1, there are only a few (16) interest choices. Thus, there are only a few potential rules. Effective marketing cannot be performed in a targeted way if only rules about a few very general interests are formulated. For 16 top level interests, only (on the order of) 16 rules could be generated. The "on the order of" appears as one interest may occur in combination with several (age, gender) pairs. Furthermore, it is doubtful whether very general rules are useful for marketing. For example, it is presumed that almost everybody has an interest in some aspects of the level 1 interest category ENTERTAINMENT_ARTS. That does not mean to market opera tickets, impressionist paintings and heavy metal music CDs to the same people. The goal is to market each of these three categories of items separately to the most promising demographic group.

In Section 3.7, at the bottom of the interest hierarchy rules were derived with too little support. As has been argued in this section, at the top there are too few potential rules. As a second disadvantage, rules near the root are, in all likelihood, too general to be of practical use. The question that will be investigated in the balance of this chapter is whether there is an optimal target level for raising. If tuples raised to this level are mined for association rules, these rules should have high enough support to represent a substantial demographic group and still be specific enough to allow targeted marketing activities. It is hypothesized that the optimal level to which to raise should be somewhere in the middle of the ontology. As data are sparse below level 6, it is hypothesized that the optimal level would be 3 or 4.

So, the question then is to find this optimal raising level. This question is of practical importance. In the previous section, all data was raised to every level. This is a time-consuming process which results in a large number of rules, many of which the user will probably not be interested in. Thus, if it is known to which level to raise the data, the mining can be performed much faster and rules that are more likely to be of interest to users can be produced.

The approach chosen is to maximize the total support of all rules generated in the mining process. Thus, at low levels there are many rules with low support, while near the root there are a few rules with high support. What is desired, ideally, are many rules with high support. Thus, the result of a combined raising–mining operation may be characterized by the sum of the support values S_l of all the rules at a specific level l. If this sum is high, then the combined operation of raising to level l, followed by mining, was successful.

To analyze the sum of support distribution, a graph was constructed that shows S_l^j, the sum of support values for all rules at level l over the level number, separately for each one of the 16 main interest categories j. It was expected that this graph would have one peak somewhere in the middle for each interest category j and that all the peaks would be more or less at the same level. Figure 3.3 shows the 16 graphs separately, one for each of the 16 interest categories at level 1. It turns out that, indeed, the graph for every interest has a global maximum. However, for several interests there is at least one other local maximum. Also, the graphs are more skewed towards the left, towards the low level numbers (i.e., towards the top of the ontology), than what was expected.

As the next step, an optimal raising level is computed, which incorporates all the information available from the 16 level-1 interest categories. While the optimal raising level of other interest hierarchies might be different, the Yahoo hierarchy is fairly large (over 31,000 interests) and diverse. Thus, there is some value in finding such an optimal raising level.

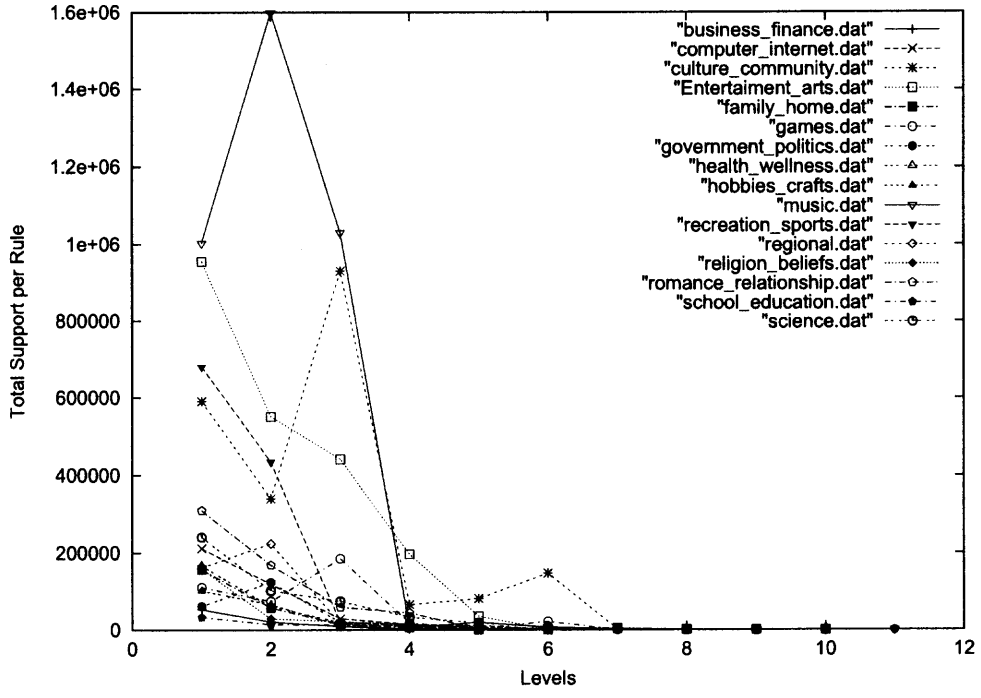


Figure 3.3 Graph for sums of support of rules over levels for 16 interest categories.

For the actual computation, two approaches were used. In both approaches, a weighted average of the 16 support sum distributions is computed.

(1) In the first approach, the *peak approach*, every one of the 16 interest graphs is represented by one single point, namely its global maximum. Looking at Figure 3.3, it becomes clear why a weighted average is needed. While it is reasonable to just average the level numbers of the 16 peaks, this would not reflect the reality that the peak of the category MUSIC is much higher than all the other peaks. Thus, there is a need to give more prominence to it. Therefore, all the level numbers are weighted by the maximum value of the support sum distribution for each category (the height). Thus, it is computed as follows:

$$L_{peak}^{opt} = \frac{\sum_{j=1}^{16} S_{peak}^{j} * L_{peak}^{j}}{\sum_{j=1}^{16} S_{peak}^{j}} \tag{3.11}$$

L^{opt}_{peak} is the optimal level to raise to, computed with peaks.

S^{j}_{peak} is the maximum support sum of the j^{th} interest category distribution.

L^{j}_{peak} is the level at which the support sum becomes maximal for category j.

The final result of the peak approach is:

$$L^{opt}_{peak} = \frac{53593 * 1 + 211991 * 2 + 930802 * 3 + 124100 * 2 + ..}{53593 + 211991 + 930802 + 124100 + ..} = 1.65$$

The peak approach is simplistic in that it represents every category distribution by its global maximum. Due to the existence of local maxima, a second approach is used.

(2) In the second approach the overall weighted average formula for the interest categories is still used. However, now each category itself is represented by a weighted average of its graph. In other words, the weighted average formula has been applied twice, once to each graph (interest category) individually, and once to the resulting numbers. Thus, this is called the *double average approach.*

First a representative level value X_j for the j^{th} interest category is computed. It is the weighted average of the interest category j, computed as

$$X_j = \frac{\sum_{i=1}^{n} i * S^{j}_{i}}{\sum_{i=1}^{n} S^{j}_{i}} \tag{3.12}$$

S^{j}_{i} is the support sum of all rules raised to level i, belonging to category j.

X_j corresponds to an x-coordinate in Figure 3.3.

For the y-coordinate the sums of support values at different levels i is summed, giving Y_j as follows:

$$Y_j = \sum_{i=1}^{n} S^{j}_{i} \tag{3.13}$$

S_i^j is the sum of the supports of all rules at level i, for category j.

n is the highest level for the interest category j for which the support sum is non-zero.

n is used instead of 11 (levels), because for some higher level values the sum of supports is 0. Now X_j and Y_j are used to represent the j^{th} distribution in a way which is more meaningful than representing it just by its peak. Thus X_j and Y_j can be plugged into the original weighted average formula.

$$L_{average}^{opt} = \frac{\sum_{j=1}^{16} Y_j * X_j}{\sum_{j=1}^{16} Y_j} \tag{3.14}$$

$L_{average}^{opt}$ is the optimal level computed by applying the weighted average to X_j, Y_j,

j represents the interest category.

As the final result for the double average approach there is:

$$L_{average}^{opt} = \frac{1.63 * 88075 + 1.631 * 379617 + 2.61 * 2158829 + ..}{88075 + 379617 + 2158829 + ..} = 1.95 \tag{3.15}$$

where 1.63 is the weighted average of the first interest category (corresponding to X_1 in (3.14)) obtained from,

$$X_1 = \frac{53593 * 1 + 20856 * 2 + 9018 * 3...}{53593 + 20856 + 9018 + ...} = 1.63 \tag{3.16}$$

88075 in the denominator of (3.15) is the sum of the supports at various levels of the first interest category (corresponding to Y_1 in (3.14)), obtained from,

$$Y_1 = 53593 + 20856 + 9018 + ... = 88075.$$

The optimal raising level computed by the second approach is 1.95. The optimal raising level computed by the first approach was 1.65. Obviously, only integer level numbers are defined, and rounding would result in both cases in an optimal raising level of 2. Still, the second method is preferable for two reasons. It more closely reflects the visual impression of the graphs, and it introduces a much smaller rounding error. Thus, it is concluded that for the specific data set in this research, raising to level 2 maximizes the total support generated by raising.

Comparing this result with the initial hypothesis, the computed optimal raising level is smaller (to the left, in Figure 3.3) of the expected value. It had been expected that the small number of possible rules at higher levels would reduce the sums of support and that the optimal level would be found as 3 or 4. Presumably, this unexpected effect occurred because so many rules were generated that the cut-off level of WEKA was reached even when mining at higher levels in the hierarchy.

This analysis has involved only support values. Confidence values could be used to compute an optimal raising level. As will be analyzed in Chapter 5, there are cases when raising reduces confidence values of association rules. Thus, the inclusion of confidence values in the calculations might push the optimal raising level to the right (in Figure 3.3), which would be more in line with the initial expectation.

CHAPTER 4

NEW RAISING WITH FP-GROWTH METHOD

4.1 Problem Description

By using proper data mining algorithms, far-reaching conclusions for marketing, in the form of association rules, can be drawn from existing customer databases. Although what is stored in an existing database would only be the purchasing records of existing customers, this information could be used to predict their future purchasing acts. However, instead of focusing on the products already purchased, people may look at the interests of a customer, since interests will possibly lead to purchases. If demographic data are also available in the database, the predictions could be extended to other potential customers in the same category, i.e. in the same age group, in the same marital status group, *etc.* The interests of a customer can be discovered during the online purchasing process because the customer pays attention to other products that are not included in the final purchase. For example, when making a purchase on Amazon.com, all the visited, but not purchased, items are recorded during your ordering process and will be recommended as interesting items (potential purchases) at the next time.

Moreover, taking advantage of the ontology, which works as the backbone in the Web Marketing Project [7, 8], it is possible to preprocess the data before data mining and thus ensure the acquisition of more and better marketing knowledge. This chapter focuses on the effects brought about by the hierarchy of the ontology used in this project.

In Section 4.2, some details of the Raising process will be discussed and the problems existing in using of the WEKA package based on "mining by instance" will be pointed out. Then, the *set-based input mining*, grouped by person, is introduced and a replacement of WEKA is presented in Section 4.3 to solve the previous problems and also to derive more

types of rules. Results are shown in Section 4.4. Conclusions and future work are given in Section 4.5.

4.2 Some Details of Raising

The Raising method improves the support values and therefore the quality of the derived association rules as described in [38, 54] and in Chapter 3. In this section, the Raising process is explained in detail to get a broader view of how the Raising method works and its effects on the results of mining.

4.2.1 What is Raising?

The goal of the Raising method is to take advantage of the interest ontology and derive better association rules with higher support values. Raising the original data to a certain level means that more specific interests are replaced by more general interests, i.e., their parents or ancestors. The pre-processing of the data file operates on the values of the Interest attribute. Age and Gender values are not changed. If an interest involved in a rule is the result of replacement of its children or descendants during Raising, this rule within the mining results will typically have a better support value than the value before Raising. This was shown in [38]. Otherwise, if the interest has no descendant in the ontology hierarchy and thus all the rules involving this interest would not be changed because of it during Raising, the support values of those rules will stay untouched. For example, if there exists a rule {30-39, M}→{Interest A} before applying the Raising method and A does not have any descendant, the "Interest A" would not be effected at all during the Raising and thus would not effect the support value of this rule after Raising. As a result, after Raising, the support values will always be increased, or at least stay the same, since the replacement of the descendants only adds occurrences to the interest or adds nothing if there is no descendant. A detail analysis of this effect will be presented in Chapter 5.

To raise to a level, the replacements of interests modify the mining input file into a new one for this level. If every Raising instance has only one replacement, the modification would be relatively easy by merely replacing that interest by its ancestor in the attribute line in the header section and in every instance line in the data section where necessary. However, such an ideal situation only happens when every interest in the data has only one parent. Since the ontology hierarchy is not a tree, but a DAG (Directed Acyclic Graph), some interests have more than one parent. Thus, when those interests are raised to higher levels, all the parents or ancestors should be included in the data file. For example, the following dataset is going to be raised to level 3. Each line stands for one interest instance with demographic information. The three attributes are now in the order Age, Gender and Interest. Text after a double slash (//) is not part of the data. It contains explanatory remarks.

(20-29),M,BUSINESS_FINANCE //level=1

(40-49),M,METRICOM_INC //level=8

(50-59),M,BUSINESS_SCHOOLS //level=2

(30-39),F,ALUMNI //level=3

(20-29),M,MAKERS //level=4

(20-29),F,INDUSTRY_ASSOCIATIONS //level=2

(30-39),M,AOL_INSTANT_MESSENGER //level=6

(30-39),F,INTRACOMPANY_GROUPS //level=3

The interests included in the data can be found in the ontology hierarchy in Figure 4.1. In order to perform Raising, ancestors at level 3 of the interests in the data need to be found. Table 4.1 and Figure 4.1 show all ancestors of the interests from levels below 3 such that the ancestors are at level 3. The following lines are now added to the file and replacing their children or descendants.

(40-49),M,COMMUNICATIONS_AND_NETWORKING

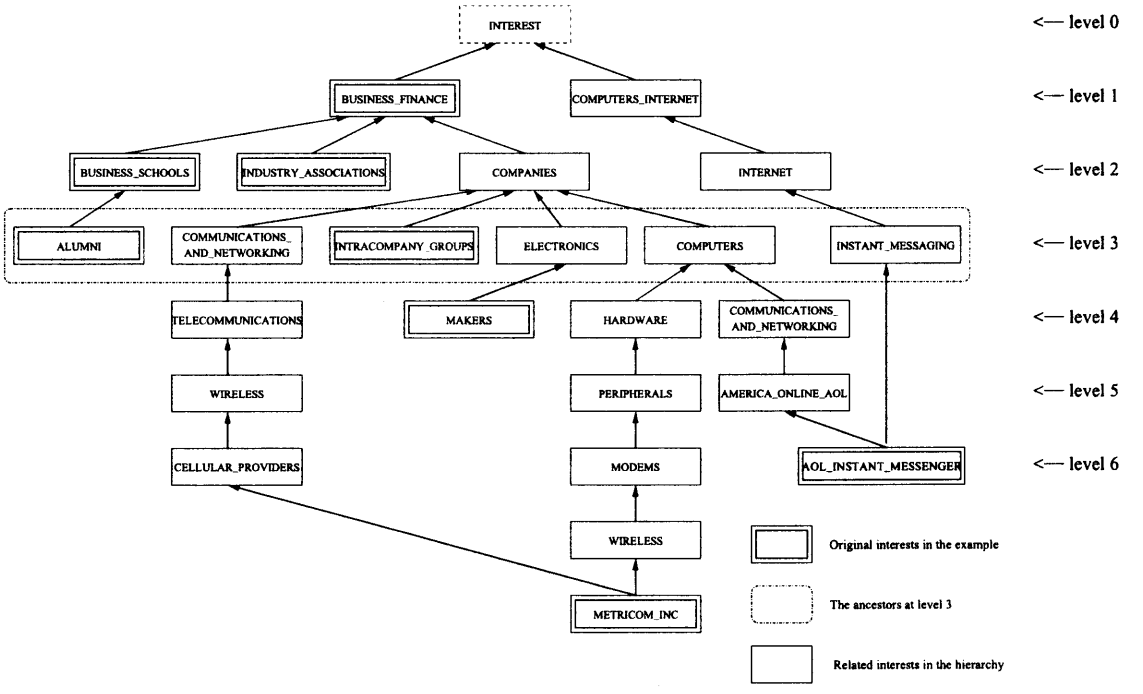


Figure 4.1 Part of an interest ontology.

// Created by replacing METRICOM_INC from level 8 (1st ancestor) by

//COMMUNICATIONS_AND_NETWORKING.

(40-49),M,COMPUTERS

// Created by replacing METRICOM_INC from level 8 (2nd ancestor) by COMPUTERS.

(20-29),M,ELECTRONICS

// Created by replacing MAKERS from level 4.

(30-39),M,COMPUTERS

// Created by replacing AOL_INSTANT_MESSENGER from level 6 (1st ancestor) by

//COMPUTERS.

(30-39),M,INSTANT_MESSAGING

// Created by replacing AOL_INSTANT_MESSENGER from level 6 (2nd ancestor) by

//INSTANT_MESSAGING.

That means, after raising the following dataset is available. Note that the interest "COMPUTER" appears twice.

(20-29),M,BUSINESS_FINANCE

(40-49),M,COMMUNICATIONS_AND_NETWORKING

(40-49),M,COMPUTERS

(50-59),M,BUSINESS_SCHOOLS

(30-39),F,ALUMNI

(20-29),M,ELECTRONICS

(20-29),F,INDUSTRY_ASSOCIATIONS

(30-39),M,COMPUTERS

(30-39),M,INSTANT_MESSAGING

(30-39),F,INTRACOMPANY_GROUPS

Now the result rule $\{\text{Male}\} \rightarrow \{\text{COMPUTER}\}$ will be discussed before and after Raising. Before Raising, the interest COMPUTER did not appear in the data file at all, thus no rule could be mined and the support value is not defined. If this rule is to be retrieved manually for comparison purposes, the support value will be 0. After Raising, COMPUTER appears two times in the data file which means the support value has been increased to 2 as desired.

4.2.2 Generalization

Raising does lose detail and specificity during the process by replacing interests by their ancestors. This is a fact that has positive and negative consequences. A disadvantage would be the missing details due to replacing interests by their ancestors. Those details could have been used as a direct business act indicator about a product. Thus, a rule which involves deep-level interests can explicitely express a connection between customers and products.

Table 4.1 Relevant ancestors

Interest Name	Its ancestor(s) at Level 3
METRICOM_INC	COMMUNICATIONS_AND_NETWORKING
METRICOM_INC	COMPUTERS
MAKERS	ELECTRONICS
AOL_INSTANT_MESSENGER	COMPUTERS
AOL_INSTANT_MESSENGER	INSTANT_MESSAGING

Thus, such a rule might connect a movie DVD with a demographic group. It is, of course, possible to perform data mining before Raising. Thus, no real loss occurs. However, if a confidence and a support threshold are given, any rule has to *qualify* to appear in the results. That is, a rule must have a greater confidence value and a greater support value than the threshold to qualify. Once a rule *qualifies*, it will appear in the results no matter whether Raising is used later or not. Many rules that are mined before Raising tend to have low support values. Thus these rules would not show up anyway. Thus, no new loss is introduced due to Raising. If a rule is not *qualified*, it does not meet the expectations of a *useful* rule. Therefore, to discard such a rule of *little use* and to lose those details is reasonable.

On the other hand, the generalization has the advantage that it provides better indicators for new product promotions. A new product would never appear in any existing rule, thus no exact match can be found. However, it is not a hard problem to categorize the new product into an existing category, or a higher level interest. For example, the 2005 TV comedy "American Dad" had not been listed in Yahoo at the time of this research, i.e. no rule can be found for it by mining. If FOX would like to attract a potential audience for it, the rules involving the interest "television comedy" would be a nice option to consider. Thus, Raising can help to generalize the information from specific interests such as "The Simpsons" or "Family Guy" (two other TV comedies) to "television comedy," if such a

rule is not there before Raising. Even if this rule exists before Raising, the new increased support value after Raising would bring about a better rule quality.

As a conclusion, the Raising operation is not meant to replace the existing mining result rules. Instead, Raising is used to strengthen the derived rules and to provide more possible rules by generalizing detail information.

4.2.3 ARFF Format Problems

The implementation of the Raising method in [38] depends on the usage of the WEKA package [3]. WEKA is a collection of machine learning algorithms for data mining tasks, developed at the University of Waikato in New Zealand. WEKA is a strong tool that provides implementations of learning algorithms for users to apply to their datasets. The datasets to be used must be converted into ARFF (Attribute-Relation File Format) files. An ARFF file includes a header section and a data section. All the attributes have to be listed in the header section, if they are to appear in the data section. Each instance is represented on a single line in the data section. Attribute values for each instance must appear in the order that they were declared in the header section. This format perfectly matches those problems in which an attribute in each instance would have exactly one value, i.e., a common decision problem. The following is the ARFF file for the example shown at the beginning of Section 4.2.1.

```
@RELATION miningsample

@ATTRIBUTE age  {A,B,C,D,E,F}
@ATTRIBUTE gender   {M,F}
@ATTRIBUTE interest {BUSINESS_FINANCE,METRICOM_INC,BUSINESS_SCHOOLS,
                     ALUMNI,MAKERS,INDUSTRY_ASSOCIATIONS,
                     AOL_INSTANT_MESSENGER,INTRACOMPANY_GROUPS}

@DATA
B,M,BUSINESS_FINANCE
D,M,METRICOM_INC
E,M,BUSINESS_SCHOOLS
C,F,ALUMNI
```

```
B,M,MAKERS
B,F,INDUSTRY_ASSOCIATIONS
C,M,AOL_INSTANT_MESSENGER
C,F,INTRACOMPANY_GROUPS
```

However, the ARFF format has its own limitations because it requires one value for each attribute in every instance. In this project, each instance includes three attributes: Age, Gender and Interest. While only one value appears in each instance for Age and Gender, one person might have expressed more than one interest. Because of the requirement of the ARFF format, only one interest can appear in each instance. To feed a dataset into WEKA, the data have to be adjusted to suit the ARFF format. Thus, every line in the data section should include only 3 values, one for age, one for gender and one for one interest. If one person has more than one interests, several lines need to be generated using each of those interests. For example, if a male teenager is interested in ONLINE GAMING and ROCK AND POP, two lines will appear in the ARFF file:

(10-19),Male,ONLINE_GAMING

(10-19),Male,ROCK_AND_POP

Those adjusted data files were used to derive association rules by feeding them into WEKA. To convert data into the ARFF format is the first step for mining using the WEKA package, and also for Raising using WEKA.

4.2.4 A problem of Mining by Instance

Now it will be shown that the traditional method of data mining as explained above produces unreasonably low confidence values with the dataset in this research, because the distribution of interests over several instances. Mining of this representation is called *mining by instance* in this research. It loses important information, namely the fact that many tuples of different interests belong to the same person. Thus a new method of mining called *set-based input mining* is introduced. This will be explained later.

In data mining, the term "set-based" has been used in different senses than in our sense. In [55], only a sub-set called small cover for association rules, instead of all possible rules, was extracted. An algorithm was proposed to refine frequent closed itemsets from frequent itemsets while another Apriori-close algorithm was used to compute simultaneously frequent and frequent closed itemsets. The *set* here is referring to generated itemsets during the mining algorithm.

In [56], an expert-driven approach was presented to validate large numbers of rules (or itemsets), which were discoverd by rule mining methods. A set validation language (SVL) was developed as a user-friendly language to support the validation. "Set-based data" refers to the itemsets and/or rules derived as results of rule mining algorithms. Data are grouped in sets for SVL by applying *cuts* to intial rules sets. However, while the term "set-based" mentioned in those papers is used either in the mining process or post-mining results, our practice is focused on pre-mining datasets. The input datasets are grouped into sets before applying any rule mining algorithms. When *set-based input mining* is mentioned, the sets are always grouped by persons.

In the analysis of the ARFF format in Section 4.2.3, it was stated that only one interest is allowed in each line in the data section. Every line is regarded as an instance. Thus, if one person has more than one interest, more lines have to be generated by combining this person's age and gender with every interest. Thus, if Tom, who is 26, is interested in X and Y, his interests are distributed over two lines by combining them with Tom's age and gender.

26,Male,X

26,Male,Y

It will now be shown that *set-based input mining* is more suitable for this project than mining by instance. In data mining, a typical example to introduce association rules is about the grocery purchases of bread, milk, eggs, *etc*. Every purchase is always regarded as

an independent instance. The association rules are thus derived by data mining to predict the possible relationships between products. These rules would help the store manager to arrange the product shelving. There might be a rule that indicates that a lot (in other words, with a high support value) of people buy milk and bread together during their visit. Among all those who buy milk, most (in other words, with a high confidence value) of them do buy bread. If such rules exist, the manager may take them into consideration when shelving items. However, while this rule mining is based on mining by independent purchase instances, the buyer involved in each purchase might not be independent. Possibly, a customer will make another purchase in the same store, maybe even on the same day. The two purchases made by the same customer therefore will be shown in the dataset as two purchase instances. However, by collecting and mining the purchases made by one customer, the rules can indicate the buying pattern for him/her. Thus, all the purchases made by the same customer should be properly combined together as a person's record. This is done in set-based input mining, grouped by person, instead of mining by instance. The derived association rules are then used to relate the person and the product they purchased.

For example, assume a dataset has n men's records, and m men are interested in X. The confidence value for rule {Male}$\rightarrow${Interest X} would be calculated by m/n. This rule result is based on applying set-based input mining, which can be interpreted as "Among n men, there are m of them who are interested in X." However, if all men in this example have two interests expressed at the same time, according to the ARFF format, every man's record has to be converted into two lines in the data section in the ARFF file. In this case, the mining process becomes mining by instance. One single person's record has been split into two instances in the ARFF file. The instance amount now is $2n$, twice the number of males. As a result, the occurrence of the attribute value Male has been doubled but the occurrence of Interest X in the data has not been changed. After input of such a data file into WEKA, the resulting confidence value from mining by instance will be $m/2n$, which is

half of the value from set-based input mining. This reduction in confidence is unacceptable and unnecessary.

4.3 Replace WEKA by FP-Growth

According to the description in Section 4.2, the ARFF format creates some problems with the data. As a solution, an alternative method is used to avoid the limitations of the ARFF files. To achieve this, the total number of all the instances should stay the same, *i.e.* the number of person records should be constant. If one line stands for a record for one unique person, then the number of lines in the input file should be equal to the number of people included in the data and also should not be changed during the Raising processing. All the instances of one person are grouped together to become a set-based input. Thus, all the records of one person are maintained in the same line in the input. All the values of attributes are listed together in a certain order. If an attribute has more than one value, all values are listed together in the same line, which is different from what is done in an ARFF file. For instance, for the record of Tom, as mentioned in Section 4.2.4, only one line is needed to present the information: [(20-29),M,X,Y]. Both interests are listed after the Gender attribute. Recalling that the same problem also exists in the process of Raising, the new format also fits into the whole processing by replacing the interests in the line. In the example in Section 4.2.1, suppose the data actually belong to 6 persons. All the interests with the same person's attribute belong to the same person. Converting the ARFF format to the new format, the data before Raising will have the following format and the interests in bold are those that are from separate lines but belong to the same person:

(20-29),M,BUSINESS_FINANCE,**MAKERS**

(40-49),M,METRICOM_INC

(50-59),M,BUSINESS_SCHOOLS

(30-39),F,ALUMNI,**INTRACOMPANY_GROUPS**

(20-29),F,INDUSTRY_ASSOCIATIONS

(30-39),M,AOL_INSTANT_MESSENGER

After raising to level 3, the ancestors at level 3 replace those interests below level 3. Thus, the input for mining will become as follows. Interests in bold are those raised from the newly combined interests above or from interests with multiple parents:

(20-29),M,BUSINESS_FINANCE,**ELECTRONICS**

(40-49),M,**COMMUNICATIONS_AND_NETWORKING,COMPUTERS**

(50-59),M,BUSINESS_SCHOOLS

(30-39),F,ALUMNI,**INTRACOMPANY_GROUPS**

(20-29),F,INDUSTRY_ASSOCIATIONS

(30-39),M,**COMPUTERS,INSTANT_MESSAGING**

The number of lines has stayed the same but the level-3 interests have been included. If one interest has more than one ancestor at level 3, all the ancestors are inserted. For example, in the last record in the data, who was interested in AOL_INSTANT_ MESSENGER, the interest AOL_INSTANT_ MESSENGER had two level-3 ancestors: COMPUTERS and INSTANT_MESSAGING. Both of them are in the same line of the data after Raising, replacing the original interest AOL_INSTANT_ MESSENGER.

There is another advantage of the new file format over the ARFF format. In some situations, if one person has several interests which are going to result in the same ancestor during Raising, the ARFF format would create more confusion. For instance, if one record of a person is:

(30-39),M,METRICOM_INC,AOL_INSTANT_MESSENGER

The ARFF format will be:

(30-39),M,METRICOM_INC

(30-39),M,AOL_INSTANT_MESSENGER

In Figure 4.1, it can be found that the interest METRICOM_INC and the interest AOL_INSTANT_MESSENGER have the same ancestor at level 2, which is COMPANIES. While raising the ARFF format file to level 2, the result data become:

(30-39),M,COMPANIES

(30-39),M,COMPANIES

Again noticing that these lines actually belong to only one person, one person should not express the same interest twice. The result in such a situation causes a confidence value error. If the confidence value of this rule while not counting this as one person is a/b, the actual value while counting this as one person should be $(a+1)/(b+1)$. However, according to the two lines of results generated above, the calculated confidence value will be $(a+2)/(b+2)$. Moreover, it will affect the support value after Raising, since duplicates of interests have been inserted. However, the solution to this problem is easy for the new format. A check for each line while Raising would eliminate those duplicates. Thus, the new line after raising to level 2 should be:

(30-39),M,COMPANIES

One of the ancestor COMPANIES has been removed during the duplicate checking. This line exactly stands for the meaning that there is one man between 30 to 39 whose interest is COMPANIES, after being raised to level 2.

Thus, the new input format is defined as follows:

- All data are contained in an ASCII file.
- Each person's information is represented by one line in the input file.
- In each line, all terms are listed followed immediately by a comma.
- A line for a person's record is listed in the order of age range, gender, and interest IDs.

For example, one line in the file "business_finance_age_gender_mining" is:

```
C,FEMALE,1600000003,1600909556,1600909559,
```

This line describes a woman whose age is between 30 and 39. She had expressed three interests, "BUSINESS_SCHOOLS" (1600000003), "CONSULTING" (1600909556), and "HUMAN_RESOURCES" (1600909559).

The two major differences between the new input format and the ARFF format of WEKA are: (1) Only data and commas are needed in the file with the new input format. There is no extra information needed, such as the header section in an ARFF file. (2) Any attribute with more than one value can be easily introduced into the new input. All one needs to do is to add the extra interest values at the end. There is no limit for the maximum number of values allowed.

The overall Raising procedure is performed in the following steps:

- Data Preparation
  1. Choose the domain (category) of mining, such as "BUSINESS_AND_FINANCE."
  2. Retrieve the corresponding data from the database.
  3. Write the data into a file "DataFile" in the new input format defined above.
- Raising Operation.
  1. Scan all the interests in "DataFile." Create the ancestor table for all the interests involved. Find the lowest level L_{low} among all the interests in the file.
  2. Raise the data in the file "DataFile" into i new files at higher levels, from level $(L_{low} - 1)$ to level 1. In the file at level i, there is no interest below level i.
- Rule Generation
  1. Select reasonable support and confidence thresholds, such as $S = 0.01$, $C = 0.1$.
  2. Feed the L_{low} input files into the FP-Growth program to derive rules.

- Result Analysis

    1. Remove useless rules from the rule files, such as demographic-demographic rules.

    2. For analysis purposes, each rule appearing in the i rule files is scanned for collecting statistical information.

4.3.1 Apriori vs. FP-Growth

The format change of the input file makes it improper to use the WEKA package, since the ARFF format is the only input WEKA will accept. In WEKA, it uses the Apriori algorithm [35, 4], which was developed as early as 1993. In the Apriori algorithm, the problem of mining association rules is decomposed into two parts: (1) Find all combinations of items that have transaction support above a minimum support value. Those combinations are called frequent itemsets. (2) Use the frequent itemsets to generate the desired rules. The core of the Apriori algorithm is the candidate set computation. It uses frequent *(k-1)*-itemsets to generate candidate frequent *k*-itemsets and uses database scans and pattern matching to collect occurrences of the candidate itemsets. However, the candidate generation is also the bottleneck of Apriori. It might cause a huge size of candidate sets and require multiple scans of the database. Researchers have done a lot to improve association rule mining algorithms. The Frequent Pattern growth (FP-Growth) method is a better algorithm. In [5,6], this method has been introduced for frequent itemset discovery without candidate generation. By avoiding multiple costly database scans, the FP-Growth method compresses a large database into a compact Frequent-Pattern tree (FP-Tree) by only two scans of the database. The FP-Tree is highly condensed, but is complete for frequent pattern mining.

As analyzed in [6], the FP-Growth method performs well in dense datasets, by having an FP-Tree of good compactness. However, when support is very low, the FP-Tree becomes bushy. In this project, all the rules to be used for marketing purposes also need a relatively

Table 4.2 Binary mapping representation

Name	Age	Music	Games	Computers
Tom	26	1	1	0
Mary	18	0	0	1
Jerry	23	1	0	1
Spike	40	0	1	1

high support value, which avoids too complicated an FP-Tree. Thus, it was decided to use the FP-Growth method to replace Apriori.

4.3.2 Binary Mapping in WEKA would not Work for Large Datasets

As stated above, the previous method using WEKA does not deal well with situations of people who have multiple interests. One may propose to use a binary mapping to solve the problem of multiple-value attributes. Thus, for every possible interest, a binary value could be assigned. A value of 1 or 0 shows whether the corresponding interest is assigned to the person or not. For example, in Table 4.2 all the interest values now become individual attributes. Though this representation might look neat, there is one major drawback, because of the data sparsity.

For a small amount of data, this representation would be a nice way to store and process it and could be translated into ARFF format. However, consider that there are 31,534 interests in this project! If the whole dataset will be processed at one time, more than thirty thousand attributes are needed and thus 31,534 binary values in one record for one person. This is not practical. Moreover, there is no person who will list thousands of their interests. In the database of this research, people expressed at most 73 interests. Thus, most of the binary values in the line will be 0s. The 1s will be distributed in the data quite sparsely, compared with the large number of interests and number of persons. Apparently, such an implementation is not practical and is a waste of resources. An experiment was

Table 4.3 Performances of both methods on two datasets

Dataset Name	HOBBIES CRAFTS	RECREATION SPORTS
Number of Persons	13,760	23,128
Number of Interests	203	1,631
Max. Interest per Person	24	49
Input File Size for FP-Growth	353,404 bytes	5,686,110 bytes
Input File Size for WEKA	892,284 bytes	75,440,380 bytes
Execution time by FP-Growth	00:00:24	00:01:08
Execution time by WEKA	00:24:16	09:11:18 (Crashed)
Confidence$\geq$0.6, Support$\geq$0.01		

performed using two real datasets as samples. Table 4.3 shows the slow processing speed and the huge input file size of the binary mapping, compared to our method.

Even if the binary mapping representation would be used for a small dataset, 0s, will cause some logical problems if not properly handled. In our result rules, the connections relate to interests. All the interests are expressed by some persons, which are the 1s in this binary mapping. When the tuples with 1s and 0s are fed into a mining method, the 0s will also be regarded as valid values in the itemsets and then appear in the result rules. Thus, there might be a rule such as:

$\{\text{Male, Interest\_A, } \neg\text{Interest\_B}\} \rightarrow \{\neg\text{Interest\_C}\}$

This rule can be interpreted as "A male who is interested in Interest_A but not interested in Interest_B will not be interested in Interest_C." Though this rules looks interesting, it would not help a lot because of two reasons. First, when a person has not listed Interest_B as one of his interests, it does not neccessarily mean he is not interested in Interest_B. The assumption is unjustified until he has expressed it explicitely. In technical terms, the researchers cannot make the closed world assumption. The mining data should be based on

existing information. Moreover, as in marketing knowledge practices, a product promotion is based on what a customer is interested in. Thus, a helpful rule is expected to imply something to be interested in, rather than something he is not interested in.

In a second experiment, several randomly generated datasets have been constructed for testing. In each dataset there are records of people and their interests. For WEKA processing, ARFF files have been created to present the binary mapping. All the interests were represented as attributes and the values for each interest were limited to 1 and 0. Moreover, to avoid the side effects mentioned above, which are caused by too many 0s, every 0 is replaced by a "?" to represent a missing value. Thus, one person's record in the data section with 10 interests might be represented as the following line:

$$Male, C, 1, ?, ?, 1, 1, 1, ?, ?, ?, 1$$

For comparison reasons, the algorithm from this research was also applied to the same dataset, in the input format described before. The same record above will be represented as:

$$Male, C, Int\_1, Int\_4, Int\_5, Int\_6, Int\_10,$$

Comparing to ARFF files, the relation section and the attribute section are not needed. Only expressed interests are shown in each record. Table 4.4 shows the details of the experiments. During the test of WEKA, the exception "java.lang.OutOfMemoryError" occurs when the number of interests reaches 187. Though the performance of the mining algorithm is not only decided by the number of the interests, but also how the data would appear in the dataset, the test still proves that a relatively large amount of attributes will crash the program. In summary, with thirty thousand interests, the idea of merely using binary mappings is not practical. Moreover, notice the execution times of the experiments in Table 4.4. The two methods have their own advantages. WEKA runs faster when the

Table 4.4 Binary mapping testing

Number of Interests	Execution Time		Exception in WEKA
	by WEKA	by FP-Growth	
25	00:00:09	00:00:32	
50	00:00:46	00:00:49	
68	00:01:33	00:01:33	
75	00:01:52	00:01:47	
100	00:08:23	00:04:16	
150	00:42:57	00:14:18	
175	00:48:54	00:21:52	
186	01:21:16	00:40:47	
187	01:15:27	00:30:01	OutOfMemoryError
200	01:02:40	00:58:15	OutOfMemoryError
250	00:46:24	02:14:37	OutOfMemoryError
200 records, Confidence≥0.8, Support≥0.2			

number of interests is relatively small, *i.e.* less than 68. When the number of interests becomes larger, WEKA is always slower than our method.

4.3.3 Deriving Interest-to-Interest Rules

By using the new set-based input file for mining by FP-Growth, the confidence reduction during mining is successfully avoid. Moreover, by having all the interests in the same line, Interest-to-Interest association rules can also be derived.

Association rules derived from the dataset may be of different formats. Among all the rule types that can be derived, there is a rule type {Interest}→{Interest}. This rule type has attributes Interest as both antecedent and consequent. It is impossible to derive

such rules when each attribute allows only one value in each line in the ARFF format. However, these kinds of rules are attractive for marketing purposes. When a retailer is going to promote a product, which is categorized by interest X, to his potential customers, he might prefer association rules which can lead to persons grouped by age, gender, *etc.* However, there might not be a rule (due to insufficient support or confidence values during rule mining) like {Age, Gender}→{Interest X} to direct him to a certain group, or there is a rule, but the number of people belonging to that group is too small. Therefore, rules such as {Interest Y}→{Interest X} would greatly support his search. This sounds similar to Agrawal's original work. However, the point is that it was very hard to be executed in the widely-used WEKA package, and it is easier to do in the implementation done in this research.

4.4 Results

The novel Raising method has been implemented using set-based input, as described in the previous sections. The new implementation avoids the problems caused by spreading out interests of one person over several lines in ARFF format and makes it possible to derive better rules. An FP-Growth mining program was implemented using JAVA. The input specifies a file which has to be formatted as required for *set-based input mining*, i.e., a person's record is in a single line, including age, gender and all his/her interests. For example, some lines in the input file after Raising at level 6 in the category "BUSINESS & FINANCE" are as follows:

```
B,FEMALE,1600840341,1602248651,
C,MALE,1600000236,1600001222,1600909559,1600909561,
C,MALE,1600840352,1602216980,1602216981,1602219139,1602236895,
D,MALE,1600000393,1600001278,1600001779,1600193918,
```

The increments of support values after Raising are shown at the left side in Table 4.5. The percentages are computed as the difference between the average support values of

mining results from raised data at level i (S_{ai}) and from unraised data (S_b) and then divided by S_b. Thus,

$$I_i = ((S_{ai} - S_b)/S_b) \times 100\%$$

and I_i is the increment rate of the support value at level i relative to the original value before Raising. Since result level 1 only contains one interest and results from levels below level 5 only have sparse data or even do not exist, only level 2 to level 5 appear. The data show the concrete increments of support values from lower levels to higher levels. The right side of Table 4.5 shows the number of newly discovered {Interest}→{Interest} rules, which are unreachable by the previous method. For example, some rules mined from the input file raised to level 6 in category "BUSINESS & FINANCE" are as follows:

```
{1600416059, } 203 ==> {1600005582, } 203    Conf:(1.0)
Supp:(0.03)

 {1600126227, 1600416059, } 72 ==> {1600005582, } 72 Conf:(1.0)
Supp:(0.01)

{1600750846, } 81 ==> {1600005582, } 81 Conf:(1.0)  Supp:(0.01)

{1600291739, } 107 ==> {1600005582, } 62 Conf:(0.58) Supp:(0.01)

{1600005582, 1600126227, } 133 ==> {1600416059, } 72 Conf:(0.54)
Supp:(0.01)

{1600126227, 1600145997, } 129 ==> {1600005582, } 63
Conf:(0.49)  Supp:(0.01)
```

However, since these rules are represented with IDs, IDs are translated into corresponding interest names as follows:

```
{INTERNET_MARKETING_AND_ADVERTISING}==>{INTERNET_BUSINESS}

{HOME_BUSINESS,
INTERNET_MARKETING\_AND_ADVERTISING}==>{INTERNET_BUSINESS}
```

Table 4.5 Support value increments and new rule discovery

Category	Level 2	Level 3	Level 4	Level 5	Interest-Interest Rules
BUSINESS FINANCE	858.79%	371.90%	74.02%	5.60%	115
COMPUTERS INTERNET	946.25%	749.90%	97.66%	3.53%	26
FAMILY HOME	341.41%	146.17%	46.16%	0.15%	6
GOVERNMENT POLITICS	4084.36%	2320.00%	2090.90%	1119.50%	169
RECREATION SPORTS	853.49%	251.86%	64.35%	11.67%	2
SCHOOLS EDUCATION	877.91%	459.82%	249.03%	20.72%	23
SCIENCE	1661.34%	971.87%	894.58%	751.98%	13
Data mined at Confidence≥0.6, Support≥0.02					

```
{STARTUPS}==>{INTERNET_BUSINESS}

{NETWORK_MARKETING}==>{INTERNET_BUSINESS}

{INTERNET_BUSINESS,
HOME_BUSINESS}==>{INTERNET_MARKETING_AND_ADVERTISING}

{HOME_BUSINESS, SMALL_BUSINESS}==>{INTERNET_BUSINESS}
```

The novel Raising implementation solves the problems described earlier in this chapter. It results in better performance while still improving support values over the previously published Raising method [38]. It also eliminates in a natural way the duplication of tuples, which might occur during Raising when an ancestor is reachable by more than one path in a DAG. The application of the FP-Growth mining algorithm also results in better efficiency than the previously used Apriori algorithm, as shown in Section 4.3.2.

4.5 Conclusions

In this chapter, it was shown that the use of a set-based input for a data mining algorithm provides a new approach to deriving association rules. Instead of mining rules based on instances, the results are derived by inputting data which have been grouped into sets according to desirable attributes of the data. In this project, the input data are grouped by individual persons. The interests of each person are created by merging all the related instances. In this way, the association rules retrieved from those set-based inputs possess better support and confidence values. Moreover, to optimize the mining processes, the FP-Growth algorithm was newly implemented and used instead of the WEKA package Apriori implementation. The implementation of FP-Growth supports set-based inputs. Though WEKA can also deal with set-based input by coding it as a binary mapping, as shown in tests at the end of Section 4.3.2, our method performs much better in the situations where the unique interest number in the dataset is large. In the experiment described before, WEKA crashed when the interest number exceeded 187. However, in the database of this research, the smallest number of interests in a dataset is 203 and the largest is 3,847, with an average of 882. Therefore, to use a WEKA binary mapping is not practical.

CHAPTER 5

RAISING RESULTS ANALYSIS

5.1 Observations about Raising Results

To derive association rules, data are selected from the database and fed into data mining algorithms. In this project, both the Apriori algorithm [35,4] in the WEKA package [3] and the FP-Growth algorithm [5, 6] are used. By providing minimum support and confidence values as input parameters, the data mining algorithms return derived association rules based on the input. However, as described in [38], a problem in the derivation of association rules is that available data is sometimes very sparse and biased. For example, among over a million of interest records in the database of this research, only 11 people expressed an interest in RECREATION_SPORTS, and nobody expressed an interest in SCIENCE which is counter-intuitive.

Below is an example to illustrate how the Raising method works to improve the quality of derived association rules repeated from Chapter 3 for readability. The following dataset is the input to a data mining algorithm and is going to be raised to level 3. Each tuple, i.e. each line, stands for one interest instance with demographic information. As before, the values of three attributes (Age, Gender and Interest) are all included in each tuple. Age values are represented as a range while Gender values of male and female are abbreviated as M and F. Text after a double slash (//) is not part of the data. It contains explanatory remarks.

(20-29), M, BUSINESS_FINANCE //level=1

(40-49), M, METRICOM_INC //level=8

(50-59), M, BUSINESS_SCHOOLS //level=2

(30-39), F, ALUMNI //level=3

(20-29), M, MAKERS //level=4

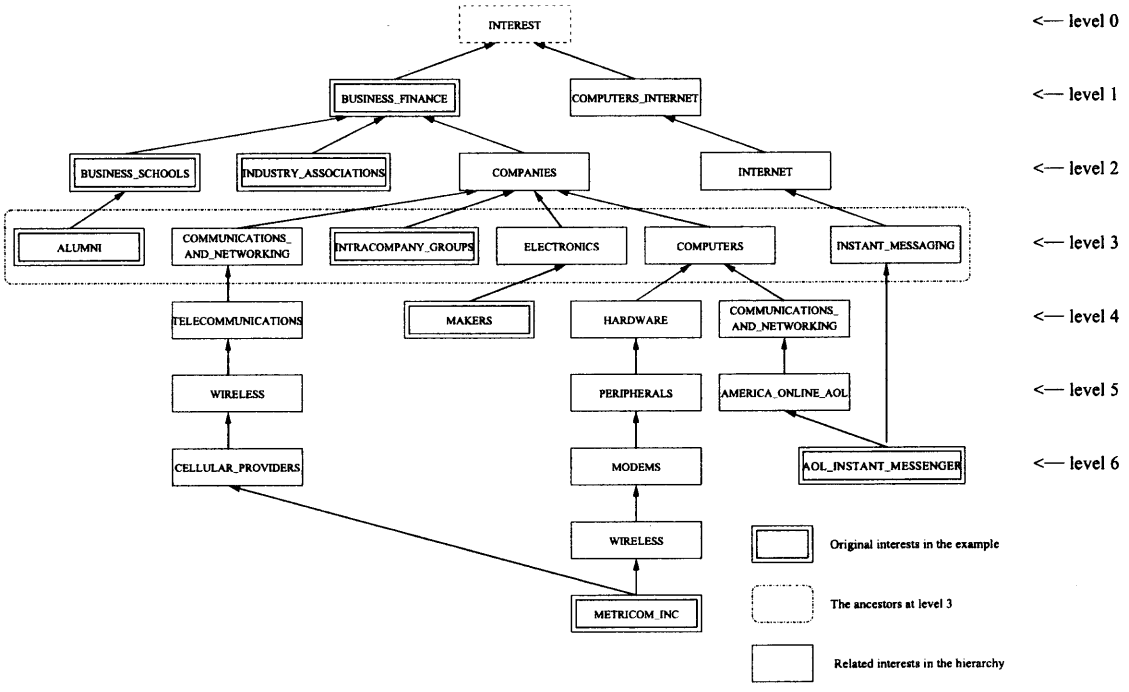


Figure 5.1 An example of raising to level 3.

(20-29), F, INDUSTRY_ASSOCIATIONS //level=2

(30-39), M, AOL_INSTANT_MESSENGER //level=6

(30-39), F, INTRACOMPANY_GROUPS //level=3

Once this original dataset is fed into a data mining algorithm, the best association rules, as measured by support value, that can be derived are $\{(20\text{-}29)\} \rightarrow \{M\}$ and $\{(30\text{-}39)\} \rightarrow \{F\}$. Both rules have confidence values of 0.67 and support values of 2. This is also an example of sparse data. Every interest only appears once in the dataset. Though rules with a confidence of 1.0 can be derived from the data, such as $\{(50\text{-}59), M\} \rightarrow \{BUSINESS\_SCHOOLS\}$, as discussed in Section 5.1, the low support value of 1 does not make the rules useful for marketing purposes.

While performing Raising, ancestors are found at level 3 of the interests in the data. Table 5.1 shows all ancestors of the interests from levels below 3 such that the ancestors are at level 3. Table 5.1 is based on the DAG hierarchy in Figure 5.1, repeated from Figure 4.1. As seen in Figure 5.1, among the eight interests in eight tuples, two of

Table 5.1 Relevant ancestors

Interest Name	Its ancestor(s) at Level 3
METRICOM_INC	COMPUTERS, COMMUNICATIONS_AND_NETWORKING
MAKERS	ELECTRONICS
AOL_INSTANT_MESSENGER	COMPUTERS, INSTANT_MESSAGING

them (ALUMNI, INTRACOMPANY_GROUPS) are already at level 3, and three of them (BUSINESS_FINANCE, BUSINESS_SCHOOLS, INDUSTRY_ASSOCIATIONS) are at levels above. Therefore, the Raising process will only function on the other three interests (METRICOM_INC, MAKERS, AOL_INSTANT_MESSENGER) which are at levels 4, 6 and 8 respectively. Table 5.1 lists their ancestors at level 3. While the interest MAKERS has only one ancestor (parent) at level 3, the other two interests METRICOM_INC and AOL_INSTANT_MESSENGER both have two ancestors at level 3. The interest ontology is not a tree but a DAG. Some interests have more than one parent and thus more than one ancestor at a certain level. The Raising to level 3 then replaces all the interests below level 3 in the original dataset by their ancestors at level 3. By doing so, all the interests in the new dataset are at level 3 or above. Thus, the new dataset after being raised to level 3 becomes:

(20-29), M, BUSINESS_FINANCE

(40-49), M, COMMUNICATIONS_AND_NETWORKING, COMPUTERS

(50-59), M, BUSINESS_SCHOOLS

(30-39), F, ALUMNI

(20-29), M, ELECTRONICS

(20-29), F, INDUSTRY_ASSOCIATIONS

(30-39), M, INSTANT_MESSAGING, COMPUTERS

(30-39), F, INTRACOMPANY_GROUPS

By feeding the new dataset as input to data mining algorithms, a new association rule, whose support value is greater than 1, is derived other than the two demographic rules derived before:

{M} → {COMPUTERS} Confidence: 0.5 Support: 2

This new rule is relatively more attractive for marketing purposes than the results from the original dataset, for the following reasons.

1. The new rule has a better support value, thus it is a rule of higher quality. The occurrence count of an interest at the raised level in the dataset is increased by replacing its descendants in all instances. During the replacement, the demographic information and the interests at levels above are not effected or updated. Thus, while the number of tuples in the dataset is still the same, the support value is improved.

2. The new rule connects demographic information with interest information. The rules derived originally, such as:

   {20-29} → {M} Confidence: 0.67 Support: 2

   {30-39} → {F} Confidence: 0.67 Support: 2

   {F} → {30-39} Confidence: 0.67 Support: 2

   only imply some connections between age and gender. Though those rules are valid, they do not contribute any useful insights for marketing purposes. For marketing usage, an interest should be included in the rules to predict future purchases of potential customers.

3. Last but not least, "brand-new" rules can be derived after Raising. Note that in the original dataset, the interest COMPUTERS did not exist at all. This interest at level 3 is introduced when several interests in the original dataset share it as ancestor. In other words, the newly appeared interest is a generalization of its descendants based on the interest ontology. As the example showed at the beginning of this section, just because people did not express interests with more general terms does

not mean they are not interested. On the contrary, people prefer to express their interests more specifically. In the data file, there are 62,734 data items in the category of RECREATION_SPORTS. These thousands of people prefer saying something like "I'm interested in BASKETBALL and FISHING" instead of saying "I'm interested in RECREATION_SPORTS." By the Raising method, those wide-spread data can be collected and thus new rules can be derived to describe the situation by using high-level interests.

5.2 Effects of Raising on Different Rule Types

Since the inputs only include three attributes for each person's record, all the association rules are combinations of those attributes in their antecedents and consequents. For the age and gender attributes, only one single value is allowed for each attribute in every tuple, i.e. one person's record. However, since a person might have expressed more than one interest at the same time, the interest attribute may have multiple values. Here all the possible rule types based on this situation are listed. (The expression {Interest(s)} stands for one or more interests. For example, the rule {Male} $\rightarrow$ {FISHING, POKER} is categorized by the rule type {Gender} $\rightarrow$ {Interest(s)}.) Rules with an empty antecedent or consequent are not interesting.

1. {Age} $\rightarrow$ {Gender}
2. {Age} $\rightarrow$ {Interest(s)}
3. {Age} $\rightarrow$ {Gender, Interest(s)}
4. {Age, Gender} $\rightarrow$ {Interest(s)}
5. {Age, Interest(s)} $\rightarrow$ {Gender}
6. {Age, Interest(s)} $\rightarrow$ {Interest(s)}
7. {Age, Interest(s)} $\rightarrow$ {Gender, Interest(s)}
8. {Age, Gender, Interest(s)} $\rightarrow$ {Interest(s)}
9. {Gender} $\rightarrow$ {Age}

10. {Gender} → {Interest(s)}
11. {Gender} → {Age, Interest(s)}
12. {Gender, Interest(s)} → {Age}
13. {Gender, Interest(s)} → {Interest(s)}
14. {Gender, Interest(s)} → {Age, Interest(s)}
15. {Interest(s)} → {Age}
16. {Interest(s)} → {Gender}
17. {Interest(s)} → {Interest(s)}
18. {Interest(s)} → {Age, Gender}
19. {Interest(s)} → {Age, Interest(s)}
20. {Interest(s)}→ {Gender, Interest(s)}
21. {Interest(s)} → {Age, Gender, Interest(s)}

These 21 rule types include all the possibilities of derived association rules. By studying these rule types one by one in groups, all the rules which are not proper for marketing purposes are filtered out and the research focuses on the effect of Raising on the remaining rule types.

- **Group (A):** The rule types #1 and #9 only involve Age and Gender. As discussed in Section 5.1, such rules are not useful for marketing purposes and thus are filtered out.

- **Group (B):** The rules which are useful for marketing purposes should be those which connect a certain group of persons to a certain interest, or product. The types #2, #4 and #10 are exactly predicting the relationship between a group of persons and their interests. They strongly tie people and their interests together. Moreover, type #4 is more specific than types #2 and #10. Once such rules with a high confidence and a high support value are found, the group of people described by the antecedent is more likely to make purchases related to the interest.

- **Group (C):** The types #15, #16 and #18 are the opposite of #2, #4 and #10. The attribute interest is in the antecedent while demographic attributes are in the consequent. The interpretation for such rule types is "If somebody is interested in A, this person is likely to be in a certain demographic group B." These rules describe the distribution of person groups within all those who are interested in an interest A. The types #5 and #12 can also be categorized in this group since there is only the demographic attribute in the consequents. These rule types are less useful for promotion purposes which this project is focused on.

- **Group (D):** The types #3 and #11 have only demographic attributes in the antecedents. In the consequents are the combinations of a demographic attribute and interest attributes. For example, #3 can be interpreted as "If a person is in the age group B, there is a certain confidence that this person has a gender C and will be interested in the interest A." A more specific example is "If a person is a teenager, there is a confidence of 0.8 that it's a girl who is interested in SOFTBALL." By connecting the Age attribute and Gender attribute in the rules, the interpretation of these two types of rules is confusing and they appear not suitable for marketing.

- **Group (E):** The types #6, #8, #13 and #17 have only the interest attribute in the consequents. The rule type #17 implies a connection between two or more interests. The types #6, #8 and #13 are more specific formats of type #17 by including demographic groups. These kinds of rules are attractive for marketing purposes. When a retailer is going to promote a product, which is categorized by interest X, he might prefer association rules which can lead to persons grouped by age, gender, etc. However, there might not be a rule (due to insufficient support or confidence values during rule mining) of rule type #4 from Group (B). Therefore, rules in this group would greatly support his search as a complement of Group (B).

- **Group (F):** The types #7, #14, #19, #20 and #21 also contain a combination of demographic attribute and interest attribute in consequents, like the rule types in Group (D). The difference to Group (D) is that interest attributes appear in the antecedent. Type #7 and #14 are more specific formats of type #20 and #19 respectively. The usefulness of these rules for marketing is doubtful. For example, a rule can be "If a man is interested in FOOTBALL, there is a certain confidence that his age is 30 to 39 and he is also interested in BEER." These rule types try to connect interest attributes to demographic attributes and are hybrids of Group (C) and Group (E). However, for marketing purposes, these rule types are weaker than those in Group (E).

All the 21 rule types have been categorized into 6 groups. The effects by Raising can be analyzed group by group. Before Raising, a rule has a support value of $S_0 = Occ_{ante\&con}$ and the confidence value is calculated by $C_0 = \frac{S_0}{Occ_{ante}}$.

- **Group (A):** After Raising, since the instances of age and gender have not been changed and no interests occur, the confidence and support values are not affected.

$$S_{new} = S_0$$
$$C_{new} = C_0$$

- **Group (B):** After Raising, the occurrences of demographic information in the antecedents are not changed. However, the occurrences of interests in the consequent might be increased by replacing descendants by multiple ancestors. If there is no replacement needed, the occurrences stay unchanged. Thus, the support values always stay unchanged or are increased and the confidence values also stay unchanged or are increased accordingly. Suppose the increment of occurrence is Inc ($Inc \geq 0$), then

$$S_{new} = S_0 + Inc \geq S_0$$
$$C_{new} = \frac{S_0 + Inc}{Occ_{ante}} \geq C_0$$

- **Group (C):** After Raising, the occurrences of interests in the antecedent are increased. Suppose the increment of occurrences of the antecedent is Inc_{ante} ($Inc_{ante} \geq 0$). However, among all the tuples updated with these interests, the demographic information might not match those in the rule, thus the increment of occurrence of both antecedent and consequent will be a different variable, $Inc_{ante\&con}$ ($Inc_{ante} \geq Inc_{ante\&con} \geq 0$). Therefore,

$$S_{new} = S_0 + Inc_{ante\&con} \geq S_0$$
$$C_{new} = \frac{S_0 + Inc_{ante\&con}}{Occ_{ante} + Inc_{ante}}$$

The comparison of C_{new} to C_0 depends on the two increments. If $Inc_{pre\&con}$ is much smaller than Inc_{pre}, C_{new} could be less than C_0. Otherwise, $C_{new}Inc \geq C_0$. More specifically,

$$\begin{aligned}
&C_{new} - C_0 \\
&= \frac{S_0 + Inc_{ante\&con}}{Occ_{ante} + Inc_{ante}} - \frac{S_0}{Occ_{ante}} \\
&= \frac{(S_0 + Inc_{ante\&con})Occ_{ante} - S_0(Occ_{ante} + Inc_{ante})}{Occ_{ante}(Occ_{ante} + Inc_{ante})} \\
&= \frac{Inc_{ante\&con}Occ_{ante} - S_0 Inc_{ante}}{Occ_{ante}(Occ_{ante} + Inc_{ante})} \\
&= \frac{\frac{Inc_{ante\&con}}{Inc_{ante}} - \frac{S_0}{Occ_{ante}}}{\frac{Occ_{ante}}{Inc_{ante}} + 1}
\end{aligned} \tag{5.1}$$

Since all the values in the equation are non-negative, the value of the numerator in Equation 5.1 shows which is greater, C_{new} or C_0. If the value is non-negative, C_{new} is greater than or equal to C_0. Otherwise, if the value is negative, C_{new} is less than C_0. Thus,

$$\frac{Inc_{ante\&con}}{Inc_{ante}} \geq \frac{S_0}{Occ_{ante}} \Longrightarrow C_{new} \geq C_0$$
$$\frac{Inc_{ante\&con}}{Inc_{ante}} < \frac{S_0}{Occ_{ante}} \Longrightarrow C_{new} < C_0$$

Notice that $C_0 = \frac{S_0}{Occ_{ante}}$. Let's take a look at the $\frac{Inc_{ante\&con}}{Inc_{ante}}$. The numerator is the increment of the records which contain all the terms in both antecedent and consequent. The denominator is the increment of the records which contain all the terms in the antecedent. Thus, for $C_{Inc} = \frac{Inc_{ante\&con}}{Inc_{ante}}$, C_{Inc} is the confidence value of the rule mined from the sub-dataset which contains all the updated records, *i.e.* records which have interests being replaced by their ancestors, during the Raising. In other words, the changes of confidence values before and after Raising are based on the confidence value from the sub-dataset which contains all the Raising-effected records.

Thus, if $C_{Inc} \geq C_0$ then $C_{new} \geq C_0$ and if $C_{Inc} < C_0$ then $C_{new} < C_0$.

- **Group (D):** As in Group (B), the occurrences of demographic information in the antecedents are not changed after Raising. The increment of occurrences of both antecedent and consequent $Inc_{pre\&con}$ ($Inc_{pre\&con} \geq 0$) depends on the increment of occurrence of interests.

$$S_{new} = S_0 + Inc_{ante\&con} \geq S_0$$
$$C_{new} = \frac{S_0 + Inc_{ante\&con}}{Occ_{ante}} \geq C_0$$

Therefore always: $S_{new} \geq S_0$ and $C_{new} \geq C_0$

- **Group (E) and Group (F):** These two groups can be put together since the interest attributes appear in both antecedent and consequent. After Raising, the increment of occurrence of both antecedent and consequent is $Inc_{ante\&con}$ ($Inc_{ante\&con} \geq 0$). However, there is also an increment of occurrence of interests in the antecedent Inc_{ante} ($Inc_{ante} \geq Inc_{ante\&con}$). Unfortunately, these two increments $Inc_{pre\&con}$ and Inc_{ante} do not have any relationship to each other. Thus,

$$S_{new} = S_0 + Incc_{ante\&con} \geq S_0$$
$$C_{new} = \frac{S_0 + Inc_{ante\&con}}{Occ_{ante} + Inc_{ante}}$$

Note the formulas are exactly the same as in **Group (C)**. Therefore Formula 5.1 can also be applied to the relationship between the confidence values before and after Raising for **Group (E)** and **Group (F)**. Thus the changes are also based on the fact of Raising-effected data.

If $C_{Inc} \geq C_0$ then $C_{new} \geq C_0$ and if $C_{Inc} < C_0$ then $C_{new} < C_0$.

According to the case analysis above, after Raising, the *support values* of all the association rules are *never decreased*, thus Raising guarantees higher or equal quality rules. For *confidence values*, the most important rule types for marketing purposes, Group (B), always have higher confidence values. This ensures high quality association rules with better support values and also better confidence values. The rule types in Groups (A), (C) and (D) are not proper for marketing purposes. Those rules are filtered out from the data mining results in a postprocessing step. The rule types in Groups (F) and (E) have increasing support values but undetermined changes of confidence values. However, as discussed before, those rule type are only used as complements for Group (B).

5.3 Computing Confidence and Support for Rules from Split Datasets

Due to the data mining algorithm principles, memory consumption is always a big concern in data mining algorithm implementations. Huge dataset will cause memory overflows. The size is not the only factor. The record patterns are also one important factor. Thus, how the items appear in each instance also greatly affects the algorithm. The more duplicated patterns appear among instances, the more memory is used. In other words, the less kinds of item sets (patterns) appear among all instances, the less memory is used. If most of the records have the same patterns, the usage of memory during the processing is not significant. More extremely, if all the instances are exactly the same, which corresponds to the least number of item sets (patterns), the operation uses the least amount of memory compared to the cases in which instances are different. The reason for this is that every

distinct pattern appears only once, with a count, even if it occurs many times in the data. However, distinct patterns must all occur in the representation.

To avoid a memory overflow, one of the simple methods is to "divide-and-conquer." Thus, the large dataset is split into several smaller datasets, which can be successfully processed by data mining algorithms without any memory problems. For each individual sub-dataset, rules can be retrieved by inputing the sub-dataset into the data mining algorithms for processing. For each rule, a confidence value and a support value can be found. However, these values are based merely on the records in the sub-dataset being processed. To get the values of this rule based on the whole dataset, some further computation is needed.

For example, there is a rule R, $\{SET_{ante}\} \rightarrow \{SET_{con}\}$ (SET_{ante} and SET_{con} are sets with at least one item, and $SET_{ante} \cap SET_{con} = \emptyset$), in the dataset D. The large dataset is divided into two small datasets: D_1 and D_2. Thus, the support value S_1 of R in the sub-dataset D_1 is equal to the occurrence of the records which contain SET_{ante} and SET_{con} at the same time, let's say $Occur_{ante\&con_1}$. The confidence value C_1 of R in the sub-dataset D_1 is the quotient of $Occur_{ante\&con_1}$ and the occurrence of the records which contains SET_{ante}, $Occur_{ante_1}$. Thus,

$$\begin{aligned} S_1 &= Occur_{ante\&con_1} \\ C_1 &= \frac{Occur_{ante\&con_1}}{Occur_{ante_1}} \end{aligned} \tag{5.2}$$

Accordingly, for the rule R in the sub-dataset D_2, there are

$$\begin{aligned} S_2 &= Occur_{ante\&con_2} \\ C_2 &= \frac{Occur_{ante\&con_2}}{Occur_{ante_2}} \end{aligned} \tag{5.3}$$

When the rule R is mined from the original dataset D, the support value is $Occur_{ante\&con}$, the occurrence of the records which contain SET_{ante} and SET_{con} at the same time in the whole dataset. Furthermore $Occur_{ante\&con} = Occur_{ante\&con_1} + Occur_{ante\&con_2}$ since the

occurence is divided into two parts for each sub-dataset. Thus,

$S = S_1 + S_2$

Also it holds that $Occur_{ante} = Occur_{ante_1} + Occur_{ante_2}$. Thus,

$$\begin{aligned} C &= \frac{Occur_{ante\&con}}{Occur_{ante}} \\ &= \frac{Occur_{ante\&con_1} + Occur_{ante\&con_2}}{Occur_{ante}} \\ &= \frac{C_1 \cdot Occur_{ante_1} + C_2 \cdot Occur_{ante_2}}{Occur_{ante}} \qquad \textit{by (5.2, 5.3)} \\ &= \frac{Occur_{ante_1}}{Occur_{ante}} \cdot C_1 + \frac{Occur_{ante_2}}{Occur_{ante}} \cdot C_2 \end{aligned}$$

This formula can also be extended if more than two sub-datasets are needed. Suppose n sub-datasets are present ($n > 1$) after splitting, thus,

$$\begin{aligned} S_i &= Occur_{ante\&con_i} (1 \leq i \leq n) \\ C_i &= \frac{Occur_{ante\&con_i}}{Occur_{ante_i}} (1 \leq i \leq n) \\ S &= \sum_{i=1}^{n} Occur_{ante\&con_i} \\ C &= \frac{Occur_{ante\&con}}{Occur_{ante}} \\ &= \frac{\sum_{i=1}^{n} Occur_{ante\&con_i}}{Occur_{ante}} \\ &= \frac{\sum_{i=1}^{n} C_i \cdot Occur_{ante_i}}{Occur_{ante}} \\ &= \sum_{i=1}^{n} \frac{Occur_{ante_i}}{Occur_{ante}} \cdot C_i \end{aligned} \tag{5.4}$$

Note that $\frac{Occur_{ante_i}}{Occur_{ante}}$ is the proportion rate of occurrence for sub-dataset i. Thus, the value C is equal to the summation of the products of the proportion rate of occurrence of each sub-dataset and the confidence value for that sub-dataset. In other words, it is the weighted average of all the sub-dataset confidence values weighted with the proportional occurrences of the antecedent. The confidence value of a sub-dataset with a large value of

occurrences of the antecedent more heavily influences the overall confidence value for the whole dataset.

In the Web marketing project, the whole dataset was split into 16 categories according to 16 top level interests in the ontology. The demographic and interest related rules were mined from each of the 16 sub-datasets. When a rule needs to be rebuilt, it is necessary to fetch all the information of that rule from every sub-dataset. Since the values $Occur_{ante_i}$ and C_i of rules are available, the combination by Formula (5.4) allows the total confidence to be computed.

One problem is that the threshold of the rules for the whole dataset cannot be directly applied to each sub-dataset. When a threshold, say 0.6 of a confidence value, is given, the threshold is applied to all the sub-datasets and the data mining algorithm retrieves all the rules satisfying the threshold from all the records in each individual sub-dataset. However, a rule from a sub-dataset D_i with confidence value of 0.7, which satisfies the threshold, is not necessarily qualified to be a rule for the whole dataset. According to the Formula (5.4), the confidence value over the whole dataset will change, depending on the number of occurrences of the antecedent in the other sub-datasets D_i.

CHAPTER 6

AN APPLICATION INTERSECTION MARKETING ONTOLOGY

6.1 Problem Description

6.1.1 Motivation

The result of market research is *marketing knowledge* that is used as input for target marketing activities. However, marketing knowledge is usually complex, consisting of many detailed facts, which by themselves do not give any clear picture and in combination are often overwhelming. What is desirable is an organization of marketing knowledge in an ontology that allows for the explicit representation of interesting abstractions and generalizations. However, the requirements for customer representation are different from those for interest representation.

In this chapter, a new kind of ontology, called an intersection ontology, is developed for customer representation and its advantages are explored.

6.1.2 Ontologies in the Context of Web Marketing

As described in Chapter 1, a large database of customers has been created. After extracting information from the home pages of individual Web users, the database contains demographic information and interests of each customer. The problem is how to organize the customers in a hierarchy.

Information about products that each of these customers has bought would be preferred. However, this information is not publicly accessible on the Web. On the other hand, there are many very low-level interests with corresponding products. For example, the Yahoo interest hierarchy contained over 31,000 interests when it was analyzed. Many of these interests are as specific as the names of actresses or singers. If somebody has an interest in "Jennifer Lopez" then one may comfortably presume that this person might buy CDs or

movies of Jennifer Lopez. Thus, information about interests can, to some degree, "stand in" for information about products.

As shown in Chapter 3 and Chapter 4, the Web Marketing Project has processed the relational database of demographic and interest information with the data mining algorithms and has found association rules between classifications of customers and interests. Thus, what is needed is an ontology that allows to represent the resulting association rules of a data mining operation in a *succinct* format.

Note that this research is not designing a marketing domain ontology which needs to represent all varied aspects of the marketing domain. An intersection ontology is being created as an integral part of a marketing system. Our application deals with customer classifications needed for a marketing ontology. Our ontology is, in Sowa's terms, an application ontology [57], serving the Web Marketing Project [7] described above. As such, the marketing ontology concentrates only on representing purchasing knowledge, as described in detail in Section 6.2.

The straightforward representation of a customer classification is a tree hierarchy. The root represents the concept PERSON. The various demographic dimensions are ordered. At each of the levels, one different demographic dimension according to the above order is considered and each node in the previous level branches into all possible options of this level's dimension. However, as it will be shown there are problems with this representation.

To overcome these problems, this research draws on Sowa's notion of representing conceptual knowledge using distinctions [58] and on Wille's use of intersections in Formal Concept Analysis [59, 60]. Due to the demands of the domain, realizing there is no natural order among the demographic dimensions and the need for an economical representation, an ontology that relies heavily on the use of "intersections" of concepts have been developed. To further economize, this ontology only contains those intersections about marketing knowledge that need to be represented. Thus, concepts are inserted into the ontology dynamically on demand. In an intersection hierarchy all the options of all dimensions

are children of PERSON. All the relevant customer classifications appear in the next level, each classification as a child of all its options. Such a representation is called a three-level intersection hierarchy. Finally a more economical solution is represented where the customer classifications can be distributed over several levels – the multi-level intersection hierarchy.

Section 6.2 discusses in more detail why an ontology for marketing knowledge is useful. In Section 6.3, the design of a customer hierarchy by ordered dimensions and the problems arising from it will be shown. Then, in Section 6.4, an alternative design for the customer hierarchy by creating "intersections" which results in an intersection ontology will be shown. In Section 6.5, the network design of a specific kind of intersection ontology is shown, called multi-level intersection ontology.

The evaluation based on the Web Marketing Project is described in Section 6.6. In Section 6.7, it is discussed how the marketing intersection ontology relates to Sowa's knowledge engineering by distinctions. The conclusions appear in Section 6.8.

6.2 Representation of Marketing Knowledge

The essence of a marketing ontology is a collection of buy-relationships from customer classifications to product classifications. The basic facts needed to be represented are of the form that a specific classification of customers tends to buy a given product or family of products. For example, "Married women with children buy toys." The challenge is to find a representation of this kind of knowledge in a convenient and economical way that fits into our ontology framework.

The marketing ontology needs to contain two hierarchies, a customer classification hierarchy, in short, *customer hierarchy*, and a product classification hierarchy, in short, *product hierarchy*. The group with the classification MARRIED WOMAN WITH CHILDREN needs to be identifiable in the customer hierarchy, either as a node or a group of nodes. To achieve the desired succinct representation, a single node for the customer classification

concept and a second single node for the product classification concept are preferred. Those two nodes are then connected by a single relationship link with the label "buys," which is an economical representation capturing the desired marketing knowledge for an ontology.

Figure 6.1 shows a tiny ontology excerpt of four nodes with three "buys" connections. The node WOMAN WITH CHILDREN and its child MARRIED WOMAN WITH CHILDREN belong to the customer hierarchy. The node TOY and its child DOLL belong to the product hierarchy. The three connections are labeled "buys." The "buys" relationship to TOYs is inherited from WOMAN WITH CHILDREN to MARRIED WOMAN WITH CHILDREN. The inherited relationship is a dashed arrow, usually not shown in diagrams, since it can be inferred.

On the other hand, if the customer classification is represented by k nodes ($k > 0$) and the product classification is represented by l nodes ($l > 0$), then up to $k * l$ "buys" relationships are needed to represent the proper marketing knowledge, which is less desirable. Figure 6.2 represents a tiny part of a customer hierarchy and a product hierarchy. In Figure 6.2, two nodes are needed to represent "men with children" or "electric toys." Thus, four arrows are needed to represent the fact that "men with children buy electric toys."

An alternative way with nodes representing "men with children" and "electric toys", respectively, with an arrow connecting them offers a more economical representation. However, if it is represented that ELECTRIC TOYS and NON-ELECTRIC TOYS are at level two and the distinction between OUTDOOR and INDOOR is at level three, then "men with children buy outdoor toys" will require four arrows. As will be discussed later, for each sequential ordering of the relevant dimensions, there are some marketing knowledge facts with an uneconomical representation.

The link with the label "buys" is used to mean "is likely to buy." Thus, "buys" is a statement strictly about a (meaningful) percentage of the population satisfying the demographic data.

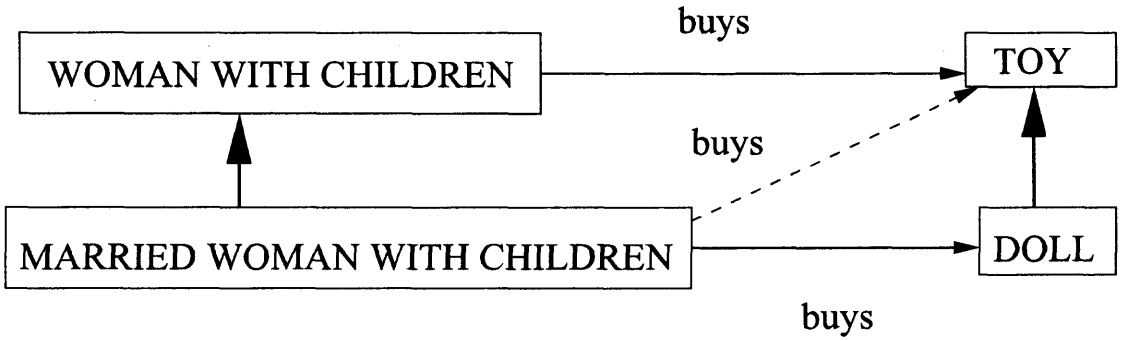


Figure 6.1 Extract of a marketing ontology.

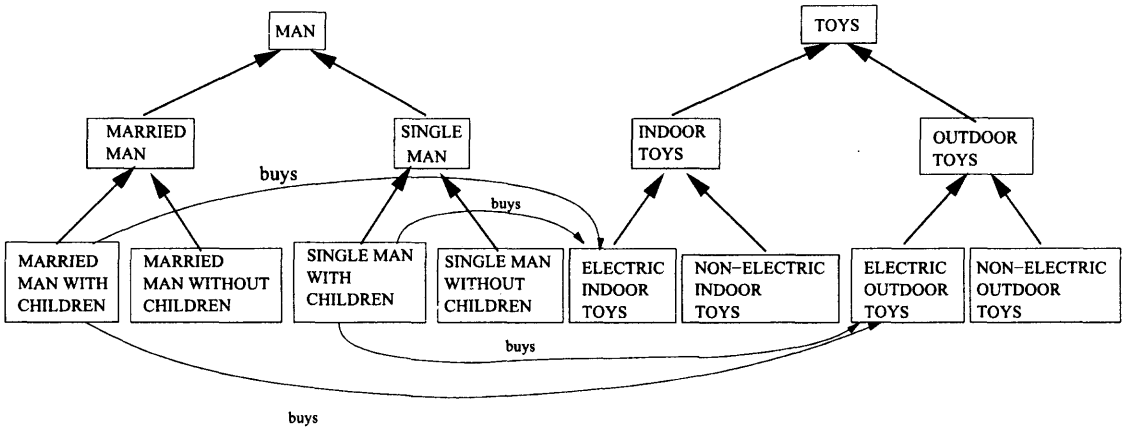


Figure 6.2 $k * l$ arrows needed to express a simple marketing fact.

For practical usability, a marketing knowledge representation should be as simple as possible. For example, if data mining finds that married men with children buy diapers, and married women with children buy diapers, then an assertion that married people with children buy diapers is better. Such information should be attached to exactly the concepts which the knowledge expressed is about. In this case, this knowledge should be associated with the concept *married people with children*, assuming such a concept *exists* in the ontology.

Finding a marketing ontology that enables the representation of *all the needed concepts* explicitly without creating a combinatorial explosion of concepts for customers is non-trivial. An intersection ontology achieves exactly this goal. However, first, the straightforward alternative, a tree representation with ordered dimensions, will be described and why it is inappropriate for marketing knowledge will be explained.

6.3 Customer Tree Hierarchy with Ordered Dimensions

Following customary practice in marketing, as used, for instance, by MediaMark [61], a classification of customers is performed along various dimensions such as gender (man, woman), age (five age groups), marital status (single, married, separated), children status (with children, no child), *etc.*

Marketing research may reveal knowledge about buying habits of a customer classified according to several dimensions simultaneously. For example, consider the sentence: "Middle-aged married men with children buy books on early childhood development." In the customer hierarchy a node which corresponds exactly to the above customer classification is needed.

Consider a tree hierarchy according to the four dimensions listed above, each dimension appearing at a different level of the hierarchy. The tree hierarchy starts with the root node

PERSON at level 1[1]. The division into the classifications MAN and WOMAN happens at level 2. The division of men (and of women) according to five age groups happens at level 3. There is an obvious redundancy, as the same age choices are made twice, once below MAN and once below WOMAN. The next two levels follow the distinction according to marital status among three options, and children status, respectively. Figure 6.3 shows such a tree hierarchy.

In this tree hierarchy, which is referred to as T, a linear order of the various dimensions of a customer is used. In other words, the different dimensions are prioritized. The above order of dimensions was working well for the above given example, because the customer class (middle-aged married men with children) is represented by a unique leaf node which is the source for the "buys" relationship to the node representing the product BOOKs ON EARLY CHILDHOOD DEVELOPMENT.

Some marketing knowledge should be attached at a single non-leaf node in the tree hierarchy T. For example, "Men buy football tickets" would be expressed by a relationship that has the second level node MAN as its source and FOOTBALL TICKET as its target.

In the last examples, the customer classification is represented as one node in T, from which one "buys" relationship link to a product node is emanating. In other situations, the description of a class of customers may not fit so neatly into the tree hierarchy T, as there might be a mismatch between this class and the order of dimensions in T. Consider, "People with children invest in Education IRAs." Even older people may have children, and people may also invest in IRAs for their grandchildren, so no age bracket applies here. To capture this class of customers, there is a need to refer to 30 leaf nodes in the tree hierarchy T, since the dimension considering children is at the lowest level in T. Furthermore, each of those nodes will require a "buys" relationship to an EDUCATION IRA node in the product hierarchy. The marketing knowledge "People with children invest in Education IRAs," expressed in a short sentence, requires 30 links in our marketing ontology. This is clearly

[1]Conceptually there is a common root "ENTITY" for the product hierarchy and the customer hierarchy at level 0. Thus PERSON is at level 1.

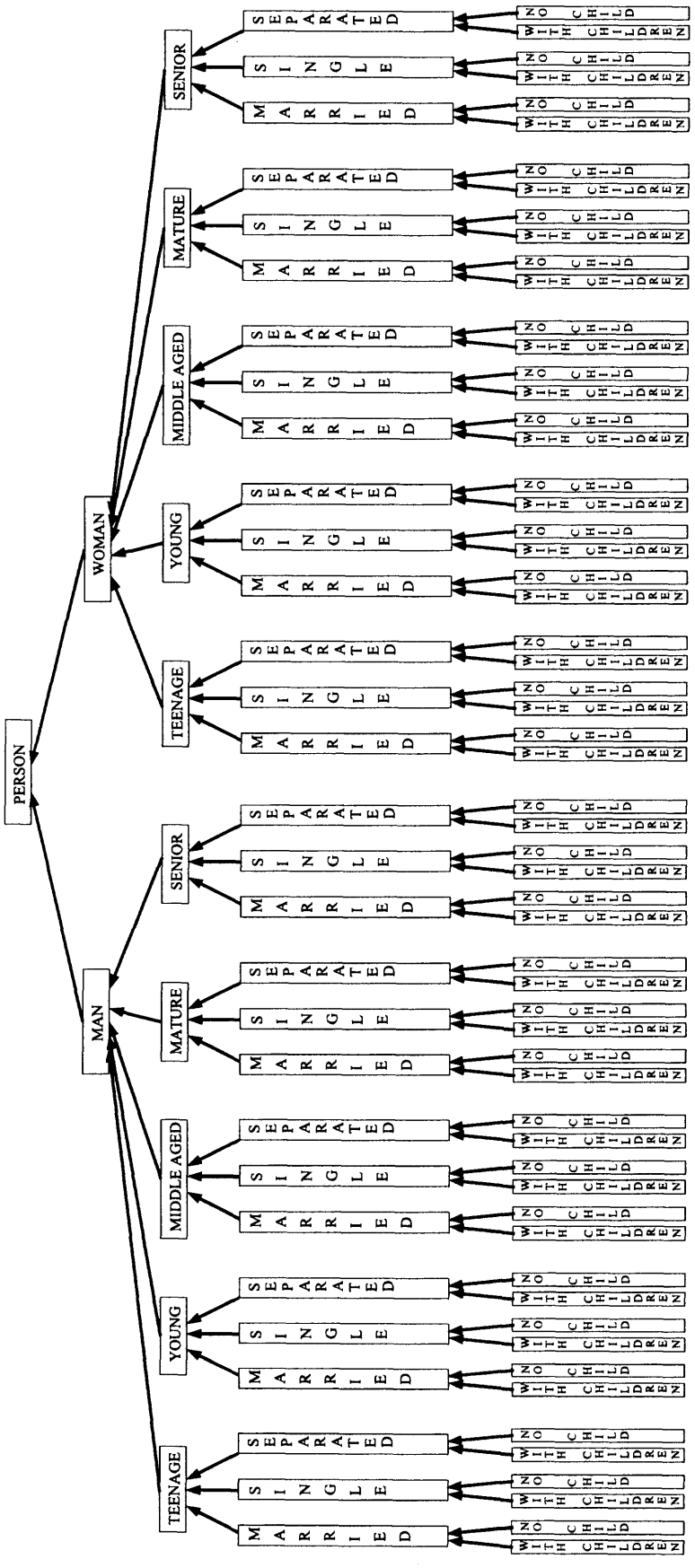


Figure 6.3 Marketing hierarchy with ordered dimensions.

an uneconomical representation of marketing knowledge. However, there is no inherent reason why it was chosen, for example, why the distinction between MAN and WOMAN is at the second level, above all the other dimensions. If, for example, the children status dimension would have been chosen as the top-level dimension in the hierarchy, then one node and one "buys" link would have been sufficient to represent this customer class and the associated marketing knowledge. Hence, for every ordering of the dimensions, the hierarchy will be well matched to some customer classes but ill fitting for others. Thus, a serious problem has been identified which may occur for *any* choice of ordering the dimensions, where many cases of marketing knowledge will require many links. The problem is inherent in the fact *that* an ordering is used.

Besides this problem of uneconomical representation of marketing knowledge, this straightforward representation has two secondary problems. One problem is the explosion of the total number of nodes. The number of just the leaves in T is the product of the numbers of options for all dimensions. In the tree hierarchy T of only four dimensions, each with few choices, there are 60 leaves. In the market research field, practitioners have identified many more dimensions. For example, the ten dimensions appearing in the MediaMark Web site [61] for customer classification are: Gender, Age, Household income, Education, Employment status/occupation, Race with region, Marital status, County size, Marketing region, and Household size. Since any combination of dimensions may appear in a customer classification, the tree hierarchy must be fully developed by expanding all dimensions. This need was demonstrated before with the example using the classification PERSON WITH CHILDREN.

The second problem with ordered dimensions is related to the explosion of nodes. Whole subtrees are repeated over and over. For example, the subtree with the marital choices is repeated for every age group. If a marketing executive decides to add a marital status "WIDOWED", then this update has to be performed in every subtree, leading to the well-known danger of inconsistencies (update anomalies).

6.4 Customer Intersection Hierarchy

The difficulties encountered in designing a tree hierarchy customer ontology stem from the fact that there is no preferred order of the various dimensions. Thus, a possible solution is to avoid prioritizing the dimensions. To solve this problem, Sowa's notion of representing conceptual knowledge using distinctions [58] is drawn on. Sowa claims, for example, that there is no order between the distinctions Concrete/Abstract and Object/Process. All four concepts: Concrete, Abstract, Object, and Process are children of Thing. A concept such as PhysicalObject is an intersection of the concepts Concrete and Object. The result of consistently applying such distinctions for all dimensions *on demand* is called an *intersection ontology* in this research. The significance of creating concepts for an ontology only on demand will be explained below.

Some dimensions without a natural priority between them may be encountered in the product hierarchy as well. Figure 6.2 demonstrates this situation between the location dimension (indoor, outdoor) and the operating mode dimension (electric, non-electric) of toys. Nevertheless, the situation in general is quite different from that of the customer hierarchy, where all dimensions are mutually independent. In the marketing field, there is a practice of considering some dimensions of a product classification prior to others. For example, Men's Wear and Women's Wear are typically in different departments and probably even on different floors of a department store. Each of these are further partitioned into various kinds of clothing, shoes, accessories *etc.* Furthermore, customers are used to this ordering of products and search accordingly for what they desire. Hence, while in the customer hierarchy, all dimensions are independent, some dimensions without natural priority between them exist for products. To handle these cases of independent dimensions for products, one could follow Sowa's [58] practice, where intersections appear only for these few mutually independent dimensions. In the balance of this chapter, the customer hierarchy will be concentrated on.

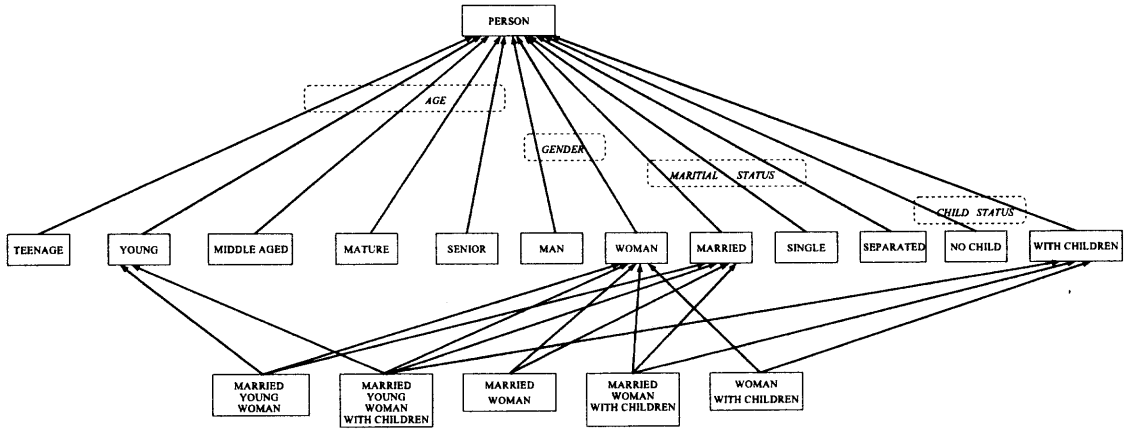


Figure 6.4 A sample customer intersection hierarchy with option nodes in level 2.

The customer intersection hierarchy designed in this research has a unique root node representing the concept PERSON at level 1. Each option of each dimension is now represented as a child of the root node at the second level of the hierarchy (see Figure 6.4). Such a node is called an *option node*. For example, in Figure 6.4, there are the WOMAN option node and the MARRIED option node.

The next question is how to represent a customer classification involving several dimensions. For example, the classification MARRIED WOMAN WITH CHILDREN involves three dimensions: gender, marital status and children status. The solution is to define in the hierarchy a new kind of node that represents a combination of several options, one option for each of several dimensions (Figure 6.4). For example, a MARRIED WOMAN node represents the combination of the option WOMAN for the gender dimension and the MARRIED option for the marital status dimension. Another node represents WOMAN WITH CHILDREN, a combination of options for gender and children status. The more complicated classification MARRIED WOMAN WITH CHILDREN represents a combination of options for three dimensions: WOMAN for gender, MARRIED for marital status and WITH CHILDREN for children status.

A node that represents a combination of options of various dimensions is called an *intersection node*, since it represents the classification of a set of customers which is

the mathematical intersection of several sets of customers, each with a one-dimensional classification. For example, the set of MARRIED WOMAN is the intersection of two sets MARRIED and WOMAN. Every intersection node is a child of each of the option nodes corresponding to the options involved in the intersection. For example, MARRIED WOMAN is a child of both MARRIED and WOMAN option nodes. Hence all intersection nodes appear in level 3 of the customer intersection hierarchy. Intersection classes of different kinds have appeared in various Object-Oriented Database (OODB) models of medical ontology representations [62, 63, 64, 65, 66, 67].

Note that the representation of Figure 6.4 is superior to the tree hierarchy representation of Section 6.3, where neither of the classifications mentioned above in this section corresponds to a single node. For instance, MARRIED WOMAN WITH CHILDREN needs to be represented by several nodes in the tree hierarchy T, because the AGE dimension is not mentioned in this classification. In T, AGE is the second dimension, and both MARRIED and WITH CHILDREN are below AGE in the hierarchy. Thus, to incorporate MARRIED, all AGE choices are included, too. As a result, five nodes of T are needed due to the five options of the AGE dimension. Each of these nodes will have a link to DOLL, to capture the marketing knowledge "Married women with children buy dolls," represented by one link in Figure 6.1. Hence T is not an economical representation of this marketing knowledge.

As another example, fifteen nodes are needed to represent WOMAN WITH CHILDREN in T. This number corresponds to the multiplication of the number of options for the AGE and MARITAL STATUS dimensions, both not mentioned in this classification. Again, 15 links will be needed to represent the marketing knowledge "Women with children buy toys," represented by one link in Figure 6.1.

The reason for this large number of nodes of T for a classification is that in the tree hierarchy, for each dimension of the classification, the number of relevant nodes is multiplied by the number of options for this dimension. This is because at each level the classification of each node of the previous level is further subdivided into nodes according

to the options considered at this level. Thus, the representation in Figure 6.4, using intersection nodes, has the advantage that each classification (selecting one option for each dimension), independent of the number of dimensions involved, and independent of the number of their options, is represented by a single node. Hence, each "buys" link, starting at a customer class that is described by an intersection node, has a unique source. In contrast, in the customer tree hierarchy T, it is typical to require several nodes with "buys" relationships for such a classification.

Option nodes may have attributes and relationships. Intersection nodes inherit these properties from all their parents, enabling multiple inheritance of properties. The root node and option nodes may also be sources in "buys" relationships.

At first glance it might appear that, with intersection nodes, hierarchies generated are even larger than with ordered dimensions, as there are a large number of nodes already at the second level. However, the opposite is the case. A crucial aspect of this definition of intersection ontologies is that concepts below the second level are only created on demand. That is, *only* nodes which represent a combination of dimensions needed for the marketing knowledge in the database are represented in the hierarchy. If no marketing knowledge about a specific combination of dimensions exists, then an intersection node for this combination is not created!

More specifically, if a specific group of customers is not needed from the database as the source of a "buys" relationship then there is no need to create the corresponding concept node. Thus, if there is no marketing knowledge available in the database about a single man of Alaskan ethnic origin over seventy, then a corresponding general concept will not be created in the ontology. Intersection nodes are created only on demand if the need for them arises. Traditional general ontologies and domain ontologies typically attempt to represent everything that may exist. For our marketing application, this would result in an explosion of concepts. With the intersection hierarchy, the explosion of nodes is controlled. Only concepts that are needed are created. In the ordered dimension representation, a

node which is not a leaf cannot be omitted from the tree hierarchy, even if no marketing knowledge is available regarding this node, since marketing knowledge may exist about any of its descendants.

Definition: The size of an ontology is a pair (a, b) where a is the number of nodes and b is the number of relationships.

For instance, the size of the ontology of Figure 6.4 is (18, 26). This definition will be used in Section 6.5 below.

6.5 Multi-level Intersection Hierarchy

Now the network connecting all the nodes in the customer intersection hierarchy will be explicitly considered. First the network of the intersection hierarchy will be described formally, informally described in the previous section. This network will be denoted "the three-level intersection hierarchy." The following discussion will show that the three-level intersection hierarchy is not a proper representation. Then an alternative network will be introduced, the multi-level intersection hierarchy, overcoming the deficiencies of the three-level intersection hierarchy.

Consider an intersection node which represents the concept of a combination of k options O_{i_1}, O_{i_2},..., O_{i_k}, one for each of the corresponding k dimensions ($k \leq n$) of the n existing dimensions. Such a concept (node) is more specific than (a child of) each of the option concepts (nodes) which represents one of the options O_{i_j}, $1 \leq j \leq k$, since the set of customers which satisfy all the options O_{i_1}, O_{i_2},..., O_{i_k} simultaneously, is a subset of each of the customer sets which satisfies one option O_{i_j}, where $1 \leq j \leq k$.

In the three-level intersection hierarchy, each intersection node is at the third level, since all its k option parents are at the second level. Hence, the name of this network. (See Figure 6.4 for a sample of a three-level customer hierarchy.)

The three-level intersection hierarchy is improper, as will be explained now. In the three-level intersection hierarchy, only IS-A relationships between intersection nodes

and option nodes are presented. Consider two specific intersection nodes in Figure 6.4, MARRIED WOMAN and MARRIED WOMAN WITH CHILDREN. The second classification is more specialized than the first classification, since the set of customers, classified by MARRIED WOMAN WITH CHILDREN, is a subset of the set of customers classified by MARRIED WOMAN. To express this specialization, the intersection node MARRIED WOMAN WITH CHILDREN should have as a parent the intersection node MARRIED WOMAN.

In the three-level intersection hierarchy in Figure 6.4, the node MARRIED WOMAN WITH CHILDREN has three parents: WOMAN, MARRIED and WITH CHILDREN. Should those parent relationships also exist after adding the parent MARRIED WOMAN? The node MARRIED WOMAN itself has as parents the option nodes WOMAN and MARRIED. A relationship from MARRIED WOMAN WITH CHILDREN to WOMAN (or to MARRIED) is implied by the transitivity of the IS-A relationship.

Thus, it is concluded that the three-level representation does not fulfill all requirements for a proper representation, because it does not capture the specialization which exists between intersection nodes. An alternative representation is now introduced, which is the more refined *multi-level intersection hierarchy* that allows expressing parent-child relationships between two intersection nodes, when one represents a more specific concept than the other. For a multi-level hierarchy representation of the nodes of Figure 6.4, see Figure 6.5.

In Figure 6.5, the node MARRIED WOMAN WITH CHILDREN has no parent relationship to the option nodes. On the other hand, the node MARRIED YOUNG WOMAN has a parent relationship to the option node YOUNG since the hierarchy contains neither the node YOUNG WOMAN nor the node YOUNG MARRIED which would have been parents of MARRIED YOUNG WOMAN and would have implied, as an intermediate node, the IS-A relationship to the option node YOUNG by transitivity.

Note that Figure 6.5 has 5 levels. The number of explicit parent relationships in Figure 6.5 is 22 versus 26 such relationships in Figure 6.4. Both figures have 18 nodes.

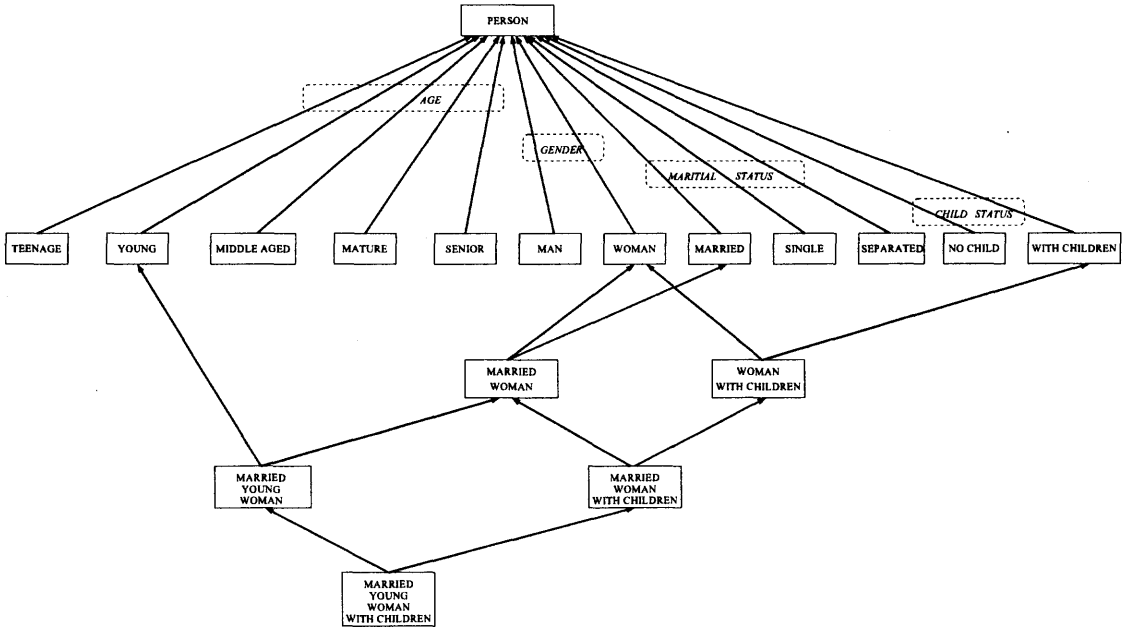


Figure 6.5 A sample multi-Level customer hierarchy.

The definition for visual complexity of a diagram was introduced and used in [68,69, 70]. It will now be modified for use with ontologies.

Definition: The *visual complexity* C of an ontology of size (a, b) is the ratio of the number of relationships (= links) to the number of nodes, $C = b/a$.

Hence the three-level intersection ontology of Figure 6.4 has size (18, 26) and visual complexity $C = 26/18 = 1.44$. On the other hand, the multi-level intersection ontology of Figure 6.5 has size (18, 22) and visual complexity $C = 22/18 = 1.22$. In this example, the multi-level ontology has lower size and lower visual complexity in comparison with the corresponding three-level ontology. Note that this visual complexity measure is a global measure for an ontology, compared to the notions of "tangled" and "sparse" used in [71] to measure local properties of the top level hierarchy.

To summarize, the three-level intersection hierarchy representation is not proper, because it does not capture IS-A relationships between intersection nodes. Such relationships are captured by the multi-level intersection hierarchy which also has other advantages, as follows.

1. The multi-level representation allows to use inheritance between intersection nodes, which is not possible in the three-level hierarchy. For example, if it is known that women with children buy toys, this fact is inherited to married women with children. In this way, the multi-level representation maintains one of the major advantages of ontologies, the economy brought about by inheritance-based reasoning.

2. The distribution of intersection nodes over several levels, due to the additional specialization IS-A relationship between such nodes, simplifies orientation of the user in such a hierarchy.

3. The number of explicit parent relationships is typically smaller than in the three-level intersection hierarchy. (This is not necessarily true, as one can intentionally design a counterexample.) This makes the multi-level intersection hierarchy diagram smaller in *size* and lower in *visual complexity* than the equivalent three-level intersection hierarchy.

6.6 Evaluation

The customer ontology of the Web Marketing Project is used to evaluate the design of the multi-level ontology versus the other designs. In the Web Marketing Project, 301,109 valid records of person's information have been collected. A record of information is considered valid when it has a valid email address and at least one expressed interest. Some of the information is expressed in foreign characters, which have been ignored. After filtering, there are 274,665 records left. However, most people also provide more information such as their age, gender and marital status. Regarding these as three dimensions for PERSON, the customer ontology was constructed for this project and the visual complexity of the *ordered dimensions* tree hierarchy, the *three-level intersection* hierarchy, and the *multi-level intersection* hierarchy representation will be compared.

The dimensions of AGE, GENDER and MARITAL STATUS have 6, 2 and 6 options respectively. Each personal record is represented as an instance of a corresponding classification (node) in the ontology. However, some nodes only contain fewer than 100 records of real people. For marketing purposes, such nodes, which do not represent useful information, are ignored.

Using the design of *ordered dimensions*, the result is the ontology shown in Figure 6.6(a). The blank boxes stand for nodes without enough instances, and are *not* created. In this figure, each node represents a meaningful customer classification from a marketing point of view, because of the corresponding number of persons in the database. For instance, there are 23709 records for those who are males, whose ages are between 10 and 19, whose marital status is not specified.

The tree hierarchy in Figure 6.6(a) has 62 nodes and 61 IS-A links and the visual complexity of 0.98. However, using this hierarchy, when trying to represent all the customer concepts with marketing knowledge, some of the concepts are not represented by a single node. To represent such a concept, multiple nodes, distributed in different parts of the hierarchy of Figure 6.6(a), have to be collected. For example, due to the order of the dimensions, to represent the concept AGE 20-29, 11 nodes, structured in two subtrees in Figure 6.6(a), are needed, as shown in Figure 6.7(a). Moreover, to represent the concept MALE and DIVORCED, four nodes need to be collected, as shown in Figure 6.7(b).

The number of possible concepts with one dimension is $2+6+6 = 14$ and with two dimensions is $2 \times 6 + 2 \times 6 + 6 \times 6 = 60$. Hence the number of possible concepts with one or two dimensions is 74. The concepts with three dimensions are not considered here, since they are properly represented in Figure 6.6(a) by a single node leaf. Among those 74 concepts, 14 can be found, in levels 2 and three in Figure 6.6(a), as corresponding single nodes. Since 48 of them do not have enough instances, there are $74-14-48 = 12$ concepts which are not represented by a single node. Figure 6.6(b) summarizes those 12 concepts needed in addition to Figure 6.6(a) to represent every needed marketing knowledge concept.

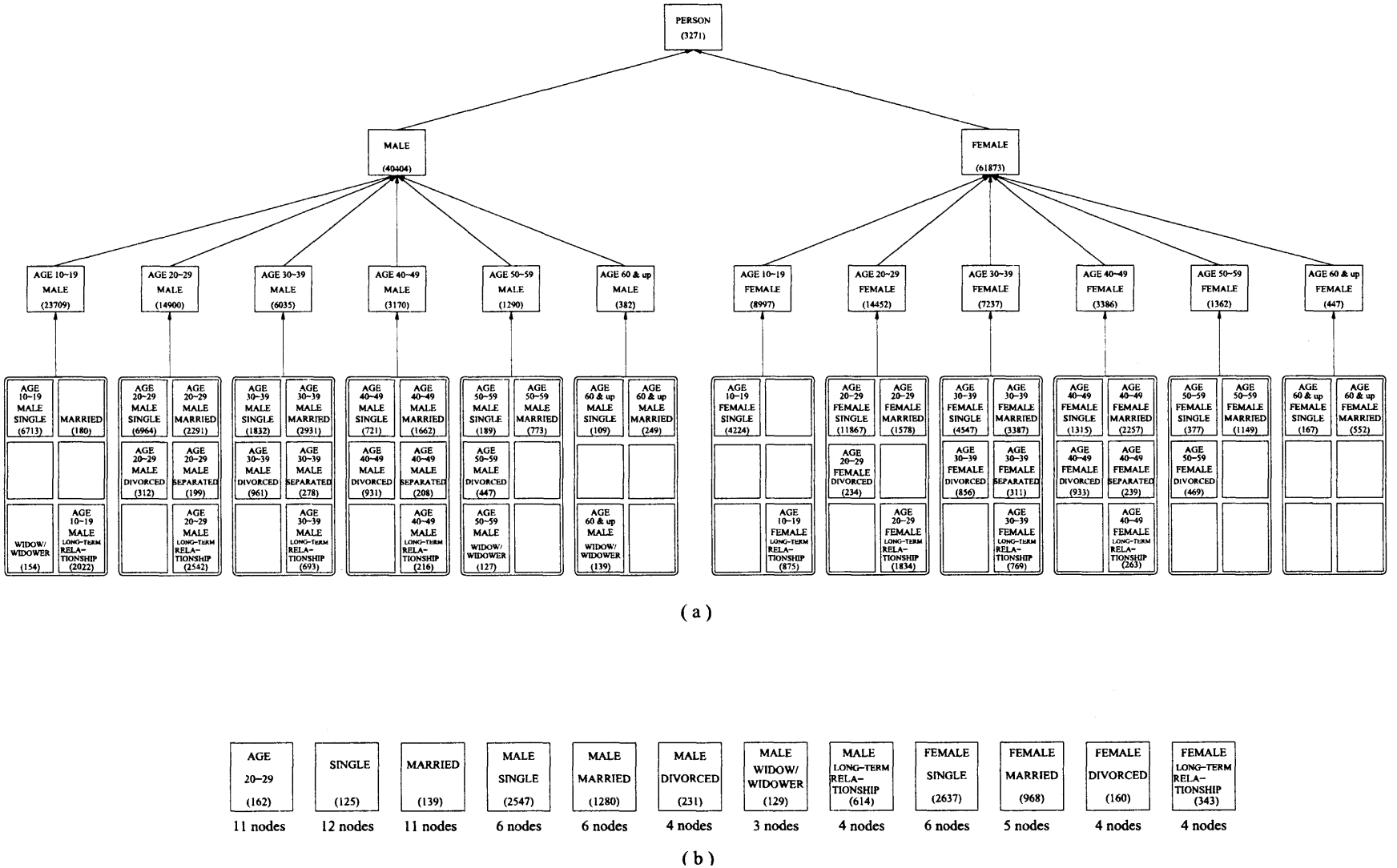


Figure 6.6 Marketing hierarchy with ordered dimensions for evaluation.

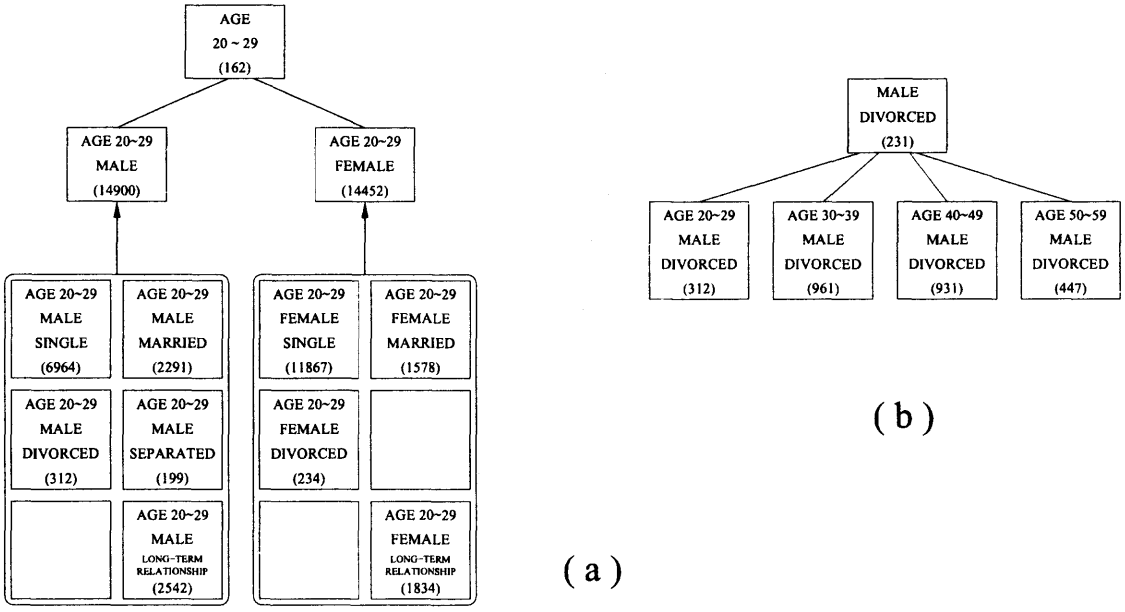


Figure 6.7 The nodes collection samples in the Figure 6.6(b).

Every one of these 12 concepts needs to be represented by a group of nodes, distributed in various parts of Figure 6.6(a), shown as its children as in the Figure 6.7. For each concept in Figure 6.6(b), the number of these nodes is listed, adding up to 76 nodes. Note that Figures 6.7(a) and 6.7(b) show only the expansions of the first node and the sixth node in Figure 6.6(b), respectively. Thus, the number of nodes representing all the relevant concepts in the customer tree hierarchy is $62 + 76 = 138$.

The design of the *multi-level intersection* hierarchy appears as the ontology hierarchy in Figure 6.8. There are 14 option nodes. The third level has 21 intersection nodes, each of which has two IS-A links pointing to option nodes. The fourth level has 47 intersection nodes combining three dimensions. Out of 72 possible intersection nodes, 25 contain fewer than 100 records and are not represented. Thus, this design has $1+14+21+(72-25) = 83$ nodes and 150 IS-A links. The visual complexity of the *multi-level intersection* hierarchy is $150/83 = 1.81$.

For the *three-level intersection* hierarchy, the figure is too large to be shown here. However, the figure is a modification of Figure 6.8 for the *multi-level intersection* hierarchy. The only difference, is that all the 47 nodes in the fourth level are moved to level 3 and are

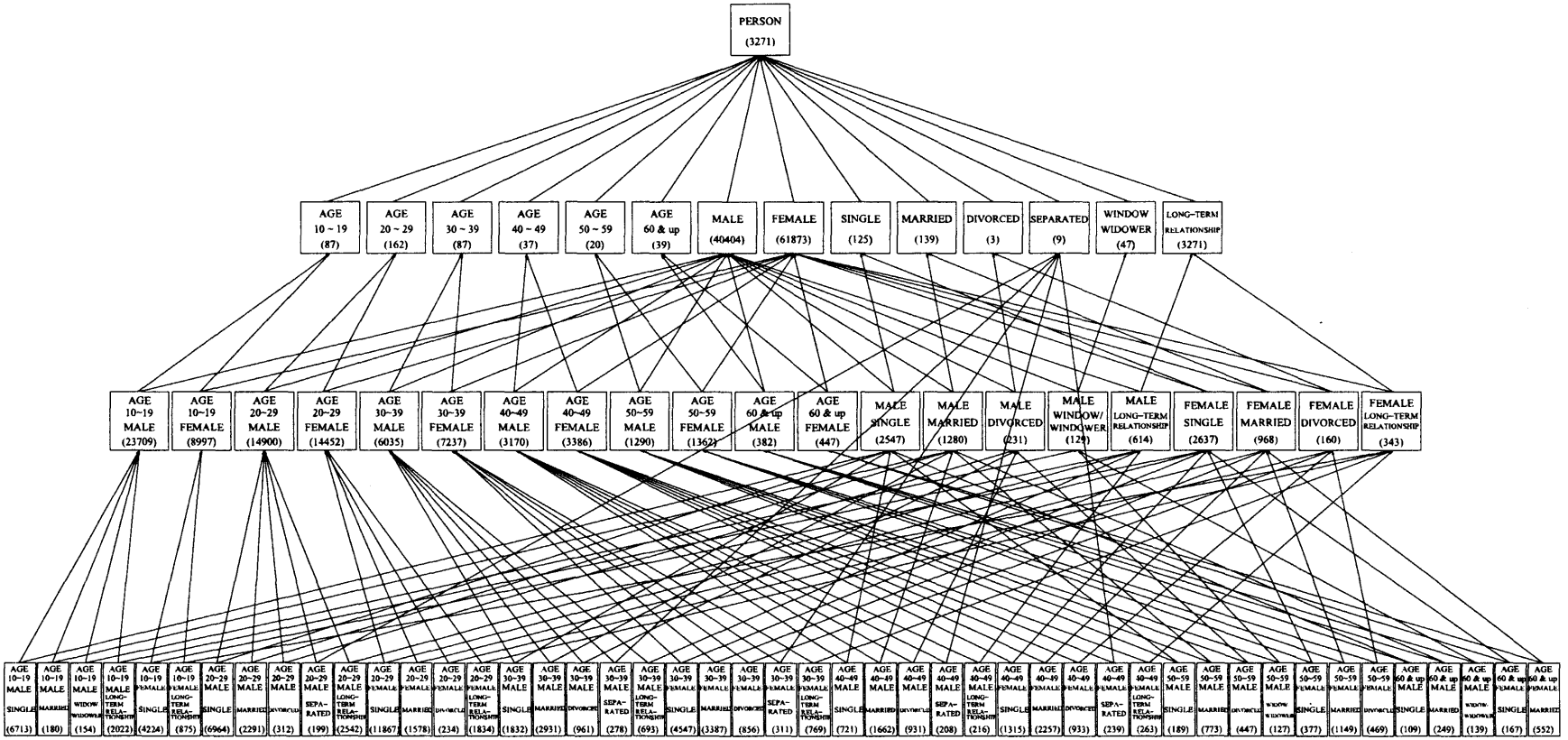


Figure 6.8 Marketing multi-level customer hierarchy.

directly connected to the option nodes. Thus, there are 68 intersection nodes at level 3. The total number of nodes is again 83, but the number of IS-A links is 197. The extra 47 IS-A links occur since each of the 47 nodes has 3 IS-A links. The visual complexity is $197/83 = 2.37$.

In summary, the usage of intersection nodes insures that every relevant customer concept is represented by one single node in the hierarchy. The three-level and multi-level intersection hierarchy have the same number of nodes. However, the *multi-level intersection* hierarchy has fewer links and lower visual complexity than the *three-level intersection* hierarchy.

In the design of the customer tree hierarchy (see Figure 6.6), there are 138 nodes. Comparing with the 83 nodes in the other two designs, the size is 66% bigger. This design has a lower visual complexity, since it is a forest of 13 trees. However, the measurement of visual complexity is secondary to the size, in this case.

6.7 Discussion

In principle, the mission of a general purpose ontology is to represent the real world and facilitate the exchange of information between heterogeneous systems. In this view, one would need to create every single intersection in the customer hierarchy, whether there is additional information about it or not, simply because it may be needed for information exchange. However, in our application ontology this would lead to an unreasonably large structure. As a matter of fact, Sowa [57], page 53, writes that "limited ontologies will always be useful for single applications in highly specialized domains. But to share knowledge with other applications, an ontology must be embedded within a more general framework." As a single application ontology, our marketing ontology has to serve its application as well as possible. Thus, it should be an economical representation that includes only nodes for which marketing knowledge is relevant. Also, for this specific application, there is no

order whatsoever between any pair of dimensions. Thus, intersections would need to be applied universally. This is not necessarily true for other application ontologies.

Thus, information that may be "inferred" is omitted from the ontology. The principle of creating an ontology that is an economical representation goes back to the first semantic networks [12]. A major reason for storing attributes at a concept high up in a semantic network hierarchy was to eliminate duplication of information. Whenever any such attribute was needed at a lower level, it was inherited down.

The major difference in this case is that attributes at lower levels are not omitted, but the lower levels are altogether omitted. Instead of using inheritance to instantiate the representation whenever needed, the on demand creation of new intersection concepts which are children of two or more existing concepts is used. Similarly, [72] writes about dynamic additions in an ontology: "Note that according to this definition, an ontology includes not only the terms that are explicitly defined in it, but also terms that can be inferred using rules." Thus, one could view our approach as implementing a global inference rule which is triggered by existing data and infers new concepts.

One problem which exists in the design of ontologies is how to forbid the representation of an impossible combination. In the intersection ontology design, this translates into the question how to forbid impossible intersections. For example, an intersection node TEENAGE MARRIED WOMAN should not be represented since it is illegal for a teenager to get married.

How can impossible combinations be prevented in the intersection ontology? Note that the ontology's intersections are created on demand, based on available marketing knowledge. There should be no such impossible combinations in the available marketing knowledge, and thus no such intersection should be created. If, however, such an intersection is created, it comes from erroneous data and can be used for auditing errors in the given marketing knowledge, as what was done in [66].

6.8 Conclusions

An application-oriented ontology for marketing knowledge has been introduced based on the introduction of intersection concepts on demand. Instead of imposing an order on the classification dimensions which is satisfactory for some purposes (and users) but not for others, ordered dimensions are completely eliminated. Instead, intersections of options for the various dimensions are consistently used.

The development of an application ontology for customer classifications in a marketing knowledge base was described. This ontology needed to conform to a number of requirements. First and foremost, it was required to make it easy to represent in the ontology, knowledge about likely buying behavior of classes of customers by a single (or a few) links from customer concepts to product concepts.

The intersection ontology representation satisfies this goal because it allows the representation of "buys" relationships by single links whenever this is warranted by the marketing knowledge. Yet, this representation does not produce a combinatorial explosion of all possible intersection nodes. Rather, it only represents the concepts for customer classes which are necessary as sources for known "buys" relationships.

In the multi-level intersection ontology representation, intersection nodes of many option nodes may be placed at various levels. These nodes may have IS-A links to other intersection nodes as well as to option nodes. These IS-A links may be used for property inheritance, as in other concept hierarchies.

The multi-level hierarchy representation fulfills a secondary requirement for ontologies, namely that they can be represented by diagrams of relatively low size and visual complexity. The multi-level hierarchy representation is typically of lower visual complexity relative to the three-level intersection hierarchy. As described in Section 6.6, in the evaluation based on the Web Marketing Project, the multi-level representation has a 24% lower visual complexity than the three-level representation. Moreover, its size is 40% smaller than the size of the ordered dimensions representation. In conclusion, it was showed that for

the marketing domain an economical intersection ontology may be created by inserting intersections of options of the various classification dimensions on demand. Such a representation may also be proper for other applications.

CHAPTER 7

IMPLEMENTATION

7.1 Problem Description

Implementations are needed to test theoretical designs with real data. Work has been done in the Web Marketing Project group to build a system that implements the theory of the previous chapters. It had to be decided where to build and how to build the system. More specifically, what operation system should the project be built on, Unix or Windows? What languages and tool packages should be used to benefit the overall building process? In this chapter implementation choices will be presented.

7.2 Environment, Languages and Packages

This research was implemented in a UNIX environment, more specifically, on Sun Solaris. The database used is Oracle. During the progression of the research, the Oracle database had been updated from Version 8 to 9i and the systems group had moved its server from *limpid.njit.edu* to *seer.njit.edu*. Several languages and packages were used in the Web Marketing Project. Some of them were used only for one module in the project. Table 7.1 describes the environment and important languages and packages used for implementation. Two versions of Java are listed because of the introduction of *generics*, which will be discussed later.

7.2.1 Languages

Different programming languages were used in the Web Marketing Project to build the whole system. Not all the languages are discussed here, such as SQL used in Oracle and HTML and JavaScript used on Web pages, as these are widely known. The other languages are described now.

Table 7.1 Research environment, languages and packages used

Operating System:	Solaris 9
Database	Oracle 9i
Languages	SQL HTML JavaScript Perl HP WebL Java 1.4 & Java 1.5
Packages	Milonic DHTML Menu JESS WEKA

The major task for CGI (Common Gateway Interface) is to get arguments submitted by users in Web forms and to pass them accordingly to a Java program for computation and database access. For example, the file "frontend.cgi" is shown below:

```
#!/bin/sh
#source /opt/bin/coraenv
#source/export/home/h2/challeng/.cshrc
java -classpath \
.:/afs/cad.njit.edu/research/p/2/public_html/Frontend/classes12.zip \
-Dcgi.request_method=$REQUEST_METHOD \
-Dcgi.query_string=$QUERY_STRING \
-Dcgi.content_type=$CONTENT_TYPE \
-Dcgi.content_length=$CONTENT_LENGTH\
-Dcgi.server_name=$SERVER_NAME \
-Dcgi.server_port=$SERVER_PORT \
-Dcgi.script_name=$SCRIPT_NAME  \
frontend
```

The information submitted by users is stored in the CGI variable "QUERY_STRING" by using the GET method of HTML forms. It is passed into the Java program "frontend.class"

for parsing and processing. The results are then returned to the Web browser in the form of Web pages. The "/bin/sh" in the first line indicates that this program is a shell script.

The programming language Perl was introduced to the project for more complicated tasks and more powerful CGI files. One main reason is that there are length restrictions for the "QUERY_STRING," usually 256 bytes/characters. Thus, for long strings which may arise from the body of an email or an automatically generated email list, the arguments have to be passed by using the POST method and have to be read from the variable "STDIN" with the string length stored in the variable "CONTENT_LENGTH" such as:

```
#!/usr/local/bin/perl
 ...
 read(STDIN, $buffer, $ENV{'CONTENT_LENGTH'});
 ...
```

The first line of the script, "/usr/local/bin/perl", indicates that it is a Perl script. In the Web page http://web.njit.edu/challeng/feedbackform.html, a Perl script was used in "feedback.cgi" to send a feedback email directly to the project team using the UNIX "sendmail" command. In the same program, a simple Thank You HTML page is displayed after sending the email,.

In the result Web page from http://web.njit.edu/challeng/top.html, "emailbcst.cgi" and "send.cgi" were also written in Perl. Each of them uses a conditional statement to find out whether the arguments are from the POST method or the GET method and then read the arguments at the corresponding places, "STDIN" or "QUERY_STRING." The contents read are then passed into the Java program and finally an email message to a long list of email addresses is sent out.

HP's WebL [2] has been briefly introduced in Chapter 1. HP's Web Language and system was designed for rapid prototyping of Web page processing. It is well-suited for the automation of tasks on the WWW. Its emphasis is on high flexibility and high-level abstractions rather than raw computation speed. It is thus better suited as a rapid

prototyping tool than a high-volume production tool. It is implemented as a stand-alone application that fetches and processes Web pages according to programmed scripts. HP's Web Language is written nearly completely in Java. It is very easy to add bridges from HP's Web Language to Java code. Java objects can be called directly from HP's Web Language code without extending HP's Web Language system. HP's Web Language is a high level, imperative, interpreted, dynamically typed, multi-threaded, expression, language. It has standard data types including boolean, character, integer (64-bit), double precision floats, Unicode strings, lists, sets, associative arrays (objects), functions, and methods. It also has special data types for processing HTML/XML that include pages, pieces (for markup elements), piece sets, and tags. HP's Web Language uses conventional control structures like if-then-else, while-do, repeat-until, try-catch, *etc.* and has a clean, easy to read syntax with C-like expressions and Modula-like control structures.

In the Web search module, HP's WebL was used to traversal Web pages in the Yahoo members directory. It WebL supports HTTP protocol and supports a markup algebra for extracting elements and text from pages, and functions for manipulating the content of a page. Thus, once the structure of a Web page is fully analyzed, it is convenient to access the text elements which are needed to be extracted, by referring to the exact locations. For example, the following line assigns all bold hyperlinks in a table to the WebL variable "y."

```
var y = Elem(U,"a") contain Elem(U,"b") inside Elem(U,"table")
```

Recursion is also used in a WebL program. To traverse from page to page, a starting page was selected. All the necessary links were then collected and visited recursively. Thus, in the *interest hierarchy collection WebL program*, the starting URL for execution was "http://members.yahoo.com," as shown in the partial program below.

```
import Str;
import Java;
...
```

```
var trace=fun(url)
   var P=GetURL(url);
   var A=Elem(P,"a") contain Elem(P,"b");
   if level > MaxLevel then
        MaxLevel = level;
   end;
   level=level+1;
   every a in A do
        var yahooID="0";
        var m=Str_Match(a.href,
"(?i)http://members.yahoo.com/interests/(.+)");
        if m!=nil then
           var list = Str_Split(ToString(m[1]), "-");
           yahooID = list[1];
        else
           yahooID = ToString(2000000000+ID);
        end;

        var TorF=Str_StartsWith(fixedtext, "@");
        if TorF==true then
           var list = Str_Split(fixedtext, "@");
           fixedtext = list[0];
        end;
        ...
        ID=ID+1;
        if TorF != true then
           trace(a.href);
        end;
   end;
   level = level-1;
end;

//Starting the traversal here
var url ="http://members.yahoo.com";
trace(url);
```

In the above program, the last two lines include a starting URL and start the traversal. The "trace()" method is defined at the beginning to extract the Yahoo ID of the currently visited page and to collect all the bold hyperlinks for recursively accessing them later. The "str" package is loaded in the first line and some string processing methods are used in the body of the program to perform string matching.

Even though what can be done with WebL can be also done by Java programming, the feature of Web page parsing makes WebL a faster and an easier option. Java was used for all the other coding of the implementation.

Two versions of JAVA were used, Java 1.4 and Java 1.5. In Java 1.5, the concept of *generics* was introduced. In earlier versions of Java, when taking an element out of a Collection, it must be cast to the type of element that is stored in the collection. Besides being inconvenient, this is unsafe. The compiler does not check that your cast is the same as the collection's type, so the cast can fail at run time. In Java 1.5, *generics* provide a way to communicate the type of a collection to the compiler, so that it can be checked. Once the compiler knows the element type of the collection, the compiler can check that you have used the collection consistently and can insert the correct casts on values being taken out of the collection. Thus, all the Java programs had to be modified to apply *generics*. For example, the usage of the variable "FPPattern" was modified from

```
private ArrayList FPpattern = new ArrayList();
...
FPpattern.add((TreeSet)(((ArrayList)(returnedpattern.get(1))).get(i)));
```

to

```
private ArrayList<TreeSet<String>> FPpattern = new
ArrayList<TreeSet<String>>(); ...
FPpattern.add((returnedpattern.get(1)).get(i));
```

Thoug, the initialization of the variable FPpattern at the beginning is a little more complicated after the modification, the clarification of the collection's type makes the usages of the variable much safer. In most case, such modifications also simplified long lines of Java code.

7.2.2 Packages

Milonic DHTML Menu The dynamic pop-up menu in the Web Marketing Project is a Milonic DHTML Menu [73], written in JavaScript. As shown in Figure 1.5, the pop-up menu is a Milonic DHTML Menu. There were some modifications made to the menu system for better navigation, such as the up and down arrows when the menu is longer than the height of the screen. The content of the interest menu is generated automatically from the database by following the stored ontology hierarchy. Moreover, though only the interest names are shown in the menu, the Yahoo IDs of all interests are also listed in the menu invisibly at hidden values. There are options on the Web page for users to choose whether to search by interest names or Yahoo IDs.

JESS JESS stands for Java Expert System Shell. It is a rule engine and scripting environment written entirely in Sun's Java language by Ernest Friedman-Hill at Sandia National Laboratories in Livermore, CA [74]. In the Web Marketing project, JESS was used in the Glossary module for free text analysis of Web pages extracted from university Web sites.

WEKA WEKA is a collection of machine learning algorithms for data mining tasks, created at the University of Waikato, New Zealand. The algorithms can either be applied directly to a dataset or called from your own Java code. WEKA contains tools for data pre-processing, classification, regression, clustering, association rules, and visualization [75]. WEKA is also implemented in Java. It has both a GUI interface and a command line interface. In this research, only the association rules tool were used to help to generate marketing knowledge, as described in Chapter 3. For the interaction with other programs, command lines were mostly used. An example of the typical usage of an association rule generating is as:

```
java weka.associations.Apriori -N 100 -C 0.1 -D 0.01
-M 0.005 -t families_age_gender_level5.nominal.arff
```

```
> families_age_gender_level5_rules
```

In this example, the Apriori module in the WEKA package is called to derived rules from an input file named "families_age_gender_level5.nominal.arff," which is in the ARFF format. The outputs will be redirected into a file named "families_age_gender_level5_rules." Among the arguments, the two most important ones are "-C" and "-D." The key "-C" refers to the threshold of the confidence value. The key "-D" refers to the threshold of the support value. As described in Section 1.4, the support value here is presented in the format of a relative value, not an absolute value. Thus, in this example, all the rules derived will be with a confidence value which is greater or equal to 0.1 and with a relative support value which is greater or equal to 0.01.

7.3 Tools and Modules

This research was implemented as several tools and modules, applying object-oriented and modular design principles. Most of the individual programs were written as methods of objects, which can either be executed alone or can be called by other programs. Some of the important tools and modules are described below:

IDSearch As described in Chapter 2, each interest name has its corresponding Yahoo ID. Yahoo IDs are the identifiers of interests and are used in most of the implementation, such as Raising and rule generation. However, a 10-digit ID is hard to understand and to remember by humans, who will prefer a more explicit interest name. Thus, the tool *IDSearch* was written to map an interest name to a Yahoo ID or several Yahoo IDs, if that interest name is used by more than one Yahoo ID[1]. When a user wants to find the IDs of an interest name, all the IDs associated with the input will be listed as the "Exact Matches."

[1]For example, the interest "TEXAS" has different IDs when refers to the Greedy Associations in Texas (1600730409) and to Skateboarding in Texas (1602749611).

Moreover, other interest names which have the input as a substring will also be searched for and their IDs will be listed in "Other Matches." An example is given below. Four Yahoo IDs are returned for "JAZZ" and thirteen other related IDs are also returned for strings that contain JAZZ.

```
Usage: java IDsearch <["[]keyword ...[]"]>

argerich-120 spencer-progs>: java IDSearch jazz
Key Word(s):    [jazz]
Searching......
Exact Matches:
 1      Level 3:        2000019527      JAZZ
 2      Level 3:        1600024648      JAZZ
 3      Level 3:        1600023878      JAZZ
 4      Level 4:        1600123889      JAZZ

Other Matches:
 5      Level 3:        1600739154      ACID_JAZZ
 6      Level 4:        1600739154      ACID_JAZZ
 7      Level 4:        1600713152      JAZZ_DRUMS
 8      Level 4:        1600713148      JAZZ_VOCALISTS
 9      Level 5:        1600328307      JAZZ_JACKRABBIT
 10     Level 5:        1600328307      JAZZ_JACKRABBIT
 11     Level 5:        1600704618      JAZZ_PIANO
 12     Level 5:        1600704616      JAZZ_GUITAR
 13     Level 5:        1600739189      JAZZ_VIOLIN
 14     Level 5:        1600739147      JAZZ_SAXOPHONE
 15     Level 5:        1600704621      JAZZ_TRUMPET
 16     Level 6:        1600328307      JAZZ_JACKRABBIT
 17     Level 6:        1600067187      UTAH_JAZZ
Total:  17
```

PathFinder One of the common requests from users and other modules is to find the location of an interest in the ontology hierarchy. *PathFinder* or *PathFinder_wID*, which is enhanced with Yahoo ID lists, was implemented for this purpose and will return all the paths, from the root to the query interest in the ontology hierarchy. Those paths are also stored in an ArrayList, which is ready to be passed to other modules or Java programs. However, this description simplifies and glosses over many implementation

problems. The DAG problem (as described in Section 3.5) needs to be taken care of during the implementation as follows.

1. A Yahoo ID is passed into the program.
2. The "geller.ontology" table in the database is searched for all the tuples associated with the given ID. The resulting tuples are in the forms of paths of interest names at each levels and are collected into a list L_{ID}.
3. The same information is maintained as a queue Q_{ID} and a list of the corresponding paths, L_{path}, of the Yahoo IDs.
4. For each tuple dequeued from the queue Q_{ID},
   (a) All the interests in the path, except the last one, are translated into Yahoo IDs. L_{ID} is updated, changing the interest name paths into Yahoo ID paths.
   (b) For each Yahoo ID, all the paths are found in the database and are enqueued into Q_{ID}, eliminating duplicates. L_{path} is updated accordingly.
5. Recursively replace the paths in L_{ID} by searching for the ID paths in L_{path}.
6. Return the list L_{ID}.

An execution example is given below:

```
Usage: java PathFinder_wID 10-digit-ID


argerich-73 spencer-progs>: java PathFinder_wID 1600016647

Input ID:1600016647
Searching......

There are 6 paths are found
Paths in the format of names
 1   Depth 4    ENTERTAINMENT_ARTS/ACTORS_AND_ACTRESSES/L/
 LOPEZ_JENNIFER/
```

```
2   Depth 4    ENTERTAINMENT_ARTS/ACTORS_AND_ACTRESSES/
COMPLETE_CATEGORY_LISTING/LOPEZ_JENNIFER/
3   Depth 4    MUSIC/ARTISTS/L/LOPEZ_JENNIFER/
4   Depth 4    MUSIC/ARTISTS/COMPLETE_CATEGORY_LISTING/
LOPEZ_JENNIFER/
5   Depth 6    MUSIC/GENRES/ROCK_AND_POP/ARTISTS/L/
LOPEZ_JENNIFER/
6   Depth 6    MUSIC/GENRES/ROCK_AND_POP/ARTISTS/
OMPLETE_CATEGORY_LISTING/LOPEZ_JENNIFER/

Paths in the format of IDs
1   Depth 4    2000002139/2000002140/2000002814/1600016647/
2   Depth 4    2000002139/2000002140/2000003561/1600016647/
3   Depth 4    2000015596/2000015597/2000016608/1600016647/
4   Depth 4    2000015596/2000015597/2000017569/1600016647/
5   Depth 6    2000015596/2000019540/1600025955/2000020438/
2000021041/1600016647/
6   Depth 6    2000015596/2000019540/1600025955/2000020438/
2000021615/1600016647/
Average Level:          4
The Lowest Level:       6
```

In the example of "LOPEZ_JENNIFER" above, there are six paths returned. The interest "LOPEZ_JENNIFER" has multiple parents and belongs to two different top-level categories, "ENTERTAINMENT_ARTS" and "MUSIC."

GetAncestor This method is used to provide a better display for users who want to find ancestors at a certain level. By creating a *PathFinder_wID* object shown above, it is convenient to get the ancestors on all paths. The ancestors at that queried level will be extracted from the paths and be displayed.

```
Usage: java GetAncestor YAHOOID LevelNo

Example: java GetAncestor 1602749611 3
```

Raiser The *Raiser* program module is used to raise all the interests in an input file to a given level. A new file "file1_raisedto_level_5" will be the output when raising a file "file1"

to level 5. During the execution, all unique interests in the file are collected and raised. A list of each interest with its ancestors at the given level is created. According to this list, the output file is created by replacing the necessary interests by their ancestors in the input file. Duplicates will be eliminated during the replacement.

```
Usage:  java Raiser filename levelraiseto

Example: java Raiser business_finance_age_gender_mining_new 3
```

FetchMiningData Input data need to be prepared before data mining. In this research, all the data are stored in the ORACLE database. The *FetchMiningData* module is used to retrieved data from the database and write the information into a file in a certain format which can be accepted by the FP-Growth data mining program. A few options are given to execute the program, either to retrieve all people from the database or only to retrieve the people with interests including at least one interest which is a descendant of the given input interest at a given level. The people and their interests are listed in the input file, one person per line as in the format described in Section 4.3. The example below is to retrieve all the information of people who have at least one interest whose level 1 ancestor is "BUSINESS_FINANCE."

```
Usage: java FetchMiningData levelnumber keyword
    or java FetchMiningData all

Examples: java FetchMiningData 1 BUSINESS_FINANCE
```

GettingAncRaising The *GettingAncRaising* module is used for Raising. The program reads the input file to find all the interests appearing in it. All the ancestors of the involved interests are found by using the *PathFinder* tool. The ancestors for interests at each level are stored in an ancestor table. Also, the lowest level of all involved interests is found.

Then the original data file is raised to all levels above the lowest level by using the ancestor table. Input files that have been raised to different levels are generated.

```
Usage:  java GettingAncRaising <inputfilename> [category ID]

Example: java GettingAncRaising business_mining
     or: java GettingAncRaising business_mining 2000000001
```

FPMining The *FPMining* module implements the FP-Growth algorithm described in Section 4.3. According to [5, 6], four major steps of the FP-Growth method are:

1. Construct FP-Tree from an input data file.

2. Construct conditional pattern base for each node in the FP-Tree.

3. Construct conditional FP-Tree from each conditional pattern-base.

4. Recursively mine conditional FP-Trees and grow frequent patterns obtained so far. If the conditional FP-Tree contains a single path, simply enumerate all the patterns.

In the FP-Tree implementation, a Java class *TreeNode* was defined for a node, which was structured as shown in Figure 7.1. An FP-Tree was represented by a set of *ArrayLists* of *TreeNodes*. Each of the *ArrayLists* represented a branch of the tree. For example, for a simple tree such as in Figure 7.2, the following five branches are needed. The alphabetic order of the node names follows the insertion order of the nodes when being inserted into the tree. Thus, A is the first node, after the root of the tree and J is the last node to be inserted.

```
(1) A - B - C - D
(2) E
(3) F - G - I
(4) H
(5) J
```

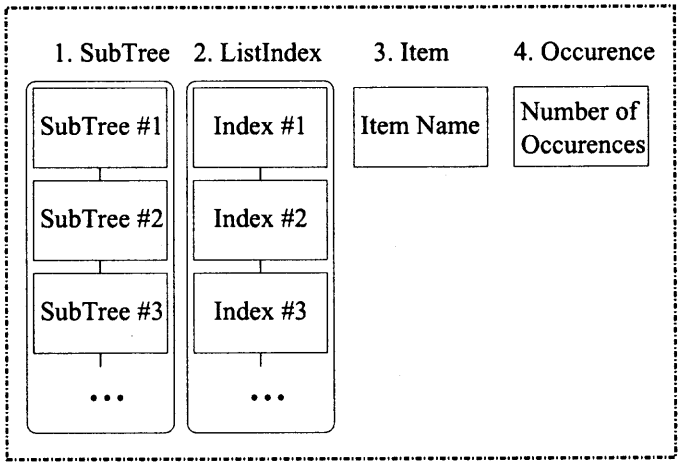


Figure 7.1 Structure of a TreeNode.

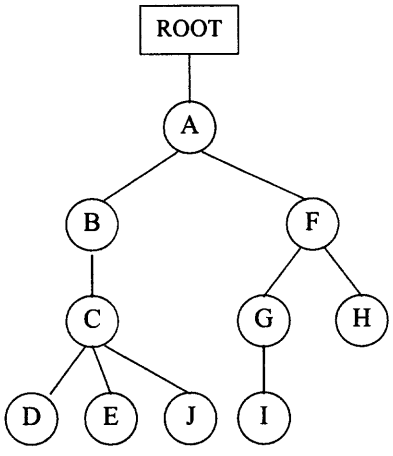


Figure 7.2 A simple tree.

In Figure 7.3, the lists are shown in the form of *TreeNodes* and with a header node at the beginning of each list. For simplicity reasons, all the numbers of occurrences are set to 1. At the beginning of each list, there is a header node to describe the origin of this branch, *i.e.*, the location of the parent of the first node in the branch. In this representation, the node C has three children, D, E, and J. F is a child of A, which located in the list #0, index #1. H is a child of F, which is located in the list #2, index #1.

```
Usage:  java FPMining <min_support> <min_confidence>
[<file_name>]/[<product_num> <transaction_num>]

Example: java FPMining 0.5 0.75
```

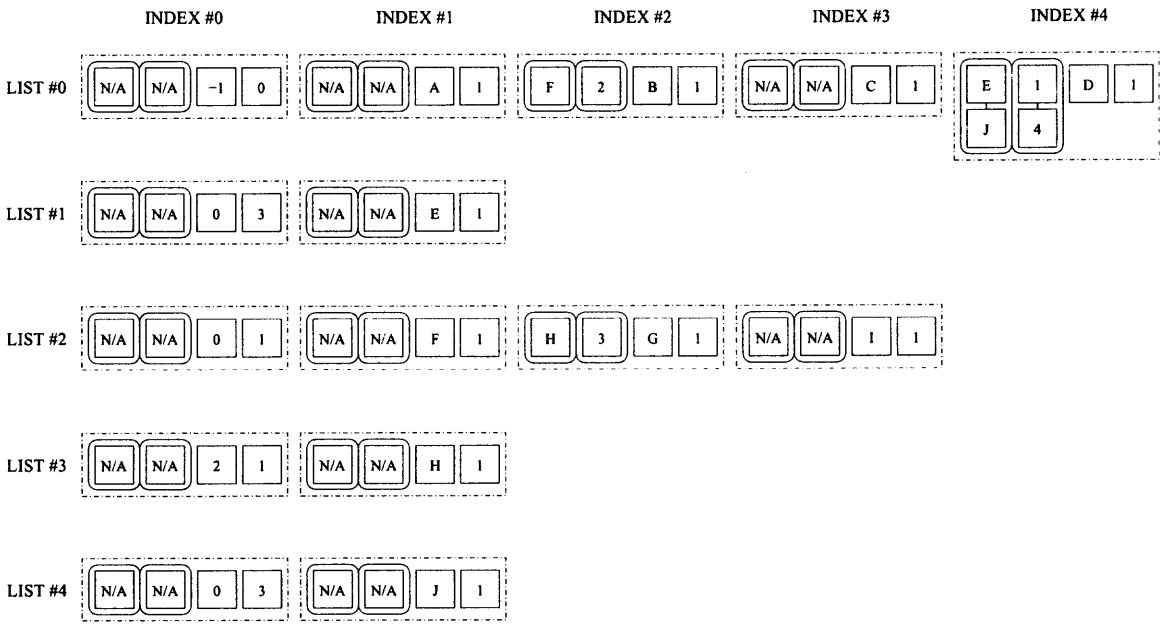


Figure 7.3 List representation of the tree in Figure 7.2.

```
or: java FPMining 0.5 0.75 4 5
or: java FPMining 0.5 0.75 data.txt
```

Filter The *Filter* module is used after rules have been derived from data files. However, as described in Chapter 5, the rule types in Group (A) are not useful for marketing purposes. Thus, this program is used to remove all the demographic-demographic association rules from the resulting files.

```
Usage:  java Filter <inputfilename>
```

CHAPTER 8

RANKING AND INTEGRATION

8.1 Personal Interest Ranking by Query

Imagine you're in a mail-order business and are trying to mail a promotional offer to a million households, most of whom will not respond [3]. Instead of sending mail to everybody in your database, a better choice is to select a group of potential customers as the target of the promotion [76]. To find such a group, one possible method is to rank all the people that were selected from a customer database according to available information and select a group of top-ranked customers. This problem also exists when users want to retrieve information from the Web Marketing Project. They will prefer an ordered list so that they will know who might be the best customers to contact.

The method of ranking is often used in data mining [3]. In [77], a *weighted frequency* is used to rank each concept appearing in a Web page and to index the page with an ontology. A ranking is also used to improve search results in [78]. More precisely, what happens in [78] is a *Re-Ranking* of Web pages. A function is applied to modify the ranking that is returned by the underlying search engine ProFusion. Since a ranking may be decided by the properties of relationships, research on similarity is also instructive for ranking purposes. In [79], semantic similarity is evaluated by the distance between the nodes corresponding to WordNet's taxonomy. In [80], similarity is examined from a spatial perspective.

Given are several people with sets of expressed interests, such as {MUSIC, MOVIE, JAZZ, *etc.*} It is necessary to construct a comparison operator which could be used to compare pairs of such people with respect to one (or several query) interests. This is interesting for the following reason. If market researcher are looking for persons in a database which are interested in one specific topic (= query interest), such as MUSIC,

then a person that specifies many interests such as BASEBALL, FISHING and MUSIC, is probably not as excited about music as a person that specifies only MUSIC. Thus, it may be proper to rank the person that lists only MUSIC higher. Practically speaking, that person is more likely to spend money on items related to music, than a person that has very many interests (including music). However, the problem gets complicated by a number of factors. One is that different interests are hierarchically related to each other.

1. One might argue that an interest that is lower in the hierarchy is more specific, and therefore more directly tied to a purchasing action. For example, if a person says s/he is interested in MUSIC, s/he cannot go out and "buy music." A major decision process is necessary to decide what to actually buy. On the other hand, if a person is interested in JENNIFER LOPEZ (lower in the hierarchy), see Figure 8.1, then almost no decision has to be made, because only a few kinds of JENNIFER LOPEZ CDs are available, and the user might already have all the old ones. Thus, one might say a specific interest should be ranked higher, because it is more likely to lead to a real purchasing action.

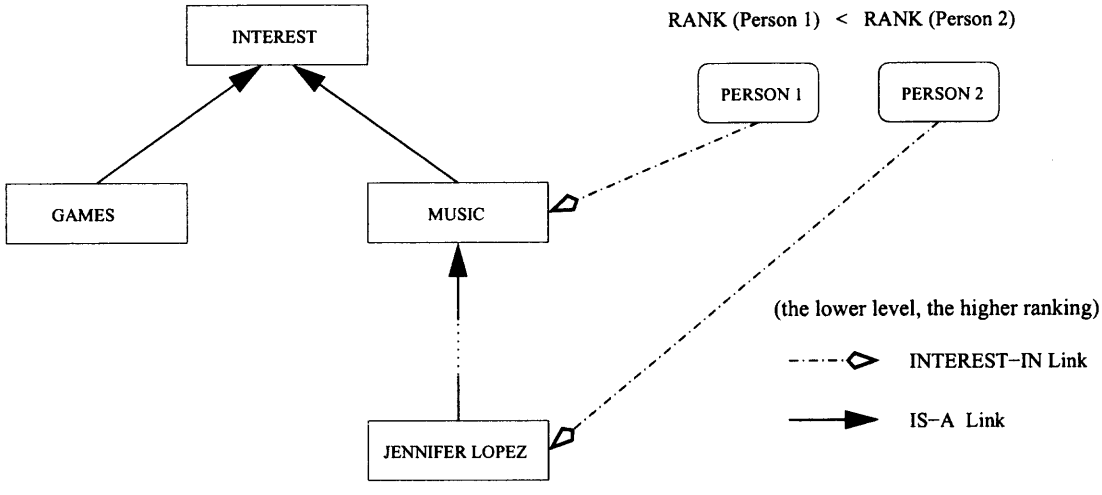


Figure 8.1 Ranking by specificity.

2. To the contrary, one may argue that a higher level interest such as MUSIC should get a higher rank, for the following reason: In the interest hierarchy, a node that is higher

up is likely to have many more nodes under it, such as in Figure 8.2. Therefore, a person saying that he is interested in MUSIC might be interested in many different kinds of music which are all under MUSIC. Thus, one might argue that a higher level interest should probably be ranked higher, because it has many interests under it. In fact, this may be refined so that the ranking is not based on the level but on the number of descendants of an interest, such as in Figure 8.2. Thus, a high level interest with no children at all would not get a high ranking just because of its level.

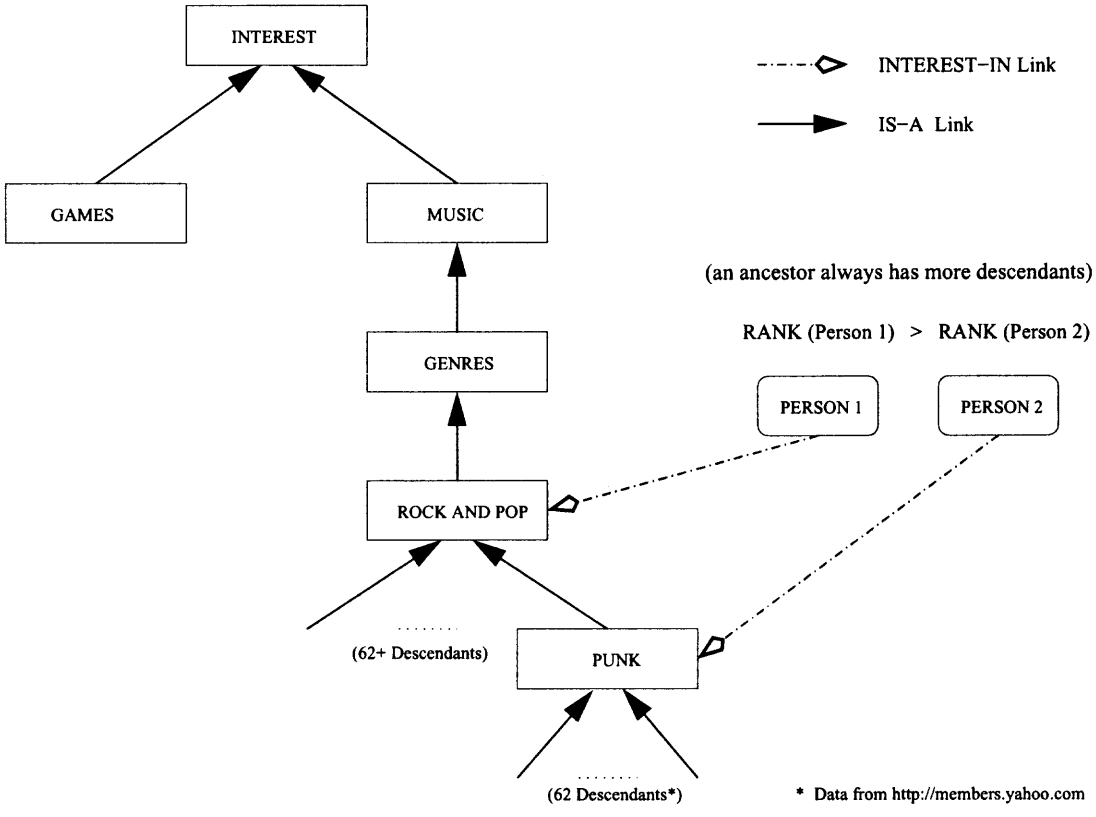


Figure 8.2 Ranking by descendant number.

As will be explained now, specificity is a better indicator for ranking than number of descendants, however, both specificity and number of descendants should figure into a final decision. Why specificity is (probably) a better way to score interests is now explained. If a person says that s/he likes MUSIC, that typically does not mean that s/he likes *all* music. For instance, Jim likes MUSIC, but if somebody asks him in detail, he does not really like

COUNTRY MUSIC. Thus, when he says MUSIC, he does not mean that he likes *every* descendant of MUSIC.

In fact, this selective thinking goes even farther. Jim does like ROCK MUSIC, but he does not like JOHN MELLENCAMP, even though he is certainly playing rock music. This idea can be developed even more. Jim likes the BEATLES, but he does not like the song Revolution Number 9. Thus, the argument that for an interest all its descendants are important is a very weak one. Especially, if a person expresses both MUSIC and one of its children, such as JAZZ, as interest, then it is possible to interpret this as "I am interested in MUSIC, especially (and mostly) in JAZZ." Therefore, the descendant count is rejected as primary discriminant in favor of specificity.

In order to rank people by their interests according to the query interest, an interest score is calculated for each person. However, a query could include more than one interest. These interests in the query are called "query terms" in this proposal. For instance, some companies might need people who are interested in both COMPUTER GAME and FOOTBALL to promote a new football computer game. There is a need to develop a method to rank people with multiple interests in different relative positions in the ontology for queries with several interests.

8.1.1 Calculate the Interest Score by Specificity

The score calculated by specificity is with reference to the location of the interest in the interest hierarchy. Since there are 16 level 1 categories in the hierarchy, the 16 categories are divided into 16 *Category Areas*. Two interests in different category areas are defined to be "unrelated." Thus, the algorithm does not need to decide whether BRIDGE or HEARTS is closer to JAZZ, it is indeed unable to tell. Since JAZZ is in the MUSIC category area and the other two are in the GAMES area, the two are equally unrelated to JAZZ. What *is* important is the distance between two interests in a same category area. Because the hierarchy is a DAG, there exist interests with multiple parents in various category areas.

Only the interests of the category area are considered in which the query interest is located. Since only a single-interest query is discussed here, no interests from two or more category areas need to be considered simultaneously. The computation will be performed in the category area in which the query term is located. Every interest of a person will be assigned a score. This score will be calculated by the distance between this interest and the query term location in the interest hierarchy. The edges which connect two of them have an "edge value" to reflect the closeness and are included into the calculation of assigned scores. From these values a score for each person can be computed.

The above definitions are demonstrated in Figure 8.3. After the interest scores for every person returned by a database query have been computed, the algorithm is able to sort the scores and rank those people by the interest scores. Those with higher scores will be those who are more interested in the query term. The descendant count in the ranking with specificity is included at this point. By doing this, the scores of two siblings with equal specificity scores are distinguished, based on descendant numbers.

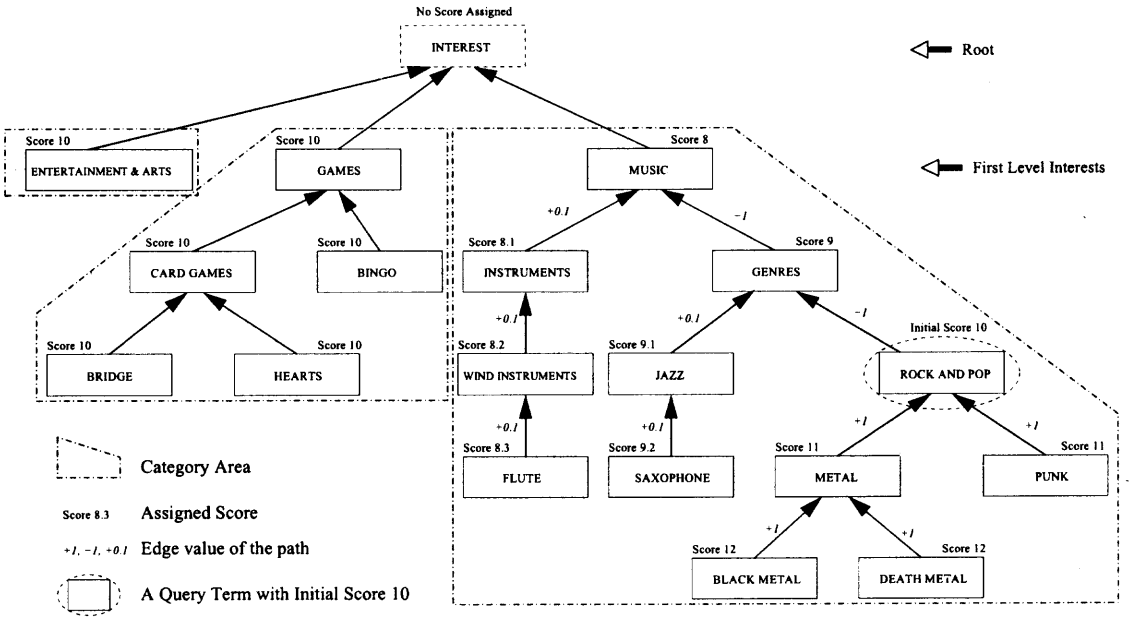


Figure 8.3 Model of interest score calculation.

8.1.2 Calculation and Analysis

At first glance, the easiest way to decide how close two concepts are is to calculate the distance by counting the intervening nodes between the two concepts. However, such a traversal will not necessarily result in a shortest path. Using a BFS strategy, starting from one of the concepts, a shortest path will be found for the shortest distance.

Will the shortest path be a good solution? The answer is negative. For example, in Figure 8.3, the concepts BLACK METAL and DEATH METAL are connected by a shortest path of length 2. The same length of 2 applies to the shorest path between MUSIC and GAMES. The intuitive response is that MUSIC and GAMES are less closely related terms than are BLACK METAL and DEATH METAL. Thus, measuring semantic distance using solely the number of nodes traversed fails.

A better approach is to take the depth of the hierarchy into account [81]. In the previous example, MUSIC and GAMES occupy a much higher location in the hierarchy than BLACK METAL and DEATH METAL. Thus, paths of the same length might not represent the same degree of closeness.

In this research, there is one fixed query term (in a single-interest query) which appears in every pair of compared nodes[1]. The interest scores is calculated by traversing paths, starting from the query term in the hierarchy. By the specificity approach described before, an algorithm that shows the following behavior is needed:

1. Any concept that is a proper descendant of the query term Q should have an interest score greater than the score of Q.

2. Any concept that is an proper ancestor of the query term Q should have an interest score less than the score of Q.

3. Any concept that is a descendant of a sibling of the concept C should have an interest score less than the score of C.

[1]Remember that query terms are selected from generated menu, thus they always exist in the hierarchy.

4. Any concept in a category area other than the category of the query term Q should have the same interest score as the score of Q.

5. When considering descendant number, when having the same interest score by specificity, the concept with more descendants should have a greater score than the other one.

Implementing this algorithm is left for future work. Then for a single-interest query, the returned results of the front-end will be ranked in a way that helps Web users to find the best prospective customers. The ranking algorithm will be evaluated for correctness and efficiency.

8.2 Multiple Marketing Ontology Hierarchy Integration

In this project, the marketing ontology is the backbone of the overall research. The interest hierarchy in the ontology currently being used was initially retrieved from the Yahoo Member Web site. This hierarchy is well structured. However, the Web Marketing Project is not limited to Yahoo. The project also extracted data from other portal sites such as ICQ and LiveJournal. However, those sites have their own interest hierarchies. For example, Yahoo has a hierarchy of 16 categories, which become the 16 level 1 concepts. Yahoo has up to 11 levels[2]). ICQ has a hierarchy of 43 categories but only two levels. LiveJournal has 400 listed concepts and has no multi-level hierarchy.

Not only the structures of the hierarchies of different portal sites are different, but the concepts in the hierarchy are not the same either. There are 31,534 different concepts in the Yahoo hierarchy while there are only 1670 in ICQ and 400 in LiveJournal. Moreover, though the latter two hierarchies are much smaller than Yahoo, there are still many concepts which are not included in the Yahoo hierarchy. This was hard to believe at first.

[2]At the time of this research.

It is preferable to have a single ontology hierarchy to include all the interests of all existing personal data. Thus, there is a need to merge the Yahoo, ICQ and LiveJournal hierarchies together into a new larger one.

To better understand the problematics of integration, a study of "integrating by hand" was performed. An overall matching was performed between the Yahoo interests and ICQ interests. Among all the interests, there are only 420 identical in Yahoo and ICQ. Since an interest name may appear more than one time in the hierarchy, it is normal that duplicated interests have larger numbers of occurrences in both hierarchies[3]. Thus duplicates appear 544 times in the ICQ hierarchy and 1,242 times in YAHOO. As a first attempt, the brute-force way was used to try string matches between two hierarchies. If two nodes $A1$ and $A2$, belonging to different hierarchies, and are the same ($A1 = A2$), the descendants of $A1$ and $A2$ can be recursively combined as $A1$'s descendants. However, after comparing the Yahoo interests and first level ICQ interests, few interests of ICQ were identical!

Thus, an attempt was made to split the high level interest names of yahoo to increase the matching rate. Since most of Yahoo's first level interests are combinations of words, such as "BUSINESS AND FINANCE" AND "COMPUTERS AND INTERNET," they can be split into "BUSINESS," "FINANCE," "COMPUTERS," and "INTERNET." Though the interests "BUSINESS AND FINANCE" and "COMPUTERS AND INTERNET" do not match any of the interests in the ICQ hierarchy, the four interests after splitting can all be matched, which greatly increased the matching rate. Thus, all 16 Yahoo first level interests were split into 28 single-word interests. Among them, 15 interests matched ICQ interests. The plural and singular formats, such as "CULTURES" and "CULTURE" or "ARTS" and "ART," were counted as the same since it is reasonable to transform them by applying the STEMMING algorithm [82, 83]. Among the rest of the 28 interests, eight interests were found to be strong matches with some ICQ interests (for the human eye). For three interests a partial match was found and for two interests no matches were found. The Table 8.1 lists

[3]Again as the example of TEXAS described in Chapter 7, several different interests may share the same interest name.

Table 8.1 Matching between Yahoo first level interests and ICQ interests

	Yahoo Interests	ICQ Interests
Exactly Matched	Business, Finance, Computers, Internet, Cultures, Entertainment, Arts, Games, Government, Health, Hobbies, Music, Sports, Religion, Science,	
Strongly Matched (to the author)	Family	Parenting
	Home	Home Automation, Household Products
	Politics	Parties
	Wellness	Health
	Crafts	hobbies
	Beliefs	Religion
	Romance	Lifestyle
	Relationships	Lifestyle
Weakly Matched	Community	Social Science, Lifestyle
	Recreation	Outdoors, Travel
	Schools	Parenting
Not Matched	Reginal	
	Education	

the matching results. Since the match was performed by a human, there are some lines in the table where one Yahoo interest matches two ICQ interests.

CHAPTER 9

CONCLUSIONS AND FUTURE WORK

In Section 9.1, conclusions are presented based on this research. Moreover, some future work is described in Section 9.2.

9.1 Conclusions

In the Web Marketing Project, four ontology usage enhancements were studied. The Raising and Ranking methods take advantage of an ontology to improve the functionality of the whole Web Marketing Project. The intersection ontology was developed to accommodate customer data. Integration is needed to merge ontologies from various sources into one, achieving a wider coverage. Below are conclusions about:

1. Use of Raising to improve the support of mined rules.
2. Design of a marketing ontology using intersections.
3. Ranking results of a query.
4. Integrating several ontologies.

9.1.1 Raising

As a result of sparse data in the real world, data mining produces many rules with very low support. The Raising method was invented to maximize the support values of all rules generated in the mining process. As described in Chapter 3, the Raising increased the support value of rules by preprocessing the input data. To raise to a level, data were replaced by their ancestors at the given level. Thus, the data mining algorithm was able to derive rules with higher support values. In the evaluation shown in Table 3.2, the average

support is 170 before raising, and 4,527 after raising. The average improvement is 2898%. This is a substantial improvement in absolute support.

Moreover, approaches were discussed on how far to raise data in Chapter 3. A graph was generated showing the sum of support values for all rules at a level over the level number. This graph had one peak at approximately level 2. This peak indicated an optimum level for Raising. Exact optimal Raising levels were computed with two approaches, resulting in 1.65 and 1.90, which both have to be rounded to 2.0.

The WEKA package was used as the data mining tool at the beginning. However, as illustrated in Section 4.2, limitations of WEKA itself were found. Triples of age, gender and one interest were used as a record for data mining. As a result, on one hand, only the association rules that connect demographic information with interests were retrieved. Thus, resulting rules were of formats such as, e.g. "Those who are interested in WARGAMING are male and between 10-19." However, more association rules, such as those that connect interests with interests, had been expected to appear in the results. Such rules like "People who are interested in WARGAMING are also interested in INTERNET" could greatly benefit strategies for marketing purposes. On the other hand, the input ARFF format of triples also led to improper changes of support and confidence values of rules generated. Spurious data were added into the datasets during Raising. To generate additional desirable rules and to correct the improper support and confidence values, the FP-Growth mining application, proposed by Jiawei Han in [5] was implemented by the author, as described in Chapter 4. It was found to be a good alternative to the Apriori algorithm implemented in the WEKA package. Raised data in the new format were then fed into the FP-Growth program for rule generation. By using this implementation, the limitations of the ARFF format were overcome; the support and confidence values were not negatively affected. Also more kinds of association rules were generated, including the desired interest-interest rules. As shown in Table 4.5, interest-interest rules were derived in all the categories listed, e.g. up to 169 interest-interest rules in the category "GOVERNMENT POLITICS."

The Raising method described in Chapter 3 only preserved interests belonging to one level in the dataset for data mining. All the interests below the given level were replaced by their ancestors and all interests above the given level were ignored. Because interest-interest rules were originally not included in the results, this effect did not become explicit. However, once interest-interest rules were generated as results, rules such as "MUSIC → WARGAMING" were also expected to be produced, even though MUSIC is at level 1, while WARGAMING is at level 2. In the new Raising method in Chapter 4, among all the interests in a dataset, Raising only affected the interests below the given level. Other interests which are above the given level were also included in the dataset for data mining. In other words, the dataset after Raising preprocessing contains interests at different levels which are equal to or higher than the given level. Thus, as a result of this research, multi-level interest-interest rules for very large attribute value sets could be generated as well. This was not possible in WEKA, due ti limitation of the ARFF format.

Chapter 5 gave a detailed analysis of the association rules generated in this research. All the 21 types of association rules that are possible were categorized into six groups. For each group, the implications for marketing purposes were discussed. Moreover, for rules in each group, the effects on the support and confidence values caused by Raising were analyzed. As a conclusion, the support values are always increased or at least kept the same during Raising. However, the changes of confidence values during Raising are different in different groups and depend on the contents of the dataset. For the important class of rules, such as type #2, #4 and #10 in Group (B), it was shown that support values and confidence values always go up, following the formula:

$$S_{new} = S_0 + Inc \geq S_0$$

$$C_{new} = \frac{S_0 + Inc}{Occ_{ante}} \geq C_0$$

9.1.2 Intersection Ontology

The development of an application ontology for customer classifications in a marketing knowledge base was described in Chapter 6. The intersection ontology representation allows the representation of "buys" relationships by single links whenever this is warranted by the mined marketing knowledge. Yet, this representation does not produce a combinatorial explosion of all possible intersection nodes. Rather, it only represents the concepts for customer classes which are necessary as sources for known "buys" relationships.

The multi-level hierarchy representation fulfills one requirement for ontologies, namely that they can be represented by diagrams of relatively low size and visual complexity. This representation typically requires lower visual complexity relative to the three-level intersection hierarchy. As described in Section 6.6, in the evaluation based on the Web Marketing Project, the multi-level representation has a 24% lower visual complexity than the three-level representation. Moreover, its size is 40% smaller than the size of the ordered dimensions representation. For the marketing domain an economical intersection ontology may be created by inserting intersections of options of the various classification dimensions on demand. Such a representation may also be proper for applications in other domains than the marketing domain.

One may argues that for a large dataset, all the nodes will have to be created. In the theoretical worst case, the nodes in the customer hierarchy would include all possible combinations. As already discussed in Section 6.7, the nodes were created on demand. Thus, some impossible nodes, such as TEENAGE MAN WITH CHILDREN, are not created without demand. Moreover, an experiment was done to evaluate this problem. Five dimensions and 19 options were involved in the experiment. Each dimension had 3, 6, 7, and 3 options individually. Thus, the total number of possible created nodes was 378. During the experiment, there were 2,000 records created randomly. Analyzing the on demand created nodes for these randomly generated records, 377 nodes were created, which is 99.7% of the maximum number of possible nodes. In this experiment, all options

were generated using uniform distributions. However, in a specific domain, the distribution of options could be different. For example, in a study of GAMING one would expect to find more young people. Thus, in another experiment 2,000 records were created randomly. This time, the distributions of options were not uniform. The first option had the probability of $1/2$. All the other options had the probability half of the previous option, except the last option had the same probability as the previous option. The total was still 1. The number of nodes created in this experiment was 251, which is 66.4% of the maximum number of possible nodes. Thus, the advantages of the intersection ontology become more explicit when the options of each dimension are not uniformly distributed.

9.1.3 Ranking

An interest scoring algorithm was introduced in Chapter 8. According to this scoring algorithm, concepts were given scores as follows:

- Any concept that is a proper descendant of the query term Q should have an interest score greater than the score of Q.
- Any concept that is an proper ancestor of the query term Q should have an interest score less than the score of Q.
- Any concept that is a descendant of a sibling of the concept Q should have an interest score less than the score of Q.
- Any concept in a category area other than the category of the query term Q should have the same interest score as the score of Q.
- If interest scores are assigned equally to two concepts by specificity, the concept with more descendants should have a greater score than the other one.

The scoring algorithm is applied to people's records when a query interest is given. People can then be ranked by the interest scores assigned to them. The person with the

highest interest score is ranked first *etc.*, i.e. people are sorted by the rank. Those with higher scores are people who are likely to be more interested in the query term.

9.1.4 Integration

Three interest hierarchies have been extracted form the Web, in this research. Though the Yahoo hierarchy was primarily used, the integration of Yahoo and ICQ and LiveJournal was studied. Yahoo has 31,534 interests up to 11 levels deep. ICQ consists of a hierarchy of 43 categories but is only 2 levels deep. LiveJournal only has 400 listed concepts and has no multi-level hierarchy. A matching was performed and only 420 identical concepts were found in Yahoo and ICQ. Moreover, a splitting of names of top level Yahoo interests added 15 out of 28 exact matches and 11 possible matches.

9.2 Future Work

Some future work is suggested by this research.

9.2.1 Confidence Values in the Raising Method

In the discussion of finding the optimal Raising level, the current analysis has involved only support values. In future work, confidence values could also be used to compute an optimal Raising level. As was shown in Chapter 5, there are cases in which Raising reduces confidence values of association rules. Thus, the inclusion of confidence values in the calculations might push the optimal raising level to the right (in Figure 3.3).

In the processing of Raising, there is a duplication check performed while replacing the interests by their corresponding ancestors at a specific level, as described in Section 4.3. Such a check eliminated one interest if it would appear in somebody's interest list more than once. However, though it is the case that people will not express the same interest twice in an interest list, it is still possible that two siblings are expressed at the same time. When Raising is performed to the level where the two siblings' parent is located, the new interest

list for this level only contains the parent once. One concern arises here whether this parent should be counted twice in the list since two of its children were originally expressed. In other words, should the interests in the list be assigned a weight in such situations? In that case, when somebody expressed those two siblings, one might want to stress his interest in the same category. Should this stress also be considered in the new list after Raising?

To expand this idea to the supermarket shopping cart example, a weight could be assigned according to the values of products. Thus, should a product for which customers paid a lot be assigned a higher weight and function as more important in the mining than other products which cost less? There is no answer to these questions now. However, a mining method with weighted items might be a solution, which will be included in future work.

9.2.2 Application for the Intersection Ontology

As described in Chapter 6, an intersection ontology would be useful in many other domains. Thus, future work includes the development of an application for creating an intersection ontology on demand. Such an application should not be restricted to a specific domain but could be used everywhere, such as in the domains of architecture or civil engineering.

9.2.3 Implementation of the Ranking Algorithm

Implementing the Ranking algorithm is left for future work. The implementation will be integrated in the Web Marketing Project Web site. For a single-interest query, the returned results of the front-end will be ranked in a way that helps users of the Web Marketing System to find the best prospective customers, i.e. the people with greater interest scores or higher ranks. Thus, an option of sorting people by their interest scores should be available in the result page shown in Figure 1.8. The Ranking algorithm needs to be evaluated for correctness and efficiency.

Moreover, multiple-interest query Ranking also needs to be done in the future. In the Web Marketing Project, in the interest query page, as in Figure 1.5, it is possible to choose multiple interests as query terms. However, the current Ranking algorithm only works for single-interest queries. Thus, an extended algorithm for multiple-interest query terms needs to be developed based on the single-interest query algorithm.

9.2.4 Extend SEMINT to Hierarchy Integration

In [84], an ontology integration method, the Semantic Integration (SEMINT) algorithm, was proposed using Algorithmic Semantic Refinement (ALSER) [63, 66]. In the future, SEMINT could be applied to hierarchy integration. Since the data from Yahoo, ICQ and LiveJournal contain three hierarchies, an integrated interest hierarchy will be created based on them. All concepts in the hierarchies from the three sources will be assigned semantic types by a human. Thus, there will result three Terminological Knowledge Bases (TKBs). Then ALSER will be applied to all TKBs and result in three new TKB's with intersection types.

REFERENCES

[1] T. Berners-Lee, "The Semantic Web (slides)," http://www.w3.org/2002/Talks/04-sweb/.

[2] "HP's Web Language (WebL)," http://www.hpl.hp.com/downloads/crl/webl/index.html.

[3] I. H. Witten and E. Frank, *Data Mining, Practical Machine Learning Tools and Techniques with Java Implementations*. San Francisco: Morgan Kaufmann Publishers, 2000.

[4] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules," in *Proceedings of the 20th International Conference on Very Large Data Bases VLDB*, J. B. Bocca, M. Jarke, and C. Zaniolo, Eds. Morgan Kaufmann, 1994, pp. 487–499. [Online]. Available: citeseer.ist.psu.edu/agrawal94fast.html

[5] J. Han, J. Pei, and Y. Yin, "Mining frequent patterns without candidate generation," in *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, W. Chen, J. Naughton, and P. A. Bernstein, Eds. ACM Press, 2000, pp. 1–12. [Online]. Available: citeseer.ist.psu.edu/han99mining.html

[6] J. Han, J. Pei, Y. Yin, and R. Mao, "Mining frequent patterns without candidate generation: A frequent-pattern tree approach," *Data Mining and Knowledge Discovery*, vol. 8, no. 1, pp. 53–87, 2004.

[7] J. Geller, R. Scherl, and Y. Perl, "Mining the Web for target marketing information," in *Proceedings of the Collaborative Electronic Commerce Technology and Research (CollECTeR)*, 2002.

[8] R. Scherl and J. Geller, "Global communities, marketing and Web mining," *Journal of Doing Business Across Borders*, vol. 1, no. 2, pp. 141–150, 2002.

[9] X. Zhou, J. Geller, Y. Perl, and M. Halper, "An application intersection marketing ontology," in *Theoretical Computer Science: Essays in Memory of Shimon Even*, ser. Lecture Notes in Computer Science, A. L. S. Oded Goldreich, Arnold L. Rosenberg, Ed., 2006, pp. 143–153.

[10] E. Portscher, J. Geller, and R. Scherl, "Using internet glossaries to determine interests from home pages," in *Proceedings of the 4th International Conference on Electronic Commerce and Web Technologies (EC-Web)*. Springer Verlag, 2003, pp. 248–258.

[11] "Web Marketing Project Web site," 2005, http://web.njit.edu/challeng/webmarketing.html.

[12] M. R. Quillian, "Semantic memory," in *Semantic Information Processing*, M. L. Minsky, Ed. Cambridge, MA: The MIT Press, 1968, pp. 227–270.

[13] L. S. P. Bouquet, A. Dona and S. Zanobini, "Contextualized local ontology specification via ctxml," in *Proceedings of the 2002 AAAI Workshop on Meaning Negotiation (MeaN)*, 2002.

[14] "Dictionary.com Web site," http://dictionary.reference.com/.

[15] T. R. Gruber, "A translation approach to portable ontology specifications," *Knowledge Acquisition*, vol. 5, pp. 199–220, 1993.

[16] J. Geller, Y. Perl, and J. Lee, "Guest editors' introduction to the special issue on ontologies: Ontology challenges: A thumbnail historical perspective," *Knowledge and Information Systems*, vol. 6, no. 4, pp. 375–379, July 2004.

[17] T. Berners-Lee, J. Hendler, and O. Lassila, "The Semantic Web," *Scientific American*, vol. 284, no. 5, pp. 34–43, May 2001.

[18] N. Guarino and R. Poli, "Special issue on formal ontology in conceptual analysis and knowledge representation," *Journal of Human-Computer Studies*, vol. 43, no. 5-6, pp. 831–846, 1995.

[19] J. Hendler, "Special issue on agents and the Semantic Web," *IEEE Intelligent Systems*, vol. 16, no. 2, pp. 30–37, March/April 2001.

[20] M. Gruninger and J. Lee, "Special issue on ontology applications and design," *Communications of the ACM*, vol. 45, no. 2, pp. 39–65, Feb. 2002.

[21] D. L. McGuinness, "Ontologies come of age," in *Spinning the Semantic Web: Bringing the World Wide Web to Its Full Potential*, D. Fensel, J. Hendler, H. Lieberman, and W. Wahlster, Eds. Washington, D.C.: MIT Press, 2002, pp. 171–194.

[22] A. Gómez-Pérez, M. Fernandex-Lopez, and O. Corcho, *Ontological Engineering*. New York, NY: Springer Verlag, 2004.

[23] K. Baclawski and T. Niu, *Ontologies for Bioinformatics*. Cambridge, MA: The MIT Press, 2006.

[24] D. Fensel, *Ontologies: A Silver Bullet for Knowledge Management and Electronic Commerce*. New York, NY: Springer Verlag, 2004.

[25] N. F. Noy and C. D. Hafner, "The state of the art in ontology design," *AI Magazine*, vol. 18, no. 3, pp. 53–74, Fall 1997.

[26] J. Geller, Y. Perl, and J. Lee, "Special issue on ontologies: Ontology challenges," *Knowledge and Information Systems*, vol. 6, no. 4, pp. 375–379, July 2004.

[27] Y. Labrou and T. Finin, "Yahoo! as an ontology - using yahoo! categories to describe documents," in *Proceedings of the 8th ACM Conference on Information and Knowledge Management (CIKM)*, October 1999, pp. 180–787.

[28] "DMOZ Open Directory Project (ODP) Web site," 2001, http://www.dmoz.org.

[29] T. H. Cormen and C. E. Leiserson, *Introduction to Algorithms*. Cambridge, MA: The MIT Press, 1999.

[30] W. J. Frawley, G. Piatetsky-Shapiro, and C. J. Matheus, "Knowledge discovery in databases - an overview," *AI Magazine*, vol. 13, no. 3, pp. 57–70, 1992. [Online]. Available: citeseer.ist.psu.edu/frawley92knowledge.html

[31] D. J. Hand, H. Mannila, and P. Smyth, *Principles of Data Mining*. Cambridge, MA: The MIT Press, 2001.

[32] P. Cabena, P. Hadjinian, R. Stadler, J. Verhees, and A. Zanasi, *Discovering Data Mining: From Concept to Implementation*. Upper Saddle River, NJ: Prentice Hall, 1998.

[33] J. Han and M. Kamber, *Data Mining: Concepts and Techniques*, ser. Data Management Systems. Morgan Kaufmann, 2001.

[34] D. T. Larose, *Discovering Knowledge in Data: An Introduction to Data Mining*. Wiley, 2004.

[35] R. Agrawal, T. Imielinski, and A. N. Swami, "Mining association rules between sets of items in large databases," in *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, P. Buneman and S. Jajodia, Eds., 1993, pp. 207–216. [Online]. Available: citeseer.ist.psu.edu/agrawal93mining.html

[36] J. Han, "Mining knowledge at multiple concept levels," in *Proceedings of the 4th International Conference on Information and Knowledge Management (CIKM)*, 1995, pp. 19–24. [Online]. Available: citeseer.ist.psu.edu/han95mining.html

[37] J. Han and Y. Fu, "Discovery of multiple-level association rules from large databases," in *Proceedings of the 21st International Conference on Very Large Data Bases (VLDB)*, September 1995, pp. 420–431. [Online]. Available: citeseer.ist.psu.edu/han95discovery.html

[38] X. Chen, X. Zhou, R. Scherl, and J. Geller, "Using an interest ontology for improved support in rule mining," in *Proceedings of the 5th International Conference on Data Warehousing and Knowledge Discovery (DaWaK)*, 2003, pp. 320–329.

[39] S. Fortin and L. Liu, "An object-oriented approach to multi-level association rule mining," in *Proceedings of the 5th International Conference on Information and Knowledge Management (CIKM)*. New York, NY: ACM Press, 1996, pp. 65–72.

[40] R. Srikant and R. Agrawal, "Mining generalized association rules," in *Proceedings of the 21st International Conference on Very Large Data Bases (VLDB)*, September 1995, pp. 407–419.

[41] G. Psaila and P. L. Lanzi, "Hierarchy-based mining of association rules in data warehouses," in *Proceedings of the 2000 ACM Symposium on Applied Computing*. ACM Press, 2000, pp. 307–312.

[42] R.Páircéir, S. McClean, and B. Scotney, "Discovery of multi-level rules and exceptions from a distributed database," in *Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM Press, 2000, pp. 523–532.

[43] V. Novacek, "Data mining query language for object-oriented database," in *Proceedings of the 2nd East European Symposium on Advances in Databases and Information Systems (ADBIS)*, ser. Lecture Notes in Computer Science, W. Litwin, T. Morzy, and G. Vossen, Eds., vol. 1475. Springer, Spetember 1998, pp. 278–283.

[44] M. G. Elfeky, A. A. Saad, and S. A. Fouad, "ODMQL: Object Data Mining Query Language," in *Proceedings of the 2000 International Symposium on Objects and Databases*, ser. Lecture Notes in Computer Science, K. R. Dittrich, G. Guerrini, I. Merlo, M. Oliva, and E. Rodríguez, Eds., vol. 1944. Springer, June 2001, pp. 128–140.

[45] M. V. Joshi, R. C. Agarwal, and V. Kumar, "Mining needles in a haystack: classifying rare classes via two-phase rule induction," *SIGMOD Record (ACM Special Interest Group on Management of Data)*, no. 2, pp. 91–102, 2001.

[46] M. J. Zaki and C.-J. Hsiao, "CHARM: An efficient algorithm for closed itemset mining," in *Proceedings of the 2nd SIAM International Conference on Data Mining*, R. L. Grossman, J. Han, V. Kumar, H. Mannila, and R. Motwani, Eds. SIAM, April 2002.

[47] H. Mannila, H. Toivonen, and A. Verkamo, "Improved methods for finding association rules," in *Proceedings of the AAAI Workshop on Knowledge Discovery*, July 1994, pp. 181–192.

[48] S. Lu, H. Hu, and F. Li, "Mining weighted association rules," *Intelligent Data Analysis*, vol. 5, no. 3, pp. 211–225, 2001.

[49] I. Bruha and J. Tkadlec, "Rule quality for multiple-rule classifier: Empirical expertise and theoretical methodology," *Intelligent Data Analysis*, vol. 7, no. 2, pp. 99–124, 2003.

[50] F. Berzal, I. Blanco, D. Sanchez, and M.-A. Vila, "Measuring the accuracy and interest of association rules: A new framework," *Intelligent Data Analysis*, vol. 6, no. 3, pp. 221–235, 2001.

[51] Z. Zhou, H. Liu, S. Z. Li, and C. S. Chua, "Rule mining with prior knowledge - a belief networks approach," *Intelligent Data Analysis*, vol. 5, no. 2, pp. 95–110, 2001.

[52] A. Mädche and R. Volz, "The ontology extraction and maintenance framework Text-To-Onto," in *Proceedings of the ICDM'01 Workshop on Integrating Data Mining and Knowledge Management*, F. J. Kurfess and M. Hilario, Eds., November 2001.

[53] D. M. Kitchen and T. Susman, "The animal world at a glance," University of Minnesota, http://www.cbs.umn.edu/class/spring2000/biol/2012/phyglnc.htm.

[54] J. Geller, X. Zhou, K. Prathipati, S. Kanigiluppai, and X. Chen, "Raising data for improved support in rule mining: How to raise and how far to raise," *Intelligent Data Analysis*, vol. 9, no. 4, pp. 397–415, 2005.

[55] N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal, "Closed set based discovery of small covers for association rules," in *Proceedings of the 15èmes Journées Bases de Données Avancées (BDA)*, 1999, pp. 361–381. [Online]. Available: citeseer.ist.psu.edu/pasquier99closed.html

[56] G. Adomavicius, "Expert-driven validation of set-based data mining results," Ph.D. dissertation, Stern School of Business, New York University, 2002. [Online]. Available: citeseer.ist.psu.edu/adomavicius02expertdriven.html

[57] J. F. Sowa, *Knowledge Representation.* Pacific Grove, CA: Brooks/Cole, 2000.

[58] ——, "Distinctions, combinations and constraints," in *Proceedings of the Workshop on Basic Ontological Issues in Knowledge Sharing*, 1995.

[59] R. Wille, "Restructuring lattice theory: An approach based on hierarchies of concepts," in *Ordered sets*, I. Rival, Ed. Dordrecht-Boston, MA: Reidel, 1982, pp. 445–470.

[60] ——, "Concept lattices and conceptual knowledge systems," *Computers and Mathematics with Application*, vol. 23, no. 6-9, pp. 493–515, 1992.

[61] "MediaMark Web site," 2001, http://www.mediamark.com.

[62] H. Gu, M. Halper, J. Geller, and Y. Perl, "Benefits of an OODB representation for controlled medical terminologies," *JAMIA*, vol. 6, no. 4, pp. 283–303, 1999.

[63] H. Gu, Y. Perl, J. Geller, M. Halper, L. Liu, and J. J. Cimino, "Representing the UMLS as an OODB: Modeling issues and advantages," *Journal of the American Medical Informatics Association (JAMIA)*, vol. 7, no. 1, pp. 66–80, January/February 2000.

[64] L. Liu, M. Halper, J. Geller, and Y. Perl, "Controlled vocabularies in OODBs: Modeling issues and implementation," *Distributed and Parallel Databases*, vol. 7, no. 1, pp. 37–65, January 1999.

[65] ——, "Using OODB modeling to partition a vocabulary into structurally and semantically uniform concept groups," *IEEE Transactions on Knowledge and Data Engineering*, vol. 14, no. 4, pp. 850–866, 2002.

[66] J. Geller, H. Gu, Y. Perl, and M. Halper, "Semantic refinement and error correction in large terminological knowledge bases," *Data and Knowledge Engineering*, vol. 45, no. 1, pp. 1–32, April 2003.

[67] H. Gu, Y. Perl, G. Elhanan, H. Min, L. Zhang, and Y. Peng, "Auditing concept categorizations in the UMLS," *Artificial Intelligence in Medicine*, vol. 31, no. 1, pp. 29–44, May 2004.

[68] H. Gu, Y. Perl, M. Halper, J. Geller, F. Kuo, and J. J. Cimino, "Partitioning an object-oriented terminology schema," *Methods in Medical Informatics*, pp. 204–212, 2001.

[69] Y. Perl, J. Geller, and H. Gu, "Identifying a forest hierarchy in an OODB specialization hierarchy satisfying disciplined modeling," in *Proceedings of the 1st IFCIS International Conference on Cooperative Information Systems (CoopIS)*, 1996, pp. 182–195.

[70] H. Gu, Y. Perl, M. Halper, J. Geller, and E. J. Neuhold, "Contextual partitioning for comprehension of OODB schemas," *Knowledge and Information Systems (KAIS)*, vol. 6, no. 3, pp. 315–344, May 2004.

[71] A. Campbell and S. Shapiro, "Ontologic mediation: An overview," in *Proceedings of the Workshop on Basic Ontological Issues in Knowledge Sharing*, 1995, pp. 16–25.

[72] A. Gómez-Pérez and V. R. Benjamins, "Overview of knowledge sharing and reuse components: Ontologies and problem-solving methods," in *Proceedings of the IJCAI'99 Workshop on Ontologies and Problem-Solving Methods (KRR5)*, V. Benjamins, B. Chandrasekaran, A. Gómez-Pérez, N. Guarino, and M. Uschold, Eds., 1999.

[73] "Milonic Web site," http://www.milonic.com/.

[74] "JESS Web site," http://www.jessrules.com/jess/index.shtml.

[75] "WEKA Web site," http://www.cs.waikato.ac.nz/ml/weka/.

[76] C. X. Ling and C. Li, "Data mining for direct marketing: Problems and solutions," in *Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining*, 1998, pp. 73–79.

[77] E. Desmontils and C. Jacquin, "Indexing a Web site with a terminology oriented ontology," in *Proceedings of the International Semantic Web Working Symposium, Infrastructure and Applications for the Semantic Web*, 2001, pp. 549–565.

[78] A. Pretschner, "Ontology based personalized search," Master's thesis, The University of Kansas, Lawrence, KS, 1999.

[79] P. Resnik, "Using information content to evaluate semantic similarity in a taxonomy," in *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, 1995, pp. 448–453.

[80] A. Holt, "The matching and ranking of surface and deep similarities of spatial data," in *Proceedings of the 10th Annual Colloquium of the Spatial Information Research Centre*, University of Otago, Dunedin, 1998, pp. 133–143.

[81] R. Richardson, A. F. Smeaton, and J. Murphy, "Using WordNet as a knowledge base for measuring semantic similarity between words," Dublin City University, Dublin, Ireland, Tech. Rep. CA-1294, 1994. [Online]. Available: citeseer.nj.nec.com/richardson94using.html

[82] C. J. van Rijsbergen, S. E. Robertson, and M. F. Porter, "New models in probabilistic information retrieval," in *British Library Research and Development Report, no. 5587*, 1980.

[83] M. F. Porter, "An algorithm for suffix stripping," *Program*, vol. 14, no. 3, pp. 130–137, 1980.

[84] Y. Lee, K. Supekar, and J. Geller, "Ontology integration experienced on medical terminologies," in *Computers in Biology and Medicine - Special Issue on: Medical Ontologies*, F. Pinciroli and D. M. Pisanelli, Eds., 2006.