Taxonomy Formation as a Restricted Form of Knowledge Discovery in Database

By:
Fatima Bellied Ben Nafi

Supervisor: Dr. Faraj A. El-Mouadib

A thesis submitted to the Department of Computer Science
In partial fulfillment of the requirements for the degree of
Master of Science

Al-Tahaddi University, Faculty of Science
Department of Computer science
Sirte, G. S. P. L. A. J.

Academic year 2005/2006

Faculty of Science

Department of Computer science

Title of Thesis

((Taxonomy Formation as A restricted form of Knowledge Discovery in Database))

By

Fatima Bellied Ben Nafi

Approved by:

Dr. Faraj A. El-Mouadib
(Supervisor) .....

Dr. Abdulhamed Mohamed Abdulkafi
(External examiner) .....

Dr. Idris S. El-Feghi
(Internal examiner) .....

Countersigned by:

Dr. Mohamed Ali Salem

(Dean of Faculty of Science)

Acknowledgement

First and foremost, I would like to thank my advisor Dr. Faraj A. El-Mouadib. It was a real pleasure working with. He was patient and gave me a lot of freedom when I was stumbling around looking for A research problem. he kept my spirits high with his motivation whenever Anything went wrong: he spent a remarkable amount of time mentoring me on a personal Level, he taught me not only how to be a successful Msc student but how to do good science In short, he was an incredible mentor. I would like to thank my supervisor for his patience and support; I would like to thank the entire tutor who assisted me throughout my study . My grateful thanks go to my husband, my mother, father and sister for their love and support. Without them, the completion of this thesis would not have been possible Special thanks to my friends who shared this experience with me.

Fatima

Dedication

To my father for his invaluable support and understanding, and my brothers and Friends for their continued encouraging.

Abstract

In the last three decades or so, the world has seen great advances in the field of computer science. One of the most significant advances is the rapid growth in both generating and collecting data. The widespread of bar code use for most commercial products, the computerization of many business and governments agencies and the advances in data collection tools have provided us with huge amounts of data. Millions of databases have been used and kept in many fields and applications; this is due to the availability of powerful and affordable database systems. The explosive growth in data and databases has generated an urgent need for new techniques and tools that can intelligently transform these data and database into useful information and knowledge. The new techniques and tools have been formulated in what is called Knowledge Discovery in Databases (KDD). KDD is a broad term used to describe various methods and techniques for discovering knowledge in data. KDD is a process to find "interesting patterns" that may be hidden in the data, based on the need and interest of the user. KDD have been known by other names such as; data mining, data archaeology, data dredging, functional dependency analysis, and data harvesting. In fact data mining is one step (the most important one) in the process of KDD. Knowledge can come in many types such as; rules, induction trees, equations and taxonomies. Taxonomy is a hierarchical system that consists of classes, typically arranged in a tree, which is exhaustive and disjoint. Taxonomy formation is unsupervised process that classifies objects or instances in order to form a hierarchical tree. The purpose of this thesis is to demonstrate the implementation of a discovery system that specialized in building binary taxonomy tree. The systems' algorithm

relies only on approximate equivalence relations and partition utility function to build binary Multi-level taxonomy. Our system is analyzed, and designed by the use of Unified Modeling Language (UML) and implemented by the use of VB.NET programming language. After the development of the system, we have tested it by a number of well known data sets of different types and sizes of data. The result that we have obtained are encouraging because they are the same as other researcher's results even though we recommend conducting more experiments with large data and deal with other aspects of problematic data such as; missing values and sparse data.

August, 2006

Table of Content

List of Figures

List of Tables

Chapter One

Introduction

In the last three decades or so, the world has seen great advances in the field of computer science. One of the most significant advances is the rapid growth in both generating and collecting data which have been increasing rapidly. The widespread of bar code use for most commercial products, the computerization of many business and governments transaction, and the advances in data collection tools have provided us with huge amounts of data. Millions of databases have been used in business management, government administration, scientific and engineering data management, and many other applications. It is noted that the number of such databases keeps growing rapidly because of the availability of powerful and affordable database systems. As it has been mentioned in many references (i.e. [7, 13,18]), this explosive growth in data and databases has generated an urgent need for new techniques and tools that can intelligently transform the processed data into useful information and knowledge called Knowledge Discovery in Databases (KDD). KDD is getting to be very popular and has grown recently. The large amounts of data collected and stored might contain some information, which could be useful, but it is not obvious to recognize, nor trivial to obtain by traditional data analysis tools. Neither human ability can sift through such huge amounts of data nor do some existing algorithms have the capability to discover knowledge from such data efficiently. So, the real challenge to KDD is to produce systems that can find "interesting patterns", which may be hidden in the data, based on the need and the interest of the user.

The concept of KDD was first introduced in 1989 to refer to the overall process and discipline of extracting useful knowledge from databases. In the past, KDD have been known by other names such as: data mining, data archaeology, data dredging, functional dependency analysis, and data harvesting [5].

As it has been mentioned in [6], the term *data mining* (DM) has been more commonly used by statisticians, data analysts and the Management Information Systems (MIS) communities; while the term *KDD* has been mostly used by artificial intelligence and machine learning researchers. There is no standard definition to the term KDD, but according to Usama M. Fayyad et al. [5].

KDD is the non-trivial process of identifying valid, novel, potentially useful and ultimately understandable patterns in data.

In general, the knowledge discovered from databases is in the form of pattern. A pattern is a statement in an understandable language, which can be used by human or as an input to another program. "Given a set of facts (data) F, a language L, and some measure of certainty C, we define a *pattern* as a statement S in L that describes relationships among a subset Fs of F with certainty C, such that S is simpler (in some sense) than the enumeration of all facts in Fs." [5].

1.1 Overview of KDD and DM

There is a difference in understanding the term KDD, which is sometimes called DM, between people from different areas contributing to this new field. Here we review some definitions of the term KDD is given in: "The process of extracting valid, previously unknown, comprehensible, and actionable information from large databases and using it to make crucial business decisions."[19]. Another definition of KDD is: "Knowledge discovery in databases (often called data mining) aims at the

discovery of useful information from large collections of data."[5]. A more common definition of KDD is "*The process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data.*"[6].

In the latter definition, the term *pattern* refers to models or structure in the data. The term *process* implies that there are a number of iterative steps involving data such data preparation, search for patterns, knowledge evaluation, and refinement. The process is *nontrivial*, in the sense that it involves the search for structure, models, or patterns. The discovered patterns must be *valid* for new data with some degree of certainty. The discovered patterns must be *Novel* which means that it must be new at least to the system and preferably to the user as well. The discovered patterns should be *potentially useful* to the user or discovery task at hand. For the discovered patterns to be useful, it should be *understandable* by the system and/or human immediately or after some post processing [6].

From all the KDD definitions, one can realize that DM is one stage within the KDD process for extracting useful rules and patterns from the data. Over the years, the DM stage has attracted most of the attention from the research community in the attempt to develop faster, more scalable algorithms to navigate the ever increasing volumes of data in search of meaningful patterns [16].

Just as electrons and waves became the substance of classical electrical engineering, we see data, information, and knowledge as being the focus of a new field of research and application knowledge discovery [5]. Here, we introduce the field of Knowledge Discovery in Databases. As it has been defined by Fayyad et. al. in [5, 18], Knowledge Discovery in Databases, is the process of identifying *valid*, *novel*, *potentially useful*, and *ultimately understandable* structure in data.

The discovered patterns should be *valid* on new data with some degree of certainty. The patterns should be *novel* to the discovery system and preferably to the user. It also should be *potentially useful* to the user and/or to the task. In order for the discovered patterns to accomplish its' purpose, it has to be *understandable* immediately or after some post-processing.

1.2 Necessities of KDD

Raw data stored in databases are rarely visited for direct use and in practical applications: the data are presented to the end-user in a modified, tailored form to satisfy his/her business needs. People usually act as a sophisticated query processor in analyzing the data due to human intellectual capabilities. This method is satisfactory given that the amount of data to be analyzed is relatively small, but is unacceptable for large amounts of data. What is needed in such a situation as an automation of data analysis tasks and that is what KDD and DM exactly provide.

In other words, the goal of KDD and DM is to find interesting patterns that exist in databases but are hidden among the huge volumes of data. The identified knowledge is then used to:

- Make predictions or classifications about new data.
- Explain existing data.
- Summaries of the contents of a large database to facilitate decision making

1.3 Properties of KDD

DM consists of the semi automatic extraction of knowledge from data. This statement allows the question of what kind of knowledge we should try to discover. According to [19], this question is a subjective issue, and we can mention three general

properties that the discovered knowledge should satisfy: it should be *accurate*, *comprehensible*, and *interesting*. Here we briefly discuss each of these properties. The basic idea is to predict the value that some attribute(s) will take on in the future, based on previously observed data. In this context, we want the discovered knowledge to have a high predictive accuracy rate. "We also want the discovered knowledge to be comprehensible for the user". [13].This is necessary whenever the discovered knowledge is to be used for supporting a decision to be made by a human. "If the discovered "knowledge" is just a black box, which makes predictions without explaining them, the user may not trust it" [7]. Knowledge comprehensibility can be achieved by using high-level knowledge representations. A popular method, in the context of DM, is a set of IF – THEN (prediction) rules, where each rule is in the form:

IF <some conditions are satisfied> THEN <predict some value for an attribute>

The third property or issue the interestingness of the discovered knowledge which is very difficult to define and quantify because it is solely subjective depending on the data and the mining task.

1.4 KDD as a process

KDD as a process is an interactive and iterative one, according to [7]; it involves several steps as depicted in the following figure[1]:

[1] The figure is borrowed from [8].

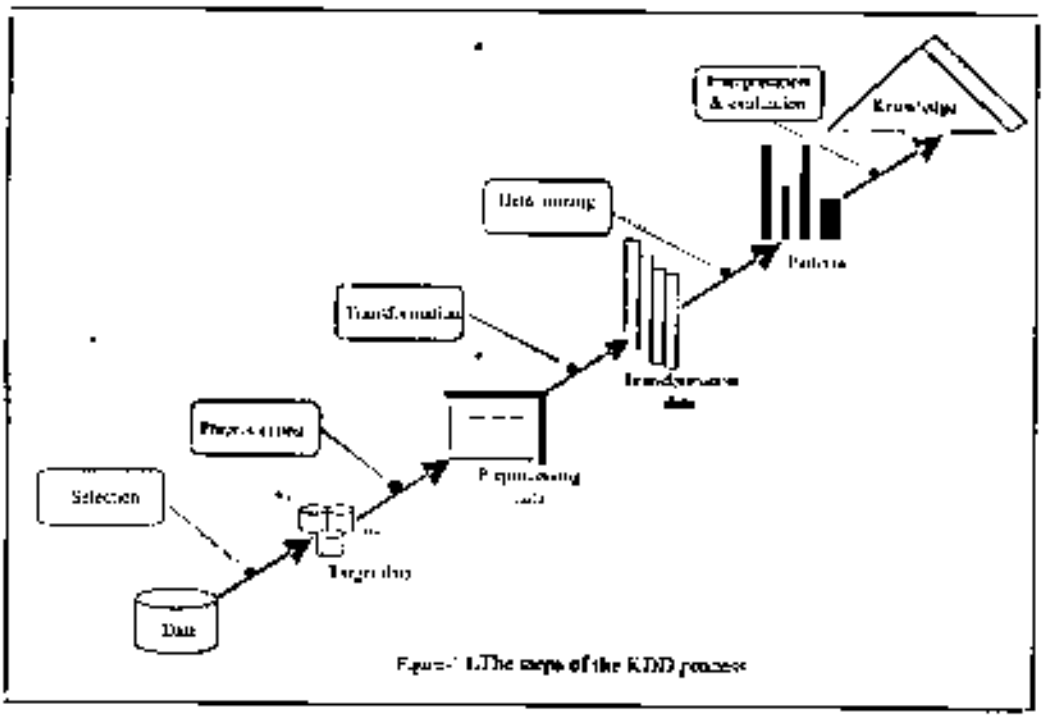


Figure 1.The steps of the KDD process

The data selection step consists of selecting the relevant data to the current mining task (target data) i.e. selecting only a subset of variables and/or data.

The data preprocessing step consists of performing some operations such as noise removal, handling of missing values, data transformation: also known as data consolidation it is a phase in which the selected data is transformed into forms appropriate for the mining procedure. Data from real-world sources are often erroneous, incomplete, and inconsistent, perhaps due to operation error or system implementation flaws. Such low quality data needs to be cleaned prior to data mining.

The DM step, which is the most important one, consists of an algorithm that searches for patterns of interest in a particular representational form.

The interpretation and evaluation step includes interpreting the discovered patterns and to determine the interestingness of the discovered patterns given some threshold, which is controlled by the user. The discovered patterns can analyzed automatically or semi automatically to identify the truly interesting/useful patterns for the user.

Iterations and loops can occur practically between any step and any preceding one.

1.5 DM Tasks

At the core of the KDD process are the DM tasks for extracting patterns. These tasks can have different goals, depending on the intended overall KDD process. We briefly review some of these tasks. Other DM tasks are briefly discussed in [5]. DM tasks can be categorized into two types; prediction methods and description methods.

In prediction methods, some variable values are used to predict unknown or future values of other variable values. Some examples of prediction methods are; Classification, Regression, and Deviation detection.

1. Classification is a model function that classifies a data item into one of several predefined categorical classes. Classification is probably the most studied DM task.
2. Regression is the analysis of the dependency of some attribute values upon the values of other attributes for the same data object. This model is mostly used by statisticians.
3. Deviation Detection focuses on discovering the most significant changes in the data from previously measured or normative values.

In description methods, the discovered patterns that describe the data are interpreted humanly. Examples of description methods are; clustering and Association discovery.

1. Clustering is a function that maps a data object into one of several clusters, where these clusters are natural groupings of data objects based on similarity metrics or probability density models. Clustering has been used in many displaces such as Biology, zoology, etc.

2. Association discovery is a model that identifies relationships between attributes and/or objects such as the presence of one pattern implies the presence of another pattern.

1.6 KDD and related other fields

According to [13], KDD is an interdisciplinary field that relates many fields such as; statistics, machine learning, databases, and knowledge acquisition for expert systems. KDD systems typically draw upon methods, algorithms, and techniques from these diverse fields. The unifying goal is extracting knowledge from data in the context of large databases. All areas aim at locating interesting regularities, patterns, or concepts from empirical data.

Statistics have a very important role in most discovery systems dealing with large amount of data. Statistical methods of exploratory analysis are quite good in describing and organizing data, although the traditional ones are not designed to discover explicit rules. Statisticians are concerned with numerically describing the strength of association between attributes, finding patterns in data, explaining variability in data, fitting models, and assessing significant departure from the model. Statistical techniques require human intervention for the interpretation of the findings and the human analysis of data becomes troublesome and error prone as the quantity of data increases. Even though statistics provides good methods for the problem of data analysis such providing the appropriate measures and tests, pure statistical methods on their own do not serve directly the purposes of KDD for the following reasons:

- In statistics, the available domain knowledge is not used efficiently.

- Statistical results can only be interpreted by statisticians or some one with good statistical background.
- Statistical methods require guidance from an expert for their application and usage.
- The application statistical methods results require some guidance for their usage.

Machine learning is data mining process that improves knowledge through experience. Humans can learn from their experience so that they perform the same task better in the future and do not commit the previous mistakes.

The experience can be acquired either from a teacher (supervised learning) or through self-study (unsupervised learning). Machine learning is a process of attempting to simulate human learning process through computer programs such as expert systems.

The aim machine learning algorithm is to learn from experience and build a model that can be used to perform similar tasks in the future more efficiently.

Database management systems (DBMS) are a system especially developed for the storage and retrieval of large masses of data, and it consists of a collection of procedures for the data management operations (i.e. retrieval, storage, and update...). A DBMS is thus a collection of procedures that provide tools for extracting tuples satisfying common conditions, which resemble the discovery ability to produce interesting and useful statements. So, in principle they should be suitable for KDD, including data mining. "A database mining system must be open to many different kinds of patterns in two or more dimensions and must consider very large number of data subsets in which a pattern might appear. On the other hand DBMS tools lack the ability to determine what computations should be done and the evaluation of the

significance of the derived patterns." [4]. Relational databases have a simple uniform structure; a real world database presents big challenges to the automated discovery system due to the huge number of records, attributes and to the variety of attribute types.

Knowledge acquisition for expert systems attempt to capture knowledge relevant to a specific problem which can be acquired from an expert via some knowledge extraction techniques. The knowledge extracted from expert is much higher in quality than the data in databases even though they only cover important cases.

Scientific Discovery (SD) may not directly apply to databases due to the quality of data. Scientific discovery requires high quality data that comes from some controlled environment with the emphasis is on a few parameters on the other hand the data in database comes from the real world which usually noisy and inconsistent. "In scientific discovery data fields can be redesigned and data can be recollected but in the real world databases the redesign and recollection is not an easy task"[4].

Domain knowledge is an important component of a typical DM system. An important issue within DM algorithms is the beneficial use of domain knowledge. "Domain knowledge is knowledge about the specific domain characterized by the database as a whole rather than information about the data itself, such as field width or attribute values."[18].The purpose using domain knowledge within a mining algorithm is to improve efficiency and focus the search for patterns. "The primary purpose of domain knowledge is to bias the search for interesting patterns. This can be achieved by focusing attention on portions of the data, biasing the extraction algorithms, and assisting in pattern evaluation."

1.7 Motivation of the work

The purpose of this thesis is to implement a computer system for KDD that specializes in Taxonomy Formation by the use of the algorithm developed in [4]. In the systems' analysis and design, UML[2] will be used and the implementation of the system will be carried out by the use of VB.NET programming language.

1.8 Thesis organization

The rest of this thesis is organized as follows: Chapter two of this thesis presents a review of three knowledge discovery systems. Each system employs different methodologies and techniques in the process of knowledge discovery. The first system is Induction of Decision Trees (ID3) developed by John Ross Quinlan, which uses statistical approach in the mining process for knowledge. The second system is The COBWEB system, developed by Fisher [1987], the third system is Forty-Niner, of J. M. Żytkow and Zembowicz in 1993,which applies modern techniques of machine learning and discovery to databases. Chapter three describes the design and implementation of our system for KDD that specializes in Taxonomy Formation. Chapter four describes various experiments that were conducted and the results obtained from these experiments and finally Chapter five summarizes the conclusion and further works.

[2] UML stands for Unified Modeling Language.

Chapter Two

Typical KDD Systems

This chapter presents a review of some of well known KDD systems. These systems employ different techniques in the process of KDD. The first system is Induction of Decision Trees (ID3) which has been developed by John Ross Quinlan in 1986. ID3 is a classification system that works in a supervised fashion. The second system is called COBWEB which was developed by Douglas H. Fisher in 1987. The COBWEB system is an incremental conceptual clustering one that works in an unsupervised fashion. The third system is called Forty-niner (49er), which was developed by Żytkow and Zembowicz in 1993. The task of the 49er system is to discover knowledge in the form of taxonomy.

The rationale behind choosing the above mentioned systems is summarized in the following points:

1. All the systems are related to the work to be carried out in this thesis in one way or another.
2. All the systems are used for the task of classification and the production or classification rules.
3. The systems 49er, COBWEB and our system find knowledge in an unsupervised fashion.
4. The systems 49er and our system are non-incremental systems.
5. The systems 49er and our system are used to discover knowledge in the form of taxonomy of classes.
6. We study the ID3 system for the sake of completeness because it operates in

supervised fashion (due to the fact that the learning task can be supervised or unsupervised).

2.1 The ID3 system

ID3 system is a non-incremental algorithm that constructs decision tree from a set of examples (call training set). A decision tree is a flowchart like tree structure, where the topmost node in a tree is the root node, the internal nodes denote a test on an attribute (decision nodes), each branch represents an outcome of the test and the leaf nodes represent classes (class nodes) or class distributions. The decision tree is constructed in a Top-Down fashion, starting at the root node and continuing to the leaves. The training set can be given in a relational database relation that consists of a set of tuples. Each tuple consists of a number of values and an additional attribute called class. Each attribute of the relation has its' domain (list of allowable values), which is usually limited to a small number of values. The system ID3, by Quinlan, is one of the systems that had great impact on machine learning research in the beginning of the eighties [23].

2.1.1 ID3 algorithm

The ID3 algorithm is used to generate all possible simple decision trees that correctly classify all objects in the training set. In the construction of decision trees, the algorithm ID3, works in an iterative fashion guided by two features. These features are being; windowing and information theoretic heuristic. Windowing in ID3 terms is to select a random subset of the training set to be used to build the initial tree. The remaining cases of the training set are then classified using the initial tree. If the initial tree gives the correct classification for all of the cases in the training set then it is accepted and considered to be the final tree and the algorithm terminates. If some of

the training cases are misclassified then these cases are appended to the window and the process will be started from the beginning until the tree gives the correct classification for all of the cases in the training set. The purpose of the information theoretic heuristic (Entropy) is to decide the order in which the attributes are to be selected to be used in building the tree. The input to the algorithm is a relational database and the output is a decision tree. The following figure 2.1[1] depicts an example of a simple decision tree that is used to classify whether it is a good day for playing golf or not.

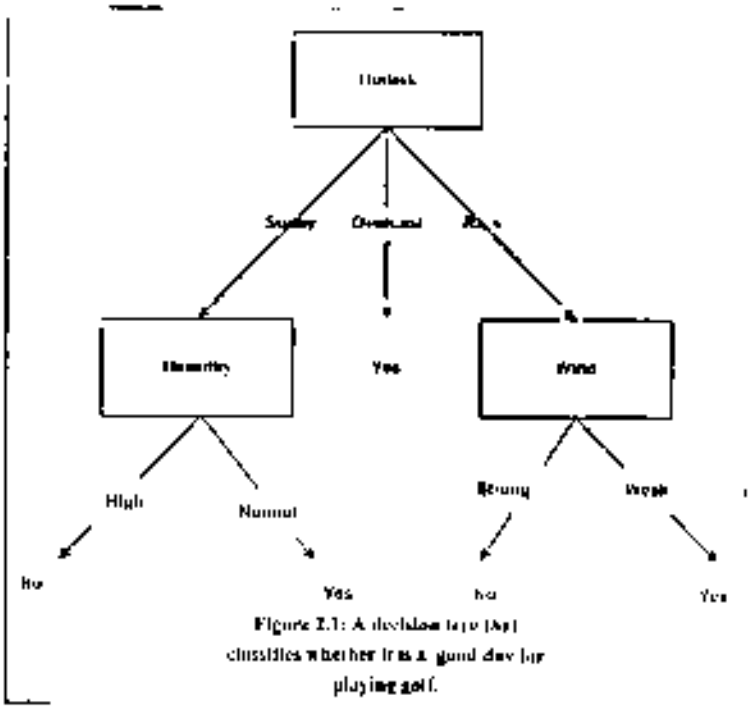


Figure 2.1: A decision tree [23] classifies whether it is a good day for playing golf.

2.1.2 Search algorithm

The ID3 system uses a top-down irrevocable strategy that searches only part of the search space, which consists of all trees that can be constructed with attributes-values combination in the test set that guarantees, that is simple but not necessarily the simplest tree is found. In ID3, a tree is constructed as follows:

1. The algorithm selects the best attribute to be used as the root node of the

[1] The figure is borrowed from [23].

decision tree. The best attribute is the one which has the highest information gain value. The branches of the root node are labelled by the different values of the selected attribute.

2. On each branch of root node a sub decision tree is developed in the same fashion as in the root node.

3. Step 2 is performed on each branch until each branch can not be developed any further (all cases belong to the same class) and then that branch will be labelled with that class.

4. The produced tree is used to classify all the cases in the training set. If all cases are correctly classified the algorithm terminates otherwise the window is updated and the process starts all over from the beginning [22].

The above algorithm guarantees that it will create a tree that classifies all cases in the training set.

In order to explain the information theoretic heuristic (Entropy) let's consider the following example: let class P which contains all positive examples and class N which contains all negative examples. Let the set of examples S contain p cases of class P and n cases of class N. The information gain is calculated by the following:

$$I(p,n) = -\frac{p}{p+n}\log_2\frac{p}{p+n} - \frac{n}{p+n}\log_2\frac{n}{p+n}$$

Note that $I(p, n)$ depends on p and n only, and that $I(p, n) = 0$ if $p = 0$ and/or $n = 0$, otherwise $I(p, n) > 0$. Assume that Attribute A is used in the root node, then it will partition S into subsets $\{S_1, S_2, \ldots S_v\}$. If S_i contains p_i examples of P and n_i examples of N, then the information needed to decide if an element in S_i belongs to P or N is $I(p_i, n_i)$. So the information needed to classify an element of S using a tree with

attribute A as root is the weighted average of the information needed to classify objects in all subtrees S_i is given by the entropy of A:

$$E(A) = \sum_{i=1}^{v} \frac{p_i + n_i}{p + n} I(p_i, n_i)$$

The attribute A is selected when the information gain is maximal, which means that the $E(A)$ is minimal. However, a known problem with this criterion is that it tends to favor attributes with many values. Different solutions to this problem are presented in [23].

2.1.3 Numerical attribute

Condition on an attribute, i.e. an internal node of the decision tree, is a test on the value of an attribute, with branches for all possible values. Although this is convenient for symbolic attributes, it would be useful if one could test on ranges of numerical continuous attributes. An extended version of ID3 algorithm, called C4.5 in [24], deal with numerical continuous attributes which allows tests on the inequality of numerical attributes, such as $A_i \leq N$ and $A_i > N$, with two possible outcomes (branches).

2.1.4 Noise

Non-systematic errors in both attribute values and class information are referred to as *noise*. The noise affected data can cause two serious problems to tree building algorithm (for more details see [21]). A corrupted set of examples causes any or both of the following problems:

1. It is not possible to generate a tree that classifies all examples correctly, i.e. the consistency condition is violated (classes overlap). This means that some set S_i contains objects that do not belong to the same class, and there are no

attributes that can be used to further branch at this leaf.

2. Corrupted examples can cause the tree to grow too indefinitely to accommodate such examples. This problem occurs when the values of an attribute A are corrupted for some examples, branching on A might give an apparent information gain, even though values in attribute A are random, and form no indication for the classification.

2.1.5 Missing attribute values

Another problem that common in the real world data is the missing attribute values which in turn will cause a problem in the construction of a decision tree. Such problem arises when we attempt to classify an object with missing some of its attribute values. In order to construct a tree with missing attribute values, several methods have been proposed in conjunction with missing values such as:

1. Replace the missing values with the most appearing value in their class, or
2. Simply discard examples with missing values, or
3. We can treat missing values as a special unknown value.

However, the latter technique increases the expected information gain for an attribute if some values are unknown, which is not a desirable property. Other proposed techniques are based on probable estimation.

2.1.6 Windowing

The idea behind the windowing is that in the process of building the decision tree not all the training set examples will be used. Instead of using the entire training set in the process of building the decision tree, a randomly chosen subset called window can be used for that purpose. Using this window, a tree is generated, and all examples in the training set are classified using this tree. If all of the training set examples are

correctly classified then it is ok, otherwise misclassified examples are marked and then added to the window and the process of building the decision tree is started all over again until all the examples in the training set are correctly classified[23].

2.1.7 Incremental learning

The ID3 algorithm constructs decision tree in a non-incremental fashion. This means that the entire training set is supplied at once. However, if the examples are supplied once at a time, the ID3 algorithm can still be used, but it would construct a new decision tree from scratch, every time a new example is observed. For such serial learning task, one would prefer an incremental algorithm, on the assumption that it is more efficient to revise an existing tree than it is to generate a new tree every time [22].

2.2 The system COBWEB

The COBWEB system, developed by Fisher [1987], is conceptual clustering system that produces a probabilistic classification tree. The system works in unsupervised incremental fashion. The classification tree produced by COBWEB is probabilistic one. And all points which are classified under that particular node, is also included a probabilistic description of that data object as well. COBWEB uses a category utility to mange the construction of the tree. From there cluster are formed based upon certain similar measures which are the interclass similarity and the interclass dissimilarity, which are both probable describing the sharing of attribute values amongst nodes in the class.

2.2.1 Overview of the system COBWEB

COBWEB is a conceptual clustering system which constructs a classification tree through incremental incorporation of new instances in classification tree. Each Node

of classification tree represents concept as the probability of the occurrence of each Attribute-value pairs present in that concept. The root node of the tree includes all information of all instances to be classified. As we descend the tree, concepts within nodes become more specific. A new instance is placed into a node (class) of the classification tree. The incorporation of an object is accomplished via the use of one of four operators. The purpose of these operators is to classify an object with respect to an existing class or to create a new class for the given object or to combine two classes into one or to divide a class into several classes. The selection of which operators to use is guided by category utility function [9].

2.2.2 Category utility

The system COBWEB use category utility in order to confirm the traditional virtue held in clustering (objects within the same class should be as similar to each other as possible and objects in different class should be as dissimilar as possible)."In particular, the category utility is a trade-off between intra-class similarity and inter-class dissimilarity."[9].

Inter-class similarity is a function on the form of $P(C_k | A_i=V_{ij})$. The larger value this probability, the fewer the objects in contrasting classes that share this value. Intra-class similarity (similarity between objects within class) is reflected by the conditional probabilities of the form $P(A_i=V_{ij}|C_k)$, where Ai=Vij is an attribute-value pair and C_k is the class. According to [11], these two probabilities are combined to give an overall measure of partition quality, where a partition is a set of mutually exclusive classes $\{C_1, C_2 \ldots C_n\}$. The measure

$$\sum_{k=1}^{n} \sum_{i} \sum_{j} P(A_i = V_{ij}) P(C_k | A_i = V_{ij}) P(A_i = V_{ij} | C_k)$$

is a trade-off between intra-class similarity (through $P(A_i=V_{ij}|C_k)$) and inter-class dissimilarity (through $P(C_k|A_i=V_{ij})$ that is summed across all classes (k), attribute (i), and value(j). the probability $P(A_i=V_{ij})$ weights the importance of individual values, in essence saying it is more important to increase the class-conditioned predictability and predictiveness of frequently occurring values than for infrequently occurring ones. The formula can be rewritten using Bayes' rule:

$$P(A_i = V_{ij})P(C_k \mid A_i = V_{ij}) = P(A_i = V_{ij})\frac{P(c_k, A_i = V_{ij})}{P(A_i = V_{ij})}$$

$$= P(c_k, A_i = V_{ij} \mid C_k)$$

$$= P(c_k)P(A_i = V_{ij})$$

Can be rewritten as:

$$\sum_{k=1}^{n} P(C_k)\sum_{i}\sum_{j} P(A_i = V_{ij} \mid C_k)^2$$

In other words, is the expected number of attribute values is that can be correctly guessed for an arbitrary member of class C_k. This expectation assumes a guessing strategy that is probable matching, meaning that an attribute value is guessed with a probability equal to its probability of occurring. Thus, it assumes that a value is guessed with probability $P(A_i = V_{ij}|C_k)$ and that this guess is correct with the same probability.

Finally, define category utility as the increase in the expected number of attribute values that can be correctly guessed ($P(C_k)\sum_{i}\sum_{j} P(A_i = V_{ij} \mid C_k)^2$) given a partition $\{C_1, ...,C_n\}$ over the expected number of correct guesses with no such knowledge ($\sum_{i}\sum_{j} P(A_i = V_{ij})^2$).More formally, $CU(\{C_1,C_2, ...C_n\})$ equals:

$$\frac{\sum_{k=1}^{n} P(C_k)\left[\sum_{i}\sum_{j} P(A_i = V_{ij} \mid C_k)^2 - \sum_{i}\sum_{j} P(A_i = V_{ij} \mid C_k)^2\right]}{n}$$

Where, for each of the N classes, $p(A_i=V_{ij}/C_k)$ is the conditional probability that the attribute Ai has the value V_{ij}, given membership in the class C_k, and $p(A_i=V_{ij})$ is the prior probability that the attribute A_i has the value of V_{ij}.

The probabilities are summed over all possible combinations of attributes and values. The maximum value of this function gives the best classification.

2.2.3 Representation of concepts

The input to COBWEB is a set of attribute-value pairs for each example. COBWEB incrementally incorporates these examples into a classification tree, where each node is a probabilistic concept that represents an item class. An example of COBWEB input is depicted in table 2.1[2] which consists of four data attributes and one class attribute.

Name	Bodycover	Heartchamber	Bodytemp	Fertilization
Mammal	Hair	Four	Regulated	Internal
Bird	Feathers	Four	Regulated	Internal
Reptile	Cornfield-skin	Imperfield-four	Unregulated	Internal
Amphibian	Moist-skin	Three	Unregulated	External
Fish	Scales	Two	Unregulated	External

Table-2.1: "Animal" attribute-value pairs.

The output of COBWEB is a classification tree as the one show in figure2.2, which is the classification tree of the data of table 2.1.

[2] This table is borrowed from [9]

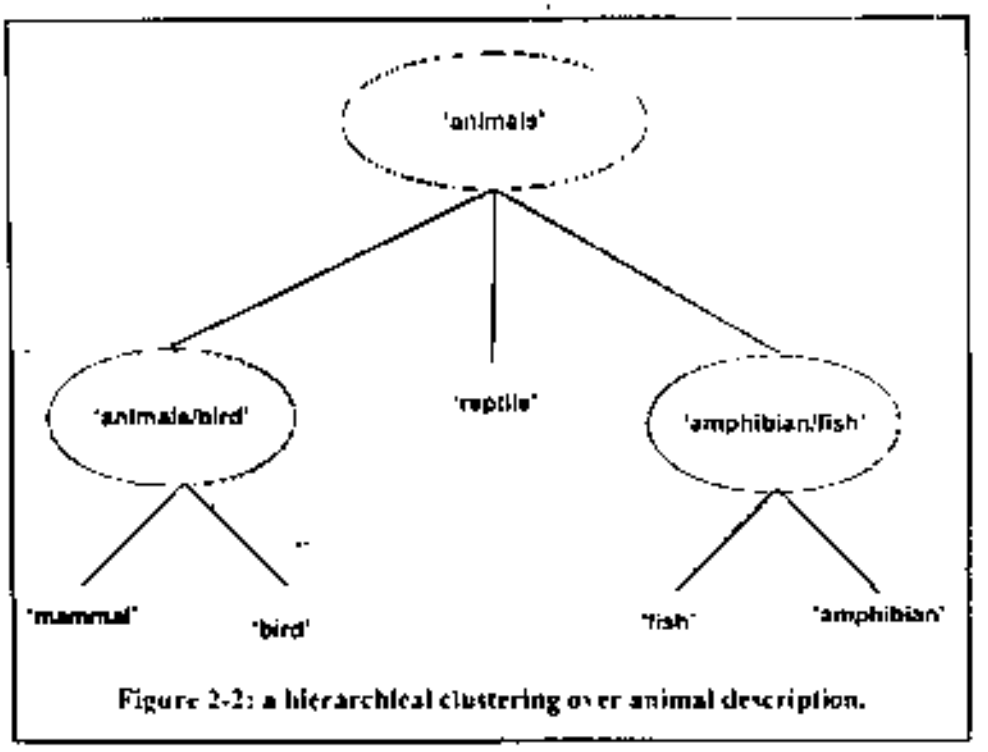


Figure 2-2: a hierarchical clustering over animal description.

The attribute value pairs that are observed in the incoming example are used to calculate the various probabilities, and hence the category utility functions. For example, the value of P(bodycover = feathers|bird) is calculated by:

$$\frac{(\#\text{ Examples with attribute bodycover = feathers})}{(\#\text{ Examples in total})}$$

where both counts are measured at the node corresponding to the concept "bird". Each node in the tree defines a probabilistic concept and provides a summary of the examples classified under the given node [10].

2.2.4 Operators

To incorporate new instance (objects) into the classifications tree, COBWEB descends the classification tree along the appropriate path, updating the example counts the way, and performs one of several operators at each level. The operators can be one of the followings:

1. Incorporate operator: adds new example (object) to an existing class.

2. Create new operator: creates a new class.

3. Merge operator: combines two classes into single classes.

4. Split operator: divides an existing class into several classes.

The two operators (Merge and Split) allow merging and splitting of existing classes and so COBWEB can move bidirectionally through the space of possible hierarchies, adjusting "poor" clustering due to unrepresentative initial instances. Thus the search through the concept space is via a heuristic called hill-climb method [12].

2.2.5 The COBWEB algorithm

The algorithm of COBWEB is a nested recursive procedure, which accepts new examples one by one. The algorithm traverses the classification tree along a particular path until the new example is incorporated into the tree. The COBWEB algorithm effectively performs a top-down, unsupervised, incremental hill-climb search through the concept space, which in turn produces a hierarchical probabilistic organization of concepts. The following pseudo code demonstrates the control strategy of COBWEB:

Function Cobweb (object, root)

1. Incorporate objects into the root cluster:
2. If root is a leaf then
   return expanded leaf with the object;
   else choose the operator those results in the best clustering:
   a) Incorporate the object into the best host;
   b) Create a new classes containing the object;
   c) Merge the two best hosts;
   d) Split the best host;
3. If (a) or (c) or (d) then
   call Cobweb (observation, best host);

According to [9], COBWEB searches for classification in which the first level of the classification tree is optimal with respect to a measure of clustering quality, on the other hand COBWEB is sensitive to the order of examples and can get stuck.

2.3 FORTY-NINER (49er) system

The system Forty-niner (49er) has been developed by Zytkow and Baker in [1991] as computer system that mines useful knowledge from relational database in the form of regularities. The system works in unsupervised fashion.

2.3.1 An overview of the 49er system

The system Forty-niner (49er) concentrates mainly on discovering knowledge in the form of regularities. Regularity is a common pattern in sets of data, analogous to scientific laws. Regularity can be expressed in the form of:

PATTERN in RANGE,

where PATTERN describes a pattern that holds for a subset of records of the database, which is known as the RANGE of the pattern. More formal definition of regularity in 49er can be found in [30]. Regularities are very useful means in the sense that they allow users to make prediction and explanations. "Regularities can be compared by their generality, determined by their RANGE, and by their strength of their PATTERN". [29].

The basic architecture of 49er as depicted in figure 2.3[3], consists of the following components:

i. Database,
ii. Two search module.
iii. Regularity evaluator.

[3] This figure is borrowed from [30].

iv. Regularity expander.

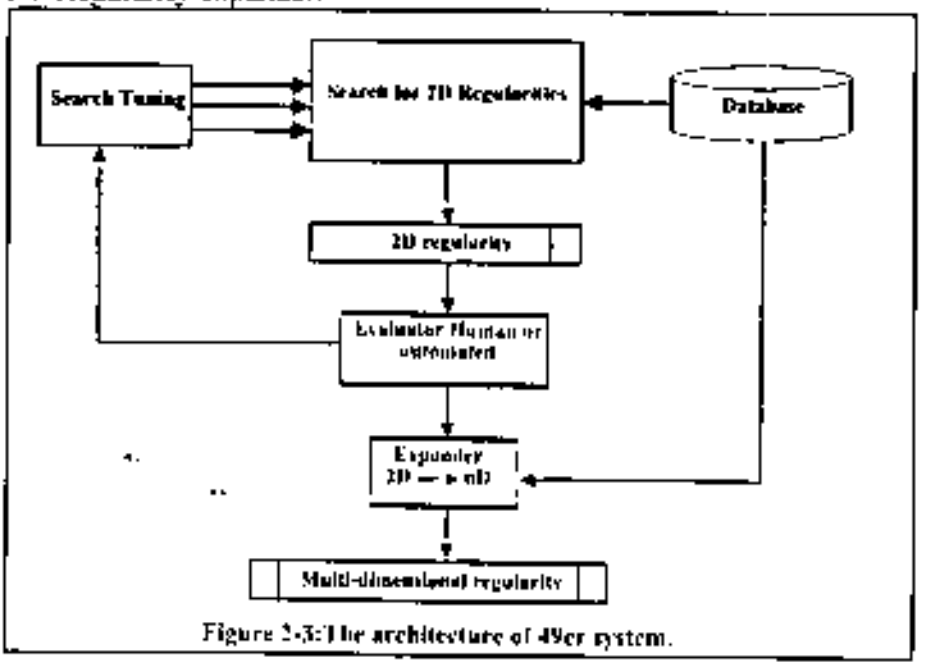


Figure 2-3: The architecture of 49er system.

The system's database consists of equal length records, each of which is composed of the same sequence of attributes. Each value is nominal numeric and any missing values are discarded.

Generally, the system operates in two search modules:

The first search module: conducts an automated search for 2D regularities in a large hypothesis space. The search space is determined by considering many subsets of data by Partitioning the values of an attribute, combinations of attributes in an exhaustive search to find two-dimensional patterns. The purpose of the first search module is to find preliminary regularities according to some threshold.

Second search module: refines the discovered 2D regularities by strengthen and/or generalize procedures. Each search module can be repeated as many times as the user desires and in each cycle the user can change search parameters and continue exploration.

Regularity evaluator: for each pair of attributes in the database, 2D contingency table is formed. Then each 2D contingency table is tested (by means of some statistical test) if it represents 2D regularity. Any found 2D regularity is accepted or rejected (according to user provided threshold).

Regularity expander: the purpose of this module is to expand each found 2D regularity to more dimensions by adding one dimension at a time. "Multidimensional regularities are more general than 2D regularities"[30].

2.3.2 The search for regularities

Regularities in the system 49er are accepted according to their significance. Regularity significance is measured by the probability that it is not random fluctuation.

The system 49er uses probability Q as a measure of significance of regularity. "Regularities can be compared by their generality, determined by their RANGE, and by the strength of their PATTERN." [30]. Greater generality, which means that regularity covers a large fraction of the population and greater predictive strength means that concrete prediction can be produced from that regularity. The system 49er uses: χ^2 test, Cramer's V coefficient and contingency coefficient C.

The Cramer's V coefficient and contingency coefficient C are based on χ^2 statistics. The Cramer's V coefficient is used by 49er as a measure of predictive power of regularity. Large value of V means that there is a strong correlation between the two variable of the contingency table. The ideal correlation (V=1) occurs when any particular value of one attribute there is one and only one value for the other attribute. Cramer's V is confined to interval [0, 1] despite of the contingency table size.

Let N be the total number of observations, α and β be the number of rows and the number of columns of the contingency table respectively, the Cramer's V coefficient is defined as:

$$V = \sqrt{\frac{\chi^2}{N * \min(\alpha - 1, \beta - 1)}}$$

Another measure of strength is the contingency coefficient C, which is based on the χ^2. The contingency table coefficient is defined as:

$$C = \sqrt{\frac{\chi^2}{\chi^2 + N}}$$

"49er computes all three measures: χ^2, V, and C, but leaves it to the user to draw conclusions about acceptance of regularities detected in the first phase and selection of regularity refinements." [34].

The measures Q, V and C are used to help the user to accept or reject regularities. For example if regularity has very small value for Q and large values for V (or C) then it probably be accepted.

It could happen that the number of values of an attribute in a contingency table are too many and in turn it will be too costly to deal with, in such cases the system 49er computes contingency-2 (the smallest 2×2 contingency table) by aggregating the values of original contingency table (contingency-all) which will be easier and cheaper to deal with as far as computation concern.

The first search phase of 49er can find a large number of 2D regularities. Such founding regularities are evaluated by some statistical tests such χ^2-test, Cramer's Coefficient V, and Contingency table strength C. For more detail of these tests see [30]. And it is left to the user to accept or reject the regularities interactively. "By

default. 49er initially looks for simple contingency tables, but realizing that the data follow specific patterns, the user may apply more subtle and more costly mechanisms focused on those patterns." [30].

After 2D regularities have been found, they can be expanded to more dimensions by adding one dimension at a time. Multidimensional regularities are more general than 2D regularities. "One n dimension regularity can replace several n-1 dimension regularities." [30].

Given the list of all 2D regularities that have been produced by the first search (successfully met the user threshold), the regularity expansion module can be used to extend regularities to more dimensions by:

1. Extending the dimensionality of each 2D regularity by trying to include every other attribute, one at a time, or
2. Increasing the empirical content of regularities via combing several related regularities into new regularity with common attributes.

49er's architecture combines the advantages of large-scale automation and human intervention. It is a non-incremental system allowing user interaction to guide discovery in intermediate stages.

2.3.3 Operations on attributes

49er system can operate on many attributes of different values and makes the necessary pre-processing on such value before the search for regularities can commence. In dealing with different scaled attribute such as; *nominal, ordinal, interval, and proportional*, the system 49er typically uses three operations namely; aggregation, slicing, and projection [28].

Aggregation operation combines values V_i of an attribute A_i into classes of abstraction. This is done to reduce the size of the search space. The default number of aggregates used by 49er is two (low and high). "The system tries to ensure that the sets of records corresponding to both aggregates are of equal size, making the best approximation."[28].

Projecting an attribute A_i is the same as ignoring or removing the attribute A_i from the database. All attribute in the database are considered for this operation. Removing an attribute from the database will save some space but it will take time. On the other hand ignoring an attribute will take no time but it will not save any space. The purpose of this operation is to reduce the size of the search space by the reduction of the database dimensionality.

When used with aggregation, it is helpful in reducing the sparseness of data in the entire set.

A partitioning search is used to produce subsets of the database to be searched for two dimensional regularities. Partitioning generates all combinations of the given independent and dependent variables using the basic operations described above.

Slicing operation is a combination of the two previous operations (selection and projection) performed on the same attribute.

2.3.4 49er enhancements

The original system 49er has been enhanced in several ways to cover broader functionality aspects of KDD. The new version 49er.b (depicted in figure 2.4) has been improved in the following aspects:

1. The use of domain knowledge to guide the search for regularities,

2. The discovery of equations by the incorporation Equation Finder (E.F.) module, and

3. The systematic use of chi-square test and regularity refinement.

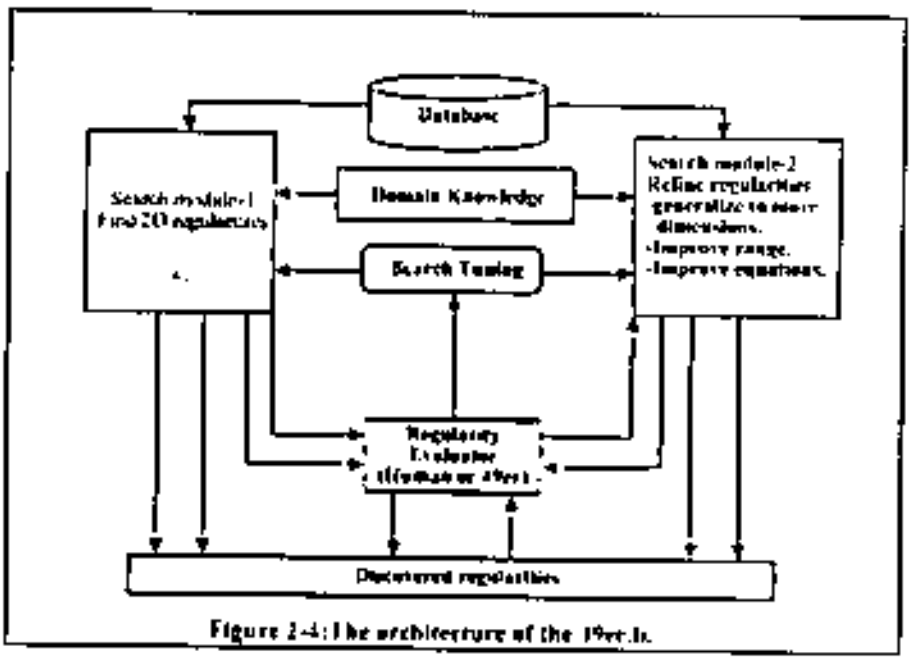


Figure 2-4: The architecture of the IDS-II.

Thick arrows indicate flow of control and thin arrows indicate flow of data.

Chapter Three

Design and implementation

In this chapter, we will describe the design and implementation of our system for KDD that specializes in Taxonomy Formation. The algorithm for this system was developed in [4]. A detailed description of the algorithm and its' various components will be given. Details of the measures used in the system to accomplish its task will be explained. A general review of the language, database and the data structure used in the implement will be given.

3.1 Taxonomy formation algorithm

Taxonomies have been used in so many areas of science in order to make classification, i.e. Botany, Zoology and other disciplines. The term taxonomy comes from the Latin word with two syllabus "taxis" and "anomy". The first syllabus "taxis" mean an arrangement or order, and the second syllabus "anomy" refers to knowledge. So, the term taxonomy means n knowledge that is arranged in some kind of order. Taxonomy can be defined as hierarchical system of selected subsets of an attribute values arranged in a tree structure that is exhaustive and disjoint. As defined by Klosgen and Zytkow [25], taxonomy is restricted form of knowledge by the hierarchical organization we mean that one in which subclasses possess the discriminating features of their super-classes, and classes which are the siblings in the hierarchy are mutually exclusive with respect to the presence or absence of some set of features.

Our taxonomy formation system for multi-level taxonomy works in an iterative fashion and is based on the following main algorithm:

Iterate the following steps until there are no more (approximate) equivalence relations.

1. Find (Approximate) Equivalence Relation (Pattern).
2. Create one-level-taxonomy.
3. Merge similar (equivalent) one-level-taxonomies.
4. Choose the appropriate partition.
5. Build Multi-level taxonomy tree.

3.2 Taxonomy formation algorithm details

The algorithm consists of a number of iterative steps, each of which accomplishes a certain task. In the following subsections details of each step is given.

3.2.1 Find (Approximate) Equivalence Relation

After reading the data file (database) or transforming a text file to an Access database file, the system will produce contingency tables for all combinations of attribute pairs in an exhaustive search. A contingency table is a two-dimensional (two attributes) table of counts. One dimension determines the row attribute values; the other dimension defines the column attribute values. The combinations of row and column categories are called cells. Contingency tables are used here to find the relationship between two attributes. The use of contingency tables is due to:

1. They are simple to form.
2. They are easy to understand.
3. They appeal to people that do not have good knowledge of statistics.
4. The data type is independent, so they can be used with any type of data such as: nominal, ordinal, interval, etc.

Each contingency table will be tested if it represents regularity. The condition for finding regularity is based on a measure called Lambda (Goodman 1952) and a user pre-specified threshold. If λ_a and/or λ_b exceeds the user pre-specified threshold then

the contingency table represents regularity. Details of the set of Lambda measures will be given later.

In order to find (approximate) equivalence relation, another two steps are performed on each contingency table; namely Correspondence Analysis (CA) and Aggregation.

3.2.1.1 Correspondence analysis

Correspondence analysis is a statistical technique, which finds a multidimensional representation of the association between the row and column categories of a two-way contingency table. We use the principle of correspondence analysis to guide the rearrangement of rows and/or columns of a contingency table so that similar rows and similar columns will be adjacent [20]. The following figure 3.1 depicts the correspondence analysis step application for any two attributes A1 and A2:

Contingency table before correspondence analysis

A1 \ A2	1	2	3	4
1	12	0	5	0
2	25	0	8	0
3	0	3	0	11
4	0	2	0	9

Contingency table after correspondence analysis

A1 \ A2	1	3	2	4
1	12	5	0	0
2	25	8	0	0
3	0	0	3	11
4	0	0	2	9

Figure 3-1: The application of the correspondence analysis step.

3.2.1.2 Aggregation

The purpose of this step is to reduce any given two-dimensional contingency table to a two-by-two (2 × 2) contingency table by merging values of the two attributes involved. The aggregation of values is performed in such a way that the resulting 2 × 2 table has the largest possible value of λ. If the value of λ is sufficiently close to 1.0, then the (approximate) equivalence described by the 2 × 2 table provides a natural basis for a binary split of the data (i.e., creation of a hierarchy element). For example,

the figure 3.2 depicts the application of the aggregation step for any given two attributes A1 and A2:

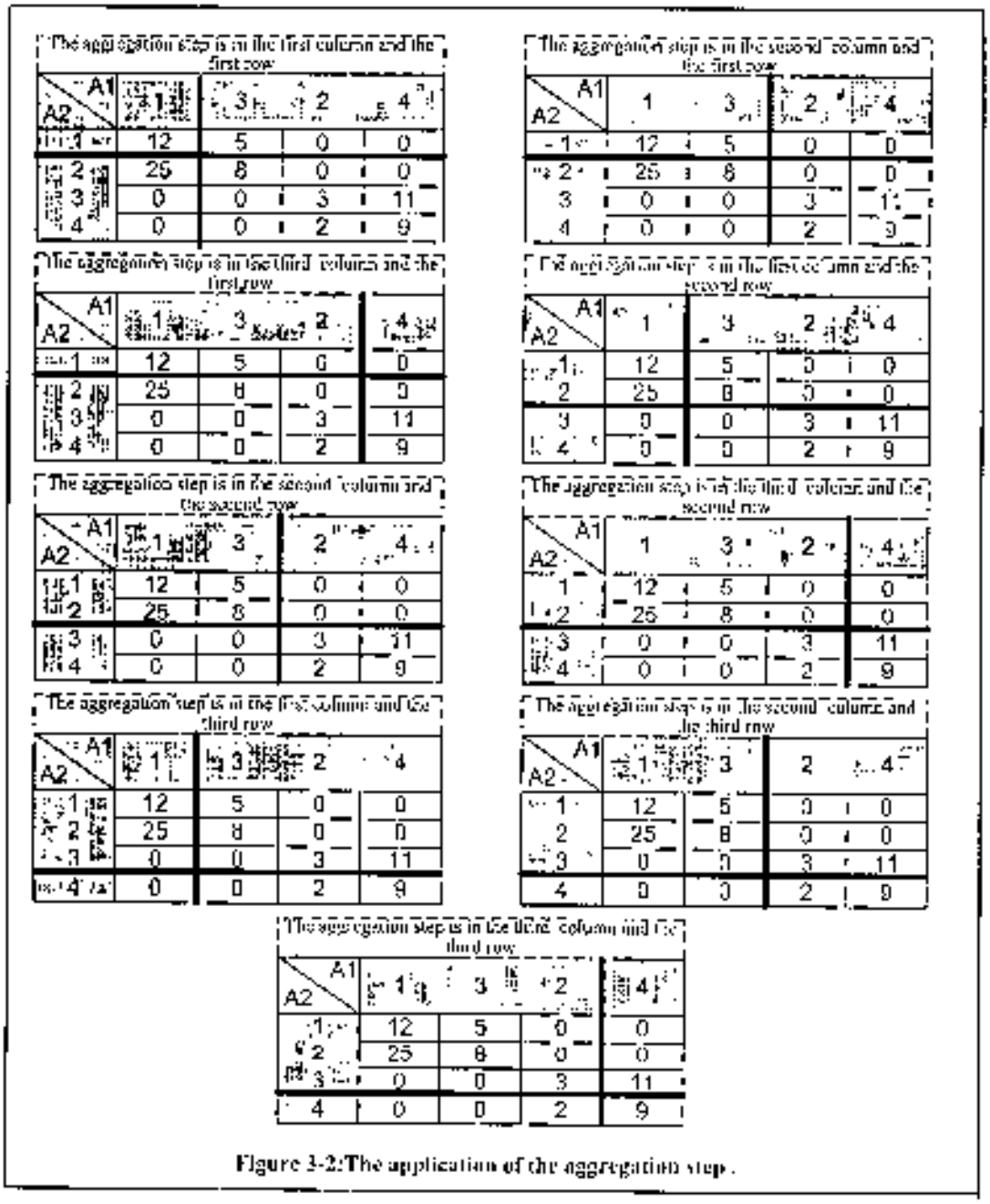


Figure 3-2:The application of the aggregation step .

From figure 3.2, we can see that the best 2 x 2 contingency table is the fifth table, which gives the highest value for λ_r (in fact λ=1.00).

3.2.2 Create hierarchy elements

After the correspondence analysis and aggregation steps have been carried out, the next step is to form or create a hierarchy element (one-level taxonomy) from each found (approximate) equivalence relation. One-level taxonomy is a simple tree structure composed of 3 classes (the root and two children). The root class is labeled with the range of the data (records) in which the (approximate) equivalence holds. Each child is labeled with the descriptors that hold for that child for the given (approximate) equivalence. Figure 3.3 depicts the creation of hierarchy element (one-level taxonomy) step.

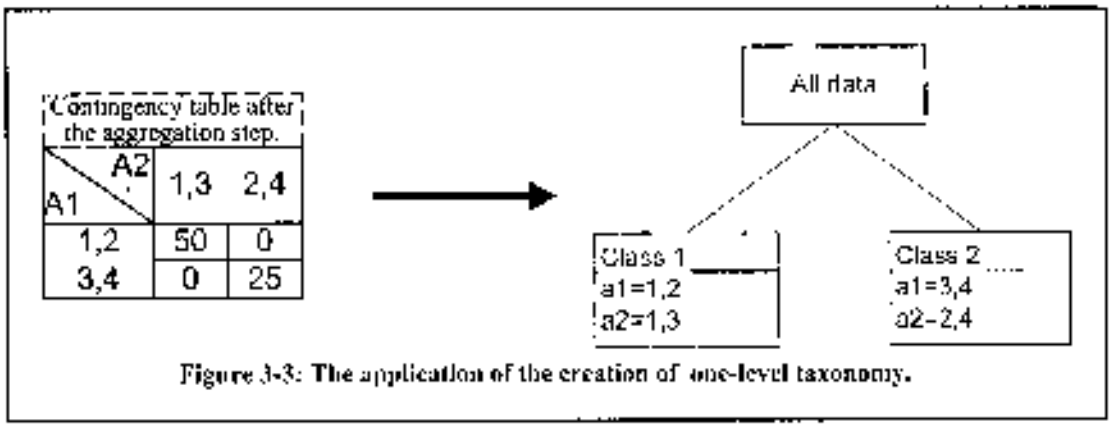


Figure 3-3: The application of the creation of one-level taxonomy.

3.2.3 Merge similar (equivalent) one-level-taxonomies

After the creation of one-level taxonomies, depending on the data and the threshold used, our system tries to find and merge equivalent hierarchy elements. Two hierarchy elements are equivalent if they share the same data range in their roots and have common descriptors in each child. Whenever two or more equivalent hierarchy elements have been found, then they are collapsed into one, where the descriptors are merged. "The equivalent hierarchy elements are joined to increase empirical content". [4]. Figure 3.4 demonstrates the merging process of one- level taxonomies (equivalent hierarchy elements).

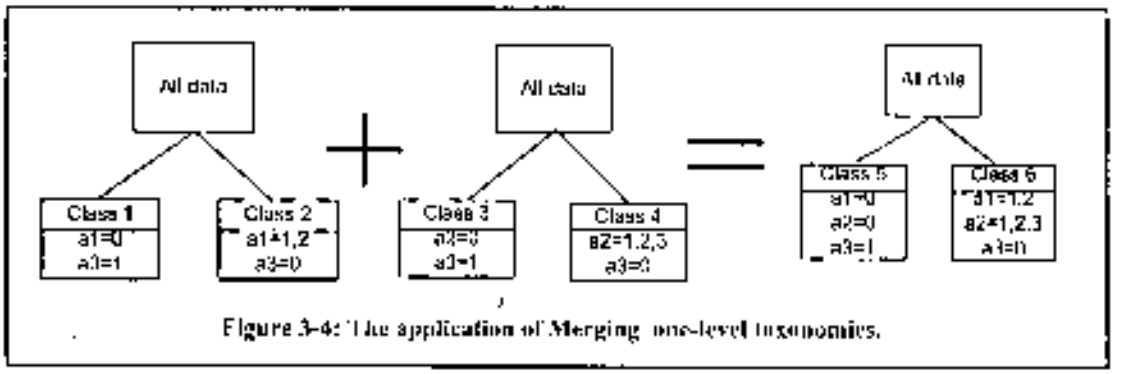


Figure 3-4: The application of Merging one-level taxonomies.

3.2.4 Choose the appropriate partition

After the exhaustive search for the one-level-taxonomies (some of them may have been obtained by merging), the next step is to evaluate all the one-level-taxonomies to choose the most appropriate one. The most appropriate one-level-taxonomy is the one with the highest partition utility function value.

The Partition Utility function (PU), introduced by Gluck and Corter [1985] and has been used in Cobweb system of Fisher [1987], is a tool used to choose the best split of the data to categories[9].

The PU function is based on the Category Utility function (CU). The CU can be based on Gini Index (GI) or Entropy (E). in our case GI is used and the partition utility function is written as:

$$PU = \frac{1}{K}\sum_{k=1}^{K} CU(C_k)$$

$$CU(C_k) = P(C_k)\sum_{i=1}^{I}\sum_{j=1}^{J}\left[P(A_i = V_{ij} \mid C_k)^2 - P(A_i = V_{ij})^2\right]$$

Where:

k=1. . . K and K is the number of categories in a partition.

i=1. . . I and I is the number of attributes in the given data.

j=1. . . J and J is the number of values in attribute A_i.

C_k, k=1. . . K, are the proposed categories whose utility is to be evaluated.

3.2.5 Build Multi-level taxonomy tree

After the appropriate one-level-taxonomy has been chosen, the next step is to place that one-level taxonomy on the multi-level taxonomy tree in a breath-first manner. After placing the one-level-taxonomy on the multi-level- taxonomy tree, the data is splited into two sub-populations (left and right). The condition for the data partitioning is taken from the chosen one-level-taxonomy.

After building the first level of the multi-taxonomy tree, the process starts allover again due to the iterative nature of the algorithm.

3.3 Programming language used

In building our system, we used Visual Basic.net (VB.net), 2005. Professional edition. This version is the highest and most recent versions of VB.net, which is part of Microsoft Visual Studio. As its name implies, VB.net is a visual programming language that facilitates the creation of programs for the Microsoft Windows operating systems environment. The VB.net is a true object-oriented programming language that compiles and runs on the .NET Framework, VB.net is a totally new tool from the ground up [15]. Additional advantages of the use of VB.net in our system are:

1. The syntax of VB.net is easy to understand and use.
2. In addition to the simplicity and ease to create GUI applications, the VB.net also has the flexibility to develop fairly complex applications very easily.
3. Programs written in other languages such as; Microsoft Visual C++ 2005, Microsoft Visual C# 2005 and Microsoft Visual J# 2005 can be easily incorporated.

4. Different types of data storage (i.e. databases and data files) can be dealt with very easy through ready frame work.

3.4 Measures used by our system

In addition to a number of tools such as; contingency tables, correspondence analysis, aggregation and partition utility function, our system uses a set of lambda (λ) as measures of association.

The set Lambda measures (λ), which were proposed by Guttman and Goodman and Kruskal [14] are used in our system to test the association between each two attributes of the given data.

In order to explain the set of lambda measures, let's assume that we have:

1. A table of two cross-classified variables A and B.
2. The variables are of nominal type (i.e. there is no ordering of interest).
3. The A classification precedes the B classification chronologically, causally, or otherwise.

The set of Lambda measures are based on the principle of Proportional Reduction of Error (PRE). The set of lambda measures measures the strength of a relationship by calculating the proportion by which we reduce the errors of predicting the value of one attribute:

$$If \begin{cases} \text{(Case 1) : we have no information on the other attribute.} \\ \qquad \text{(assuming B is statistically independent of A)} \\ \quad \text{or} \\ \text{(Case 2) : we are given the value of the other attribute} \\ \qquad \text{(assuming B is a function of A)} \end{cases}$$

Assume that $p_{.m}$ represents the largest marginal proportion among B classes (the maximum of columns totals) and p_{am} represents the largest proportion in the a^{th} row of the cross-classification table (the maximum of rows totals). Formally:

$$p_{.m} = \underset{b}{MAX}\ p_{.b}$$

$$p_{am} = \underset{b}{MAX}\ p_{ab}$$

In Case 1, it is best to predict B_b for which $p_{.b} = p_{.m}$ that is to predict the B category that has the largest marginal proportion and the probability of error, would be: $1 - p_{.m}$.

In Case 2, it is best to predict B_b for which $p_{ab} = p_{am}$ that is to predict the B category, which has the largest proportion in the A category and then the probability of error is: $1 - \sum_{a} p_{am}$. The principle of PRE relates these two cases by:

$$\text{PRE} = \lambda_b \equiv \frac{\text{probability of error in case(1)} - \text{probability of error in case(2)}}{\text{probability of error in case(1)}}$$

Replacing the probability of errors by their corresponding formula, the measure λ_b is then:

$$\lambda_b = \frac{1 - p_{.m} - (1 - \sum_{a} p_{am})}{1 - p_{.m}}$$

The measure λ_b can be rewritten as:

$$\lambda_b = \frac{\sum_{a} p_{am} - p_{.m}}{1 - p_{.m}}.$$

The replacement of the probability of errors by their proportions, then the measure λ_b is:

$$\lambda_b = \frac{\sum_{a} n_{am} - n_{.m}}{N - n_{.m}}$$

In this sense the λ_b measure represents the relative decrease of error in predicting the B category when the A category is known, as opposed to when the A category is not known.

Analogously to λ_b, the measure λ_a can be defined, which represents the relative decrease of error in predicting the A category when the B category is known, as opposed to when the B category is not known. The measure λ_a is then defined as:

$$\lambda_a = \frac{\sum_b n_{ab} - n_{a.}}{N - n_{a.}}$$

Where $n_{a.} = \max_a n_{a.}$

$$n_{.b} = \max_a n_{ab}$$

The measure λ_a can be rewritten as:

$$\lambda_a = \frac{\sum_b n_{ab} - n_{a.}}{N - n_{a.}}$$

In many situations, we want to predict the A class half of the time and the B class half of the time either by:

1) Given no information, or
2) Given the A_a class when the B_b class is to be predicted or given the B_b class when the A_a class to be predicted.

In this situation we can define the λ measure in the same way as λ_b and λ_a. In the first case, where we have no a priori information, the probability of error in prediction is $1-\frac{1}{2}(\rho_{.m} + \rho_{m.})$ and in the second case, the probability of error in prediction

is $1-\frac{1}{2}\left(\sum_{a} p_{am} - \sum_{A} p_{mA}\right)$. The relative decrease in probability of error is defined by the coefficient:

$$\lambda = \frac{\frac{1}{2}\left[\sum_{a} p_{am} + \sum_{A} p_{mA} - p_{.m} - p_{m.}\right]}{1 - \frac{1}{2}\left(p_{.m} + p_{m.}\right)}$$

The λ measure can be written in terms of counts as:

$$\lambda = \frac{\sum_{a} n_{am} + \sum_{A} n_{mA} - n_{.m} - n_{m.}}{2N - \left(n_{.m} + n_{m.}\right)}$$

The set of lambda measures the same properties due to the fact that they are based on the same principle PRE. According to Goodman and Kruskal [14], the set of lambda measure has the following properties:

i. The value of λ lies between λ_a and λ_b inclusive.

ii. The value of λ is 1 if and only if all population are concentrated in cells no two of which are in the same row or column: there is only one non-zero probability cell in each row.

iii. The values of λ is indeterminate if and only if when the entire population lie in a single cell of the table.

iv. The value λ is 0 in the case of statistical independence.

v. The value λ is unchanged by permutation of rows or columns.

3.5 Design of our system

Our taxonomy formation system consists mainly of three modules. These modules are: Data entry, Search for regularities and Build taxonomy. These modules are executed serially. The data used to test our system is real data taken from the

INTERNET[1]. The used data is all pre-classified and in our system, when we build the taxonomy tree the class labels are discarded (removed). In the analysis and design of our system, we used Unified Modeling Language (UML) in the development of the implementation of our system. According to [8], UML is a standard graphical notation for expressing a system, including conceptual things such as business processes and system functions as well as concrete things such as programming language statements, database schemas, and reusable software components. In the analysis and design of our system, we have used only two main groups of UML diagrams. These two types of diagrams are: structural diagrams and activity diagrams.

3.5.1 Structural diagrams of our system

Structural diagrams are used to display the structural elements that make up a system or functions. These diagrams show a static view of the system model and the relationships between the different structures. In our system, we have used two types of structural diagrams which are: Use Case diagram and Class diagram.

3.5.1.1 Use Case diagram

The use case diagram models the functionality of a system using case and actor's interactions. It describes the functional requirements of the system in the manner that outside actors interact at the system level (boundary) and the response of the system. Our systems' Use Case diagram is depicted in figure 3-5.

[1] Merz, C. J., & Murphy, P.M.1996. UCI Repository of machine learning databases [http://www.ics.uci.edu/~mlearn/MLRepository.html]. Irvine, CA: University of California, Department of Information and Computer Science.

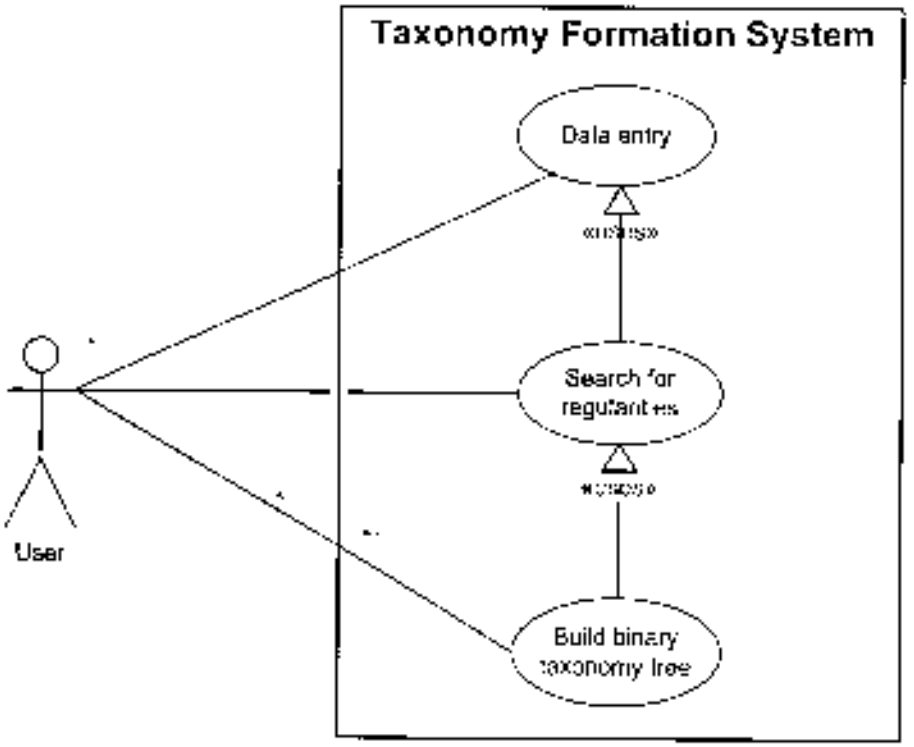


Figure 3-5: Use Case diagram for taxonomy formation system.

The above Use Case identifies the following functions:

Data entry

Used by:

- User.

Inputs:

- Data file name. The file is in text format.

Outputs:

- An Access database file. The database file is a transformation of the text file.

Pre-Conditions:

- The data file name and path must be correct. The data must be noise free and all records must have common delimiter.

Post-Conditions:

- None.

Search for regularities

Use by:

- User.

Inputs:

- An Access database file name. Threshold value.

Outputs:

- None

Pre-Conditions:

- Valid threshold value [0.0 ... 1.0].

Post-Conditions:

- None.

Build binary taxonomy tree

Use by:

- User

Inputs:

- One-level taxonomies.

Outputs:

- Binary taxonomy tree.

Pre-Conditions:

- None.

Post-Conditions:

- None.

3.5.1.2 Class diagram

The purpose of class diagram is to depict the classes within a model. The Class diagram captures the logical structure of the system. It is a static model, describing what exists and what attributes and behavior it has, rather than how something is done. The UML class diagram can depict the contents of a class such as; attributes (member variables), operations (member functions) and relationships with other

classes quite easily. Figure 3.6 depicts the Class diagrams of our system, which consists of the following classes:

Search_for_Regulraity class is one of the most essential classes. This class is responsible of finding the basic building block, for our system, which is the regularity or pattern. . This class has several methods, which are:

- **Create_Con_Tbl:** is a method to create a contingency table for each two attributes in the given database in an exhaustive search.
- **Fre_Count:** is a method to provide the frequency counts for each created contingency table.
- **Reg_Test:** is a method to test each contingency table if it is conductive to be regularity by the use the set of lambda measures of association.

Correspondence_analysis class is a class that rearranges the columns and row of a contingency table so that similar rows/ columns will be adjacent. This class has two methods, which are:

- **Run_Correspondence_analysis: is a method that executes a number of statistical merits that enables the system to find the reordering of columns and rows.**
- **New_columns_order: is a method that reorder the rows and columns of the contingency table**

Aggregation class is a class that tries to find the best 2 × 2 contingency table that (approximate) equivalence relation by the use of λ measure. This class has two methods, which are:

- **Con_Tbl_Test:** is a method that tests each produced 2 × 2 contingency if it approximates an equivalence relation in an exhaustive fashion.

- **Run_ aggregation:** is a method that holds the best 2 × 2 contingency that approximates an equivalence relation.

Merge class is a class that merges two equivalent one-level taxonomies into one. This class has only one method, which is:

- **Merge_test:** is a method that tests if the given two one-level taxonomies can be merged

Build taxonomy tree class is a class that chooses the best one-level taxonomy (merged or not) to be placed next on the binary multi-level taxonomy. This class has only one method, which is:

- **Part_Util_Fn:** is a method to test all the created one-level taxonomies and to choose the best of them to split the data into two parts.

Load_Tree class is a class for storing and displaying binary taxonomy tree. This class has two methods, which are:

- **Save_TaxonomyTree:** is a method to store binary taxonomy tree after its construction.
- **Load_TaxonomyTree:** is a method to load saved binary taxonomy tree to be displayed.

Main class is a class that controls the over all operations of the system. This class has several methods, which are:

- **Read_data:** is a method to data read file.
- **Write_to_DataSet:** is a method to load data from data file to displaying container.
- **CopytableTOarray:** is a method that copies a database file to a matrix.
- **Count_Rows_Tbl:** is a method to count the number rows in a table.
- **Count_colum_Tble:** is a method to count the number columns in a table.

- **Get_Domain:** is a method to get the domain of each attribute.

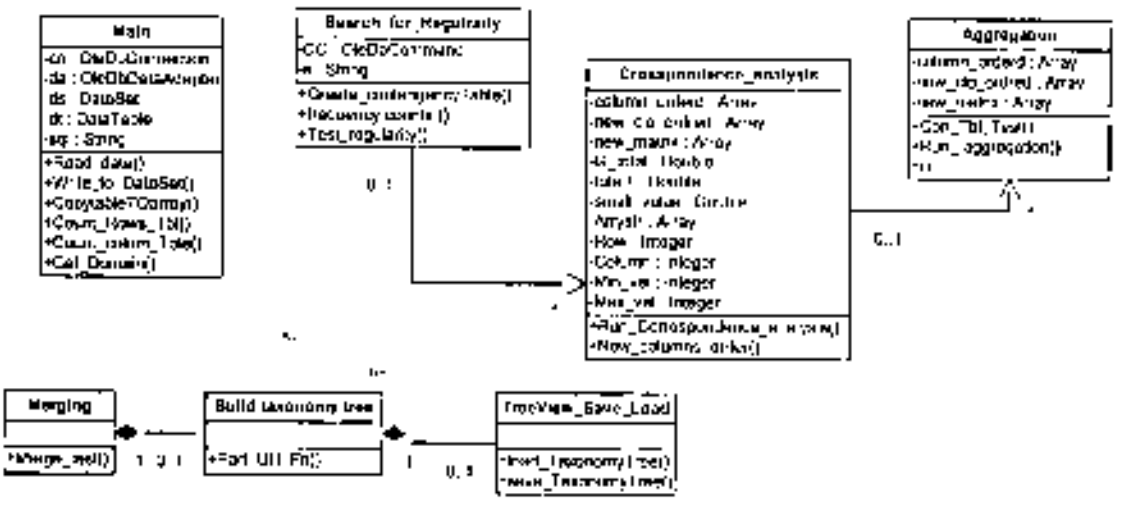
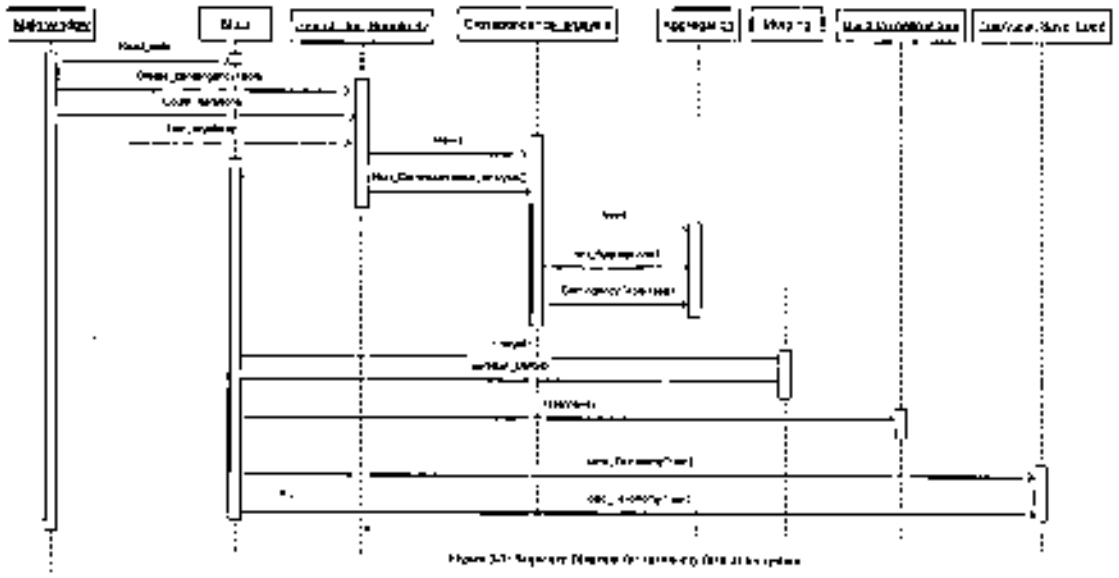


Figure 3-6: Class diagram for taxonomy formation system.

3.5.2 Behavioral diagrams

Behavioral diagrams are used for describing behavioral perspectives of a given system as well as messages between objects (process workflows). From the group of behavioral diagrams, in our system we have used only the sequence diagram.

A **sequence diagram** details the overall flow of control in the sense of how operations are performed and what messages are sent and when sent. Sequence diagrams are organized according to time wise fashion. Objects involved in a system are listed from top to bottom and from left to right according to their use. Figure 3.7depicts the sequence diagram of our system.



Chapter Four

Experiments and results

The formation of taxonomies is very useful in so many domains such as Botany, Zoology, Astronomy and many other fields. Taxonomy is a hierarchical system that consists of classes. The classes are usually mutually exclusive with respect to the presence or absence of some features. The subclasses of the taxonomy possess discriminating features of their super classes

The purpose of this chapter is to report on a number of experiments that have been conducted by the use of our system, "Taxonomy formation".

4.1 data sets

The data sets that have been used to test our taxonomy formation system are taken from public domains on the INTERNET [2]. The data sets that we have used to conduct our experiments are previously been classified. We have deleted the classification (Class) attribute from the data before used in our system. All the data sets that we have used are noise free (no missing data). Our system is limited to nominal data values and for continuous data we have used a system that is available in the public domain of the University of Liverpool [3] to convert continuous data values to discrete data values.

4.1.1 Small soybean database experiment

This experiment deals with the well-known Small soybean data. This data set is concerned with the diseases of the soybean plant. The description of this data set is as follows:

Small soybean data description	
Number of instances	47
Number of attributes	35
Class names (diseases names)	D1: Diaporthe Stem Canker D2: Charcoal Rot D3: Rhizoctonia Root Rot D4: Phytuphthora Rot
Class distribution ...	D1: 10 instances. D2: 10 instances. D3: 10 instances. D4: 17 instances.

Table 4-1: Small soybean data set information.

Our taxonomy formation has produced the following taxonomy for the small soybean data set in figure 4-1:

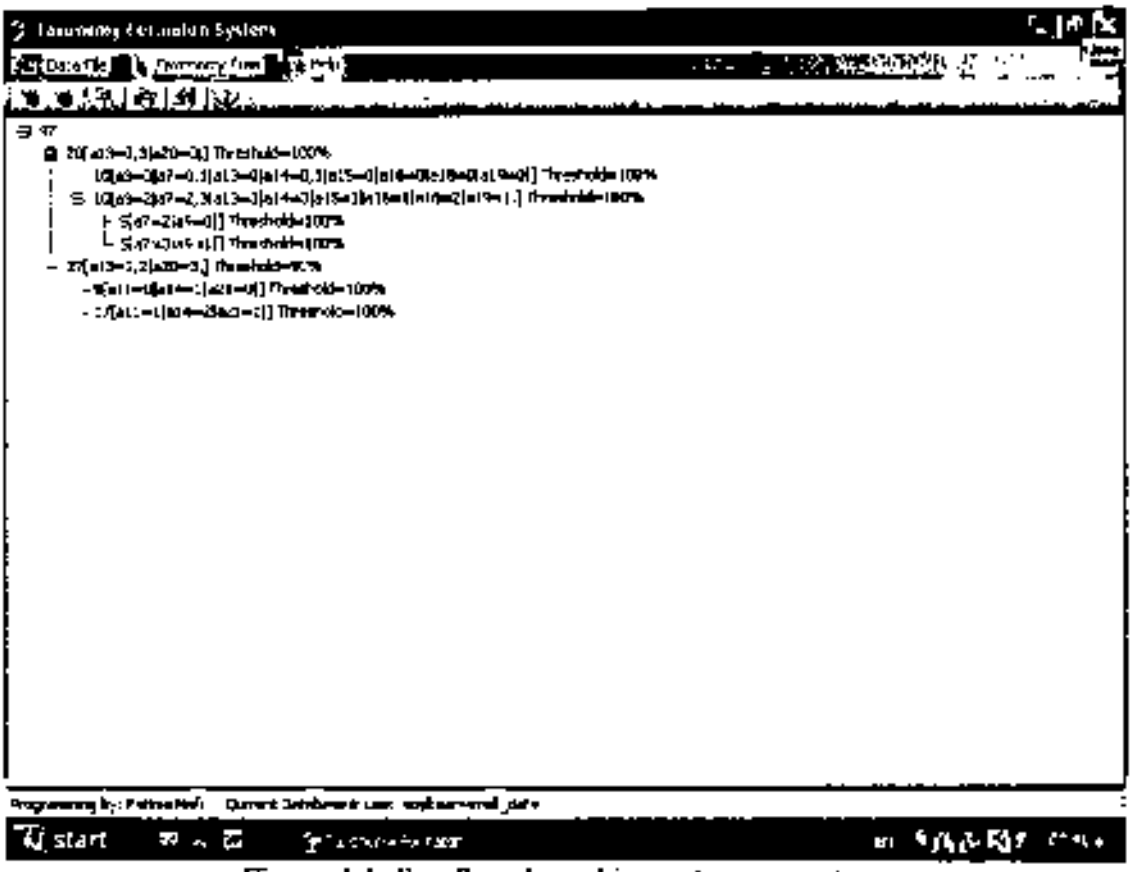


Figure 4-1: Small soybean binary taxonomy tree.

In this experiment, our system has considered threshold of 0.90 for the set of lambda measures when ever threshold of 1.00 is not possible. For purpose of clarity, we have redrawn the above binary taxonomy tree in figure 4-2

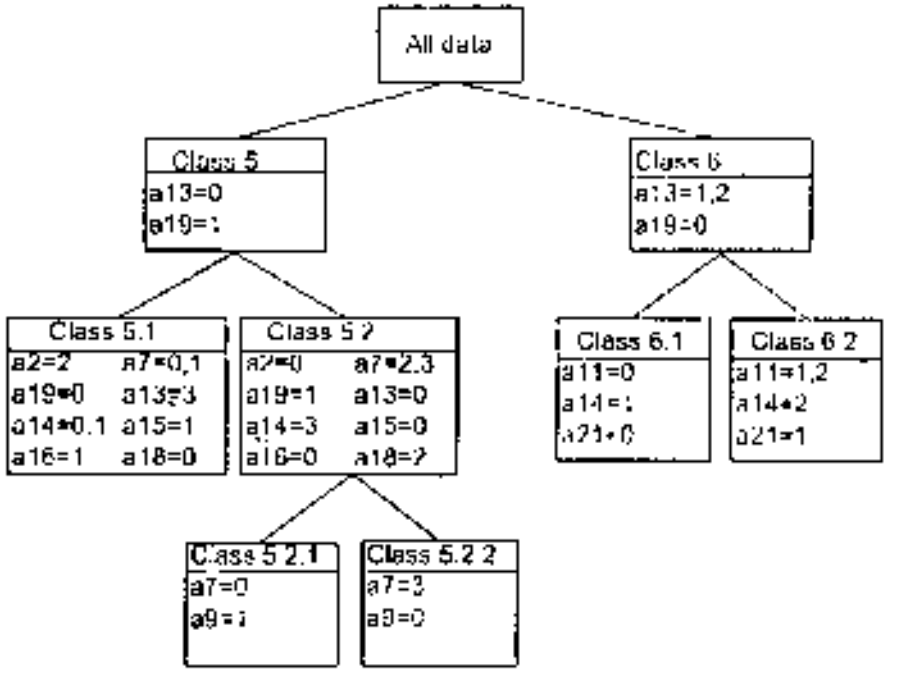


Figure 4-2: Redrawing of the small soybean binary taxotomy tree.

4.1.2 Large soybean database experiment

The second experiment deals with the large soybean database experiment. It is also concerned with the diseases of the soybean plant. This data has some missing values. After manual preprocessing, the description of the data is depicted in table 4-2.

Data	Description before processing	Description after processing
Number of instances	307	266
Number of attributes	35	35
Number of instance with missing values	307	none
Class names (diseases names)	D1: diaporthe-stem-canker D2:charcoal-rot D3:rhizoctonia-root-rot D4:phytophthora-rot D5:brown-stem-rot	The same classes

Data	Description before processing	Description after processing
	D6:powdery-mildew D7:downy-mildew D8:brown-spot D9:bacterial-blight D10:bacterial-pustule D11:purple-seed-stain D12:anthracnose D13:phyllosticta-leaf-spot D14:alternarialeaf-spot D15:frog-eye-leaf-spot D16:diaporthe-pod-&-stem-blight D17. cyst-nematode D18: 2-4-d-injury D19: herbicide-injury	
Class distribution	D1: 10 instances. D2: 10 instances. D3: 10 instances. D4: 40 instances. D5: 20 instances. D6: 10 instances. D7: 10 instances. D8: 40 instances. D9: 10 instances. D10: 10 instances. D11: 10 instances. D12: 20 instances. D13: 10 instances. D14: 40 instances. D15: 40 instances. D16: 6 instances. D17: 6 instances. D18: 1 instance.	D1: 10 instances. D2: 10 instances. D3: 10 instances. D4: 16 instances. D5: 20 instances. D6: 10 instances. D7: 10 instances. D8: 40 instances. D9: 10 instances. D10: 10 instances. D11: 10 instances. D12: 20 instances. D13: 10 instances D14: 16 instances. D15: 40 instances. D16: 0. D17: 0. D18: 0.

Data	Description before processing	Description after processing
	D19: 4 instances.	D19: 0.

Table 4-2: Large soybean data set before and after preprocessing.

After preprocessing (removal of missing data), we have ended up with that has 266 instances and 15 classes [27, 31 and 25]

Our taxonomy formation system has produced the binary taxonomy tree that is depicted in figure 4-3

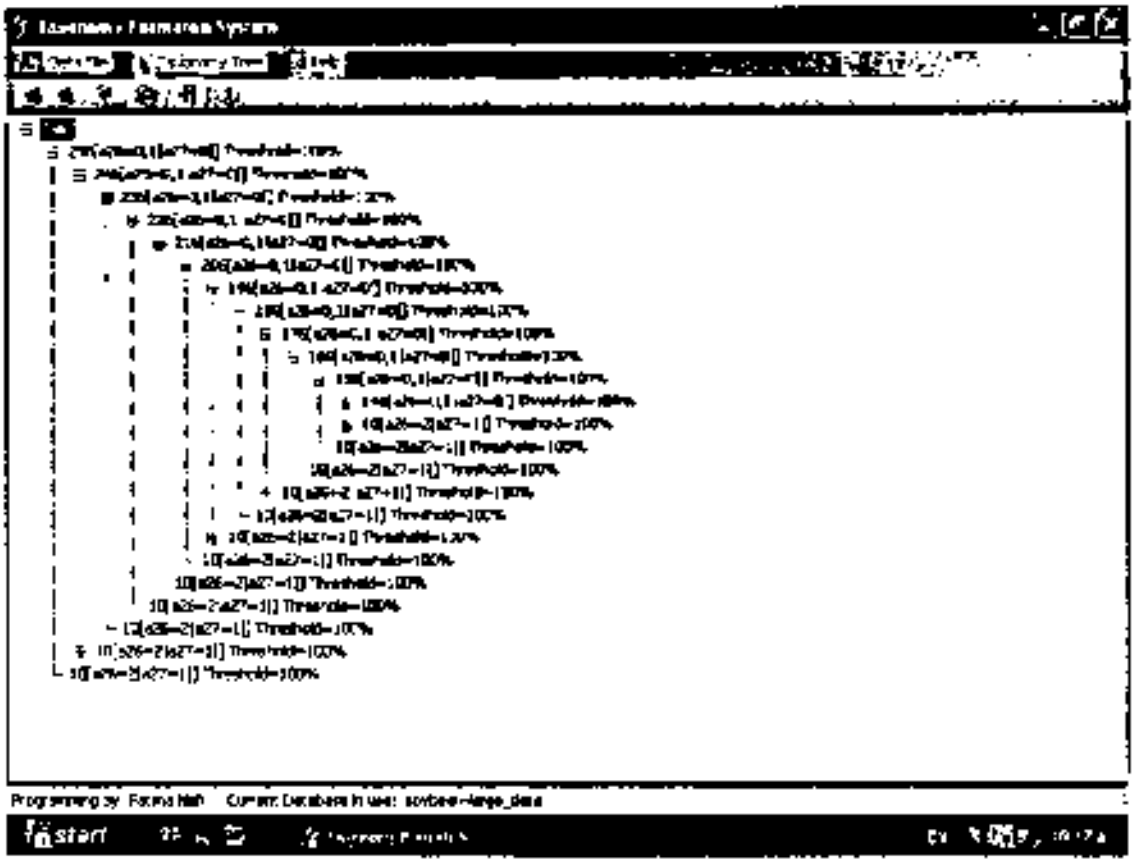


Figure 4-3: Large soybean binary taxonomy tree.

For simplicity purposes of the binary taxonomy tree, our system has considered only threshold of 1.0 for the set of lambda measures.

4.1.3 Iris database experiments

The third experiment deals with the measurements of an Iris plant. This data consists of four numeric attributes Sepal Length in cm, Sepal Width in cm, Petal Length in cm, and Petal Width in cm, the description of the data is depicted in table 4-3.

Iris data description	
Number of instances	150
Number of attributes	4
Class names	Iris Setosa Iris Versicolour Iris Virginica.
Class distribution	Iris Setosa: 50 instances. Iris Versicolour: 50 instances. Iris Virginica: 50 instances.

Table 4-3: Iris data set information.

Due to the fact that our system works only with discrete data, we have discretizied the Iris data by the use of a system that is available from Liverpool University [3].

Figure 4-4 depicts the binary taxonomy tree that our system has produced for the Iris data.

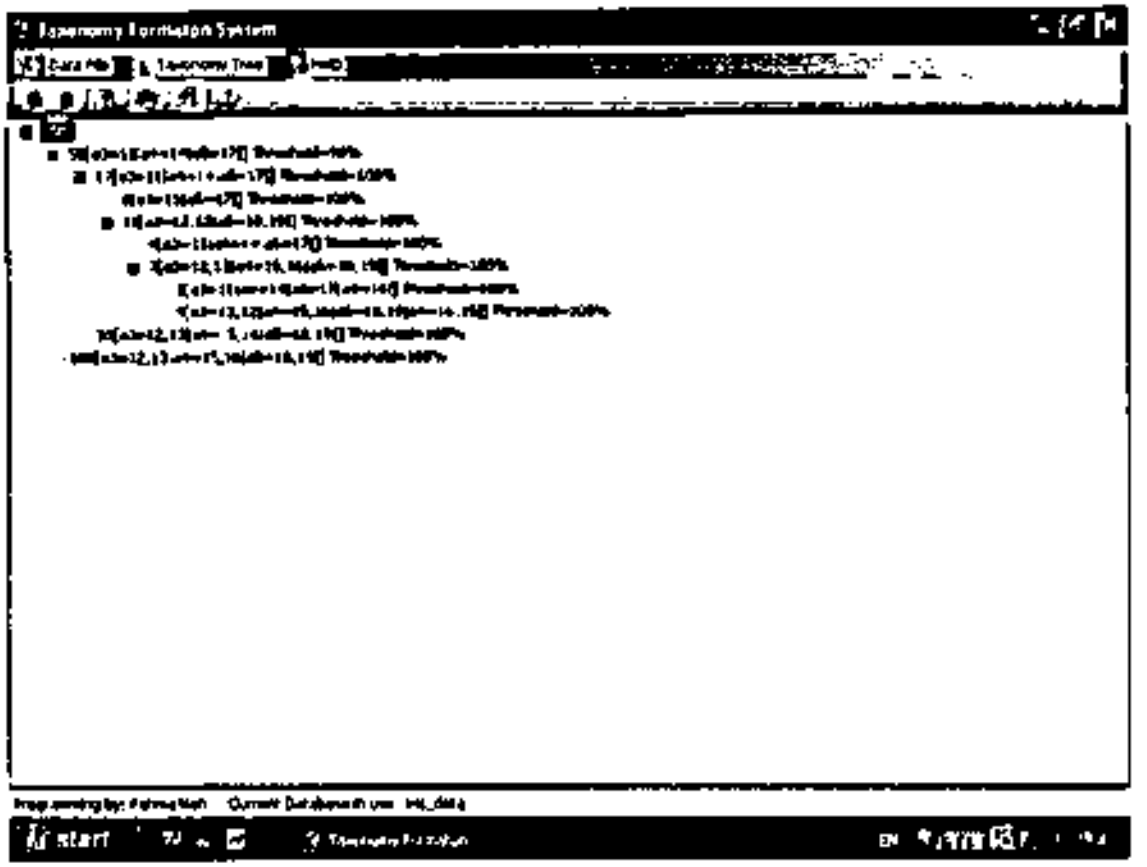


Figure 4-4: Iris binary taxonomy tree.

For the Iris data, our system has considered threshold of 0.90 when ever threshold of 1.00 is not possible. Figure 4-5 represents a redrawn of the taxonomy tree in figure 4-4

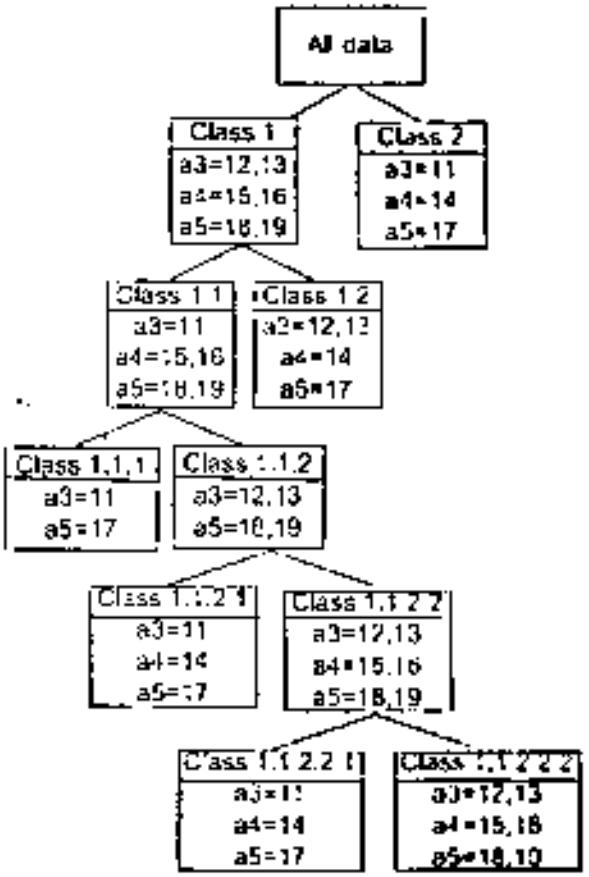


Figure 4-5: Redrawing of the iris binary taxonomy tree.

4.1.4 Glass database experiments

The fourth experiment deals with the Glass identification database. In this experiment, all attributes are of continuous type. This data is concerned with the identification of different types of glasses that can be used in criminological investigation at scenes of the crimes. A description of this data is given in table 4-4.

Glass identification data description	
Number of instances	214
Number of attributes	10
Class names	float processed building windows float processed vehicle windows

Glass identification data description	
	non-float processed building windows non-float processed vehicle windows containers tableware headlamps
Class distribution	float processed building windows: 70 float processed vehicle windows: 17 non-float processed building windows: 76 non-float processed vehicle windows: 0 Containers: 13 Tableware: 9 Headlamps: 29

Table 4-4: Glass data set information.

Liverpool University [3] system for discretization is used to discretize this experiments' data.

Our taxonomy formation system has produced the binary taxonomy tree that is depicted in figure 4-6 for the glass data.

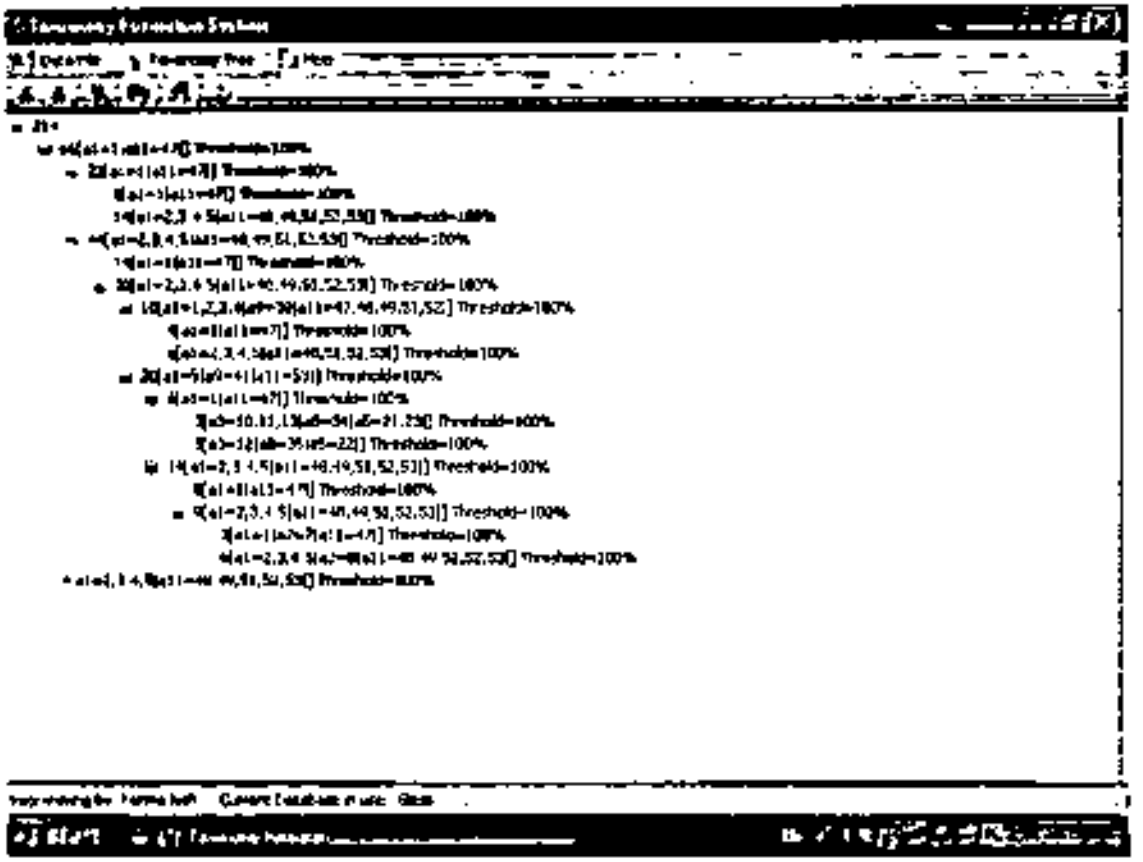


Figure 4-6: Glass binary taxonomy tree.

For purpose of clarity, we have redrawn the above binary taxonomy tree in figure 4-7.

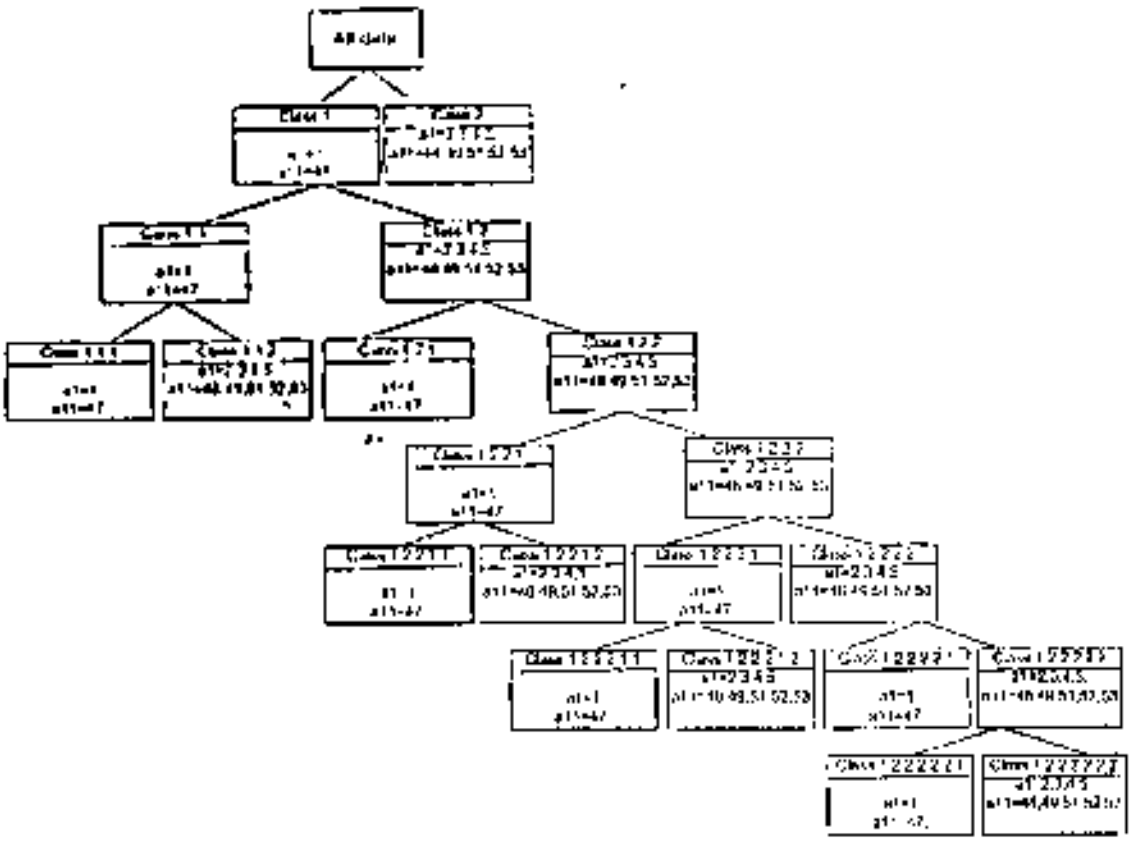


Figure 4-7: Redrawing of the glass binary taxonomy tree.

4.1.5 Zoo database experiments

This experiment deals with the Zoo data. The data characterize number of animals within a Zoo. The data set consists of 15 Boolean variables and one numeric variable for the number of legs for the animal. Because there are too many animal names in each of the classes, we resort to numbering the class (Class1, Class2… Class7). Table 4-5 depicts Zoo data description.

Zoo data description	
Number of instances	101
Number of attributes	18
Class names	Too many
Class distribution	Class1:40. Class 2: 20.

	Class 3: 5. Class 4. 13. Class 5: 4 Class 6: 8. Class7: 10

Table 4-5; Zoo data set information.

Figure 4-8: Zoo binary taxonomy tree.

Figure 4-9, depicts redrawing of the Zoo binary taxonomy tree.

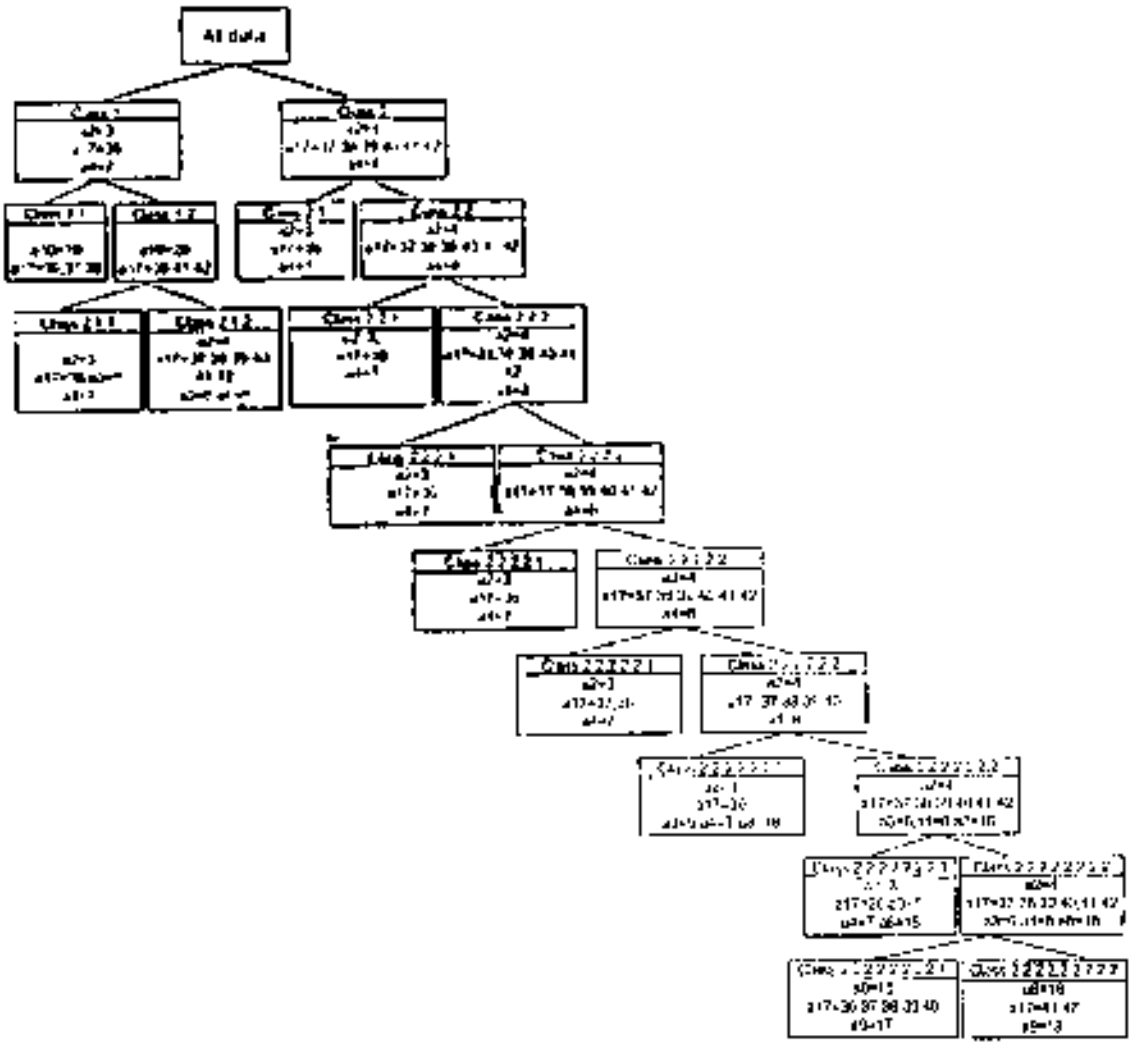


Figure 4-9: Redrawing of the zoo binary taxonomy tree.

Chapter Five

Conclusion and further research

As it is defined in [25], Taxonomy is a hierarchal system of selected subsets of a domain, typically arranged in a tree, which is exhaustive and disjoint. In the field of data mining, taxonomies are a restricted or (special) type of knowledge, which can be transformed to other forms such as rules.

The purpose of this thesis is to implement a computer system for KDD that specializes in binary taxonomy formation. The algorithm of this system has been developed in [4].

The following main algorithm governs the process of taxonomy formation:

Main Algorithm: Build taxonomy tree

> Iterate the following steps until there are no more (approximate) equivalence relations.
>
> - Find (Approximate) Equivalence Relation (Pattern).
> - Create one-level-taxonomy.
> - Merge similar (equivalent) one-level-taxonomies.
> - Choose the appropriate partition.
> - Build Multi-level taxonomy tree.

The algorithm does not depend on any similarity/dissimilarities measures but it relies on approximated equivalence relations, by the use of a set of lambda (λ) measures as measure of association, and partition utility functions in the process of forming a taxonomy tree.

The algorithms' taxonomy formation process is a type of unsupervised learning, i.e., no a priori model of the data is assumed (neither the number of classes is pre-specified nor any assumptions about data distribution are made).

We have developed the above mentioned algorithm in a system that specializes in binary taxonomy formation by:

1. The use of UML, for the analysis and design phases.
2. The use of VB.NET programming language for the implementation phase.

After the development of our system, we have tested it by a number of well known data sets. In the testing phase of the system, we used data sets of different data types and sizes.

The results obtained from the different experiments that we have conducted using our system are the same as expected. Our results are the same results of [1, 4, 9, 27 and 32].

Depending on the obtained results from our system, the author would like to make a number of observations:

1. **The approval of the validity of use of approximate equivalence relations as a basis for taxonomy formation.**
2. **The validity of use of the set of λ measures as measures of association.**
3. Our system can be used as a preprocessing system for other classification algorithms and data reduction

The author would like make a number of suggestions for further research:

1. Incorporation of subsystem to deal with missing data values and noisy data.
2. In incorporation of subsystem to deal with sparse data.

3. Test our system with larger data.

4. The extension of our system to deal with non binary taxonomy trees.

5. The study of the possibility to apply this system in Libya in the field of Medicine (i.e. Cancer diagnosis, classification of hart diseases and of heart and diabetes), in the field of Social and Economical studies, in field of Oil (i.e. Geophysics) and many other fields.

References

1. **Biswas, G., Weinberg J., Yang Q. and Koller G. (1991). Conceptual Clustering and Exploratory Data Analysis, Proceedings of the 8th International Workshop on Machine Learning, Evanston, IL, pp. 591–595.**

2. **Blake, C.L. and Merz, C.J. (1998). UCI Repository of machine learning databases http://www.ics.uci.edu/~mlearn/MLRepository.html. Irvine, CA: University of California, Department of Information and Computer Science.**

3. Coenen, F. (2003). LUCS-KDD DN Software, http://www.csc.liv.ac.uk/~frans/KDD/Software/LUCS_KDD_DN/, Department of Computer Science, The University of Liverpool, UK.

4. El-Mouadib, F., PhD. (2000). Thesis in title of Taxonomy Formation By Approximate Equivalence Relations, Polish Academy of Science, Institute of Computer Science, Warsaw, Poland.

5. Fayyad, U. (1996). Data Mining and Knowledge Discovery: Making Sense Out of Data. IEEE EXPERT, Vol.11, No.5, pp. 20-25.

6. Fayyad, U., Piatetsky-Shapiro G., and Smyth, P. (1995). From Data Mining to Knowledge Discovery: An Overview", In Fayyad U., Piatetsky-Shapiro G., Smyth P, and Uthurusamy R. (eds.), Advances in Knowledge Discovery and Data Mining, AAAI Press, pp. 1-34.

7. Fayyad, U.M., Piatetsky-Shapiro, G., Smyth, P. (1996). The KDD Process for Extracting Useful Knowledge from Volumes of Data. Communications of the ACM, Vol.39, No.11, pp. 27-34.

8. **Filev, A., Loton, T., McNeish, K., Schoellmann, B., Slater, J. and Wu, C. (2003). Professional UML with Visual Studio .NET: Unmasking Visio for Enterprise Architects, Wrox Press Ltd.**

9. **Fisher, D. (1987). Knowledge Acquisition Via Incremental Concept Clustering. Machine Learning, 2, pp. 139–172.**

10. **Fisher, D. (1996). Iterative Optimization and Simplification of Hierarchical Clusterings, Journal of Artificial Intelligence Research, 4, pp. 147–179.**

11. Fisher, D. and Hapanyengwi G. (1992). Database Management and Analysis Tools of Machine Induction, Journal of Intelligent Information Systems, 2, pp. 5–38.

12. Fisher, D. and Langley P. (1986). Conceptual Clustering and Its Relation to Numerical Taxonomy, In Artificial Intelligence and Statistics, Gale W. A. (ed.), AT&T Bell Laboratories, Addison-Wesley Pub, pp. 77–117.

13. Frawley W. J., Piatetsky-Shapiro G. and Matheus C. J. (1992). Knowledge Discovery in Databases: An Overview", AI Magazine, 13(3), pp. 57–70.

14. Goodman, L. and Kruskal, W. (1979). Measure of association for Cross Classification, Springer Series in statistics, Springer-Verlag, 1979, New York, USA. Reprinted from the Journal of The American Statistical Association, 1954, 49, pp. 732–764.

15. Halvorson, M. (2005). Microsoft Visual Basic 2005 Step by Step, Microsoft Press.

16. Kantardzic, M. (2003). DATA MINING: Concepts, Models, Methods, and Algorithms. John Wiley & Sons.

17. Klosgen, W. and Zytkow, J. (1995). Knowledge Discovery in Databases Terminology. In Fayyad U. M., Piatetsky-Shapiro G., Smyth P. and Uthurusamy R. (eds.), Advances in Knowledge Discovery and Data Mining, AAAI Press, pp. 574-592.

18. Larose, D. (2005).Discovering Knowledge in Data an Introduction to Data Mining. John Wiley & Sons, Inc.

19. Mannila, H. (1997). Methods and Problems in Data Mining. In the proceedings of International Conference on Database Theory, Afrati, F. - Kolaitis, P., Delphi. Springer-Verlag.

20. Mardia, K. V., Kent, J. T. and Bibby, J. M., (1979). Multivariate Analysis, Academic Press, New York.

21. Quinlan, J. R. (1983). The effect of noise on concept learning. In michalski et al Machine learning: An artificial intelligence approach. Palo Alto: Tioga Publishing Company., pp. 149-166.

22. Quinlan, J. R. (1989). Incremental Induction of Decision Trees. Readings in machine learning, Original paper was published in Machine Learning, 4, 1986. Boston, Kluwer Academic Publishers, 1986, pp. 161 – 186.

23. Quinlan, J. R. (1990). Induction of Decision Trees, Readings in machine learning. Shavlik J. W. and Dietterch T. G. (eds.), Morgan Kufmann, pp 57–69. Original paper was published in Machine Learning, 1, 1986, Boston. Kluwer Academic Publishers, 1986, pp. 18 – 106.

24. Quinlan, J. R. (1992). C4.5: programs for machine learning, Readings in machine learning, Shavlik J. W. and Dietterch T. G. (eds.), Morgan Kufmann, pp. 57-69. Original paper was published in Machine Learning, 1, 1986, Boston, Kluwer Academic Publishers, 1986, pp. 37

25. Troxel, M., Swann, K., Zembowicz, R. and Żytkow, J. (1994). "Concept Hierarchies: a Restricted Form of Knowledge Derived From Regularities", In

Proceeding of the seventh International Symposium on Methodologies for Intelligent Systems held. Raś Z. and Zemankova M. (eds.), pp. 437–447.

26. Ye, N. (2003). The Handbook of Data Mining, Lawrence Erlbaum Associates, Inc., Mahwah, New Jersey.

27. Zembowicz, R. and Żytkow, J. (1995). From Contingency Tables To Various Forms of Knowledge in Databases, In Fayyad U. M., Piatetsky-Shapiro G., Smyth P. and Uthurusamy R. (eds.), Advances in Knowledge Discovery and Data Mining, AAAI Press, pp. 329–349.

28. Żytkow, J., and Baker, J. (1991). Interactive Mining of Regularities in Databases, in Piatetsky-Shapiro G. and Frawley W. J. (eds.), Knowledge Discovery in Databases, AAAI/MIT Press, pp. 31–53.

29. Żytkow, J. and Zembowicz, R. (1992). Discovery of Regularities in database, In Proceedings of the ML-92 workshop on Machine Discovery (MD-92). Jan M. Żytkow (ed.), Aberdeen, Scotland, UK, pp. 18–27.

30. Żytkow, J. and Zembowicz, R. (1993). Database Exploration in Search of Regularities, Journal of Intelligent Information Systems, 2, pp. 39–81.

31. Żytkow, J. and Zembowicz, R. (1997). Contingency Tables as the Foundation for Concepts, Concept Hierarchies and Rules: The 49er System Approach, Fundamenta Informaticae, IOS Press, 30, pp. 383–399.

ملخص الرسالة

تتسارع خطا التقدم في مجال تكنولوجيا الحاسوب بشكل يصعب معه ملاحقة التطور، ولاسيما في مجال توليد وتجميع البيانات، بحيث أصبح الحاسوب عنصرا هاما ومؤثرا في جوانب عديدة من المجتمع، لذلك فإن التقنيات التقليدية لمعالجة البيانات لم تعد قادرة على التعامل مع هذا الكم الهائل من البيانات لكي تساعد في اتخاذ القرار.

لذلك أصبح من الضروري وجود أدوات جديدة تتعامل مع هذه البيانات لغرض المساعدة في اتخاذ القرار، ومن هذه الأدوات أداة جديدة تعرف باسم البحث عن المعرفة في قواعد البيانات، ويعني ذلك عملية الكشف والعثور على معرفة ذات فائدة من خلال استعمال مجموعة من الأدوات المعقدة، مثل المعايير الإحصائية، والنوات والذكاء الاصطناعي، والرسومات البيانية. وتتكون هذه الاداة من سلسلة من الخطوات أهمها استخراج المعرفة، وهي منهجية جديدة تجمع بين نتائج العديد من الأبحاث، وهناك عدة مهام مختلفة من أجل استخراج المعرفه، كما ان اختيار المهمة المناسبة يعتمد على طبيعة البيانات وعلى حجمها. ومن أهم هذه المهام التصنيف وهو إيجاد المعرفه في شكل شجرة هرميه مثل: تكوين التصنيفات

تكوين التصنيفات: يمكن تعريفها بأنها نظام هيكلي مختار لمجموعة من قيم الصفات مرتبة في هيكل شجرة، بحيث تكون مفصلة وشاملة.

خوارزمية تكوين التصنيفات وهي تشمل عددا من الخطوات التكرارية، كلّ منها يُنجزُ مهمه معينة، وقد اقتصر موضوع هذه الرسالة على تكوين التصنيفات كنوع محدد من المعرفة في قواعد البيانات. بحيث تم تصميم و تنفيذ نظام برمجي كامل لهذه الخوارزمية، وأطلق على هذا النظام تكوين التصنيفات، كما تم اختبار النظام بمجموعة من البيانات الحقيقية، وبمقارنة النتائج المتحصل عليها مع نتائج بحوث سابقة تحصلنا على نتائج عالية الجوده من نظامنا.

وتحتوي هذه الرساله على الفصول التاليه:

الفصل الأول: عبارة عن مقدمة للرسالة تتضمن الدافع وراء هذا العمل، وأهم الاهداف.

الفصل الثاني: يعرض هذا الفصل ملخص لبعض من أنظمة اكتشاف المعرفة في قواعد للبيانات حيث أن هذه الأنظمة طرق مختلفه في إنجاز عملها.

الفصل الثالث: يعرض اسلوب التحليل والتصميم المستخدم في وصف العناصر الرئيسية لنظام تكوين التصنيفات وذلك بإستخدام لغة النمذجة الموحدة(Unified Modeling Language) كما يعرض لغة البرمجة المستخدمة (الفيجول بيسك دوت نت2005).

الفصل الرابع: يستعرض نتائج التجارب التي تم تنفيذها بإستخدام نظامنا ومقارنة النتائج المتحصل عليها بنتائج بحوث علمية سابقة.

الفصل الخامس: هو خاتمة الرسالة، ويعطي ملخص للنتائج التي تم التوصل إليها ويعرض بعض التوصيات البحثيه المستقبليه وإمكانية الإستفاده من هذا العمل في الجماهيريه.

كليـــــة العلـــــوم

قسم الحاسوب

عنـــوان البحـــث

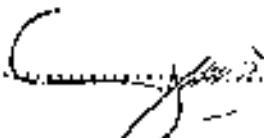
((تكوين التصنيفات كنوع محدد من المعرفة في قواعد البيانات))

مقدمـــة مـــن الطـــالبة

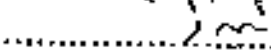
فاطمة بلعيد بن نافع

** لجنـــة المناقشـــة :

الدكتور / فرج عبدالقادر المؤدب
(مشـــرف الرســـالـــة)



الدكتور / إدريس ساسي الثقي
(ممتحـــن داخلـــي)



الدكتور / عبد الحميد عبد الكافي
(ممتحـــن خارجـــي)

يعتمـــد ..

د. محمد علي سليم المرجاني

أمين اللجنة الشعبية لكلية العلوم

جامعة التحدي-كلية العلوم

تكوين التصنيفات كنوع محدد من المعرفة في قواعد البيانات

رسالة مقدمة لقسم الحاسوب

كمتطلب جزئي للحصول على درجة الماجستير في علوم الحاسوب

مقدمة من الطالبة:

فاطمة بلعيد بن نافع العماري

إشراف: د. فرج المؤدب

العام الجامعي 2005/ 2006