Visualizing Conceptual Lattice of Textual Data Using ToscanaJ

Thesis submitted in partial fulfillment of the requirements for the award of degree of

Master of Engineering
In
Computer Science & Engineering

By:
SHIKHA
(80732019)

Under the supervision of:
Ms. Shalini Batra
Sr. Lecturer, CSED

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT
THAPAR UNIVERSITY
PATIALA – 147004

MAY 2009

Certificate

I hereby certify that the work which is being presented in the thesis entitled, **"Visualizing Conceptual Lattice of Textual Data Using ToscanaJ"**, in partial fulfillment of the requirements for the award of degree of Master of Engineering in computer science and engineering in Computer Science and Engineering Department of Thapar University, Patiala is an authentic record of my own work carried out under the supervision of **Ms. Shalini Batra** and refers other researcher's works which are duly listed in the reference section.

The matter presented in this thesis has not been submitted for the award of any other degree of this or any other university.

Shikha
(SHIKHA)

This is to certify that the above statement made by the candidate is correct and true to the best of my knowledge.

(Ms. Shalini Batra)
Supervisor
Computer Science and Engineering Department
Thapar University
Patiala

Countersigned by

Dr. Mrs. Seema Bawa
Professor & Head
Computer Science & Engineering Department
Thapar University
Patiala

Dr. R. K. Sharma 7/8/09
Dean, Academic Affairs
Thapar University,
Patiala

Acknowledgement

It is a great pleasure for me to acknowledge the guidance, assistance and help I have received from my supervisor– Ms. Shalini Batra. I am thankful for her continual support, encouragement, and invaluable suggestions throughout my degree. The good paragraphs in this work are the result of her influence; the chaotic parts belong to me alone. She not only provided me help whenever needed but also provided me with resources required to complete this work. She proved to be an ideal supervisor during the thesis.

I am very thankful to my respected parents for everything they have done for me throughout my life. It is because of their blessings that I came to study at this place. I was nothing without them, I am nothing without them and I will be nothing without them. Although thanks is a very small word in return for their contributions to my life, but still a very heartily thanks for their unconditional love and support throughout my life. I am also thankful to Computer Science and Engineering Department, for giving me an opportunity to do this work.

I would like to say many thanks for everyday support to my dearest Classmates. I want to express my appreciation to every person who contributed with either inspirational or actual work to this thesis.

(SHIKHA)

ABSTRACT

Clustering is a technique which divides the data into objects of similar type. The term *cluster analysis* encompasses a number of different algorithms and methods for grouping objects of similar kind into respective categories. Cluster analysis is an exploratory data analysis tool which aims at sorting different objects into groups in a way that the degree of association between two objects is maximal if they belong to the same group and minimal otherwise. Clustering can be done on the basis of data or concepts. Concepts are necessary for representing human knowledge therefore it is beneficial to use Conceptual Clustering. Conceptual clustering is based on the concept that is it clusters the data on the basis of concept and formal context. It is distinguished from ordinary data clustering by generating a concept description. A system that store, process and present information using concept-oriented representation is called Conceptual Information system. Formal concept analysis uses the CIS for conceptual clustering. Formal Concept Analysis (FCA) is a technique that explains how the document clusters are clustered conceptually. FCA was introduced for modeling the concept in terms of lattice theory. It is a method for data analysis, knowledge representation and information management, representing the data in form of concept lattice after clustering. ToscanJ, a Java based package for creating CIS and viewing the different concept lattice is the tool used for implementing FCA.

TABLE OF CONTENTS

LIST OF FIGURES AND TABLES

LIST OF FIGURES AND TABLES

Chapter 1

INTRODUCTION

1.1 Introduction to MACHINE LEARNING

Machine learning is a branch of artificial intelligence concerned with the construction of programs that learn from experience. Learning may take many forms, ranging from learning from examples and learning by analogy to autonomous learning of concepts and learning by discovery.

According to Mitchell [10] "A computer program is said to learn from experience *E* with respect to some class of tasks *T* and performance measure *P*, if its performance at tasks in *T*, as measured by *P*, improves with experience *E"*.

Another researcher well known in the area of Machine Learning Witten and Frank [10] are of the view that "Things learn when they change their behavior in a way that makes them perform better in the future."

Hence we can conclude from the above definitions that the field of machine learning studies the design of computer programs able to induce patterns, regularities, or rules from past experiences. Learner (a computer program) processes data D representing past experiences and tries to either develop an appropriate response to future data, or describe in some meaningful way the data seen [2]. Machine learning is the process or technique by which a device modifies its own behavior as the result of its past experience and performance. Learning is an inherent characteristic of the human beings. By virtue of this, people, while executing similar tasks, acquire the ability to improve their performance. Such learning is usually referred to as 'machine learning'. Machine learning can be broadly classified into three categories:

i) Supervised learning,
ii) Unsupervised learning and
iii) Reinforcement learning.

Supervised learning requires a trainer, who supplies the input-output training instances. The learning system adapts its parameters by some algorithms to generate the desired output patterns from a given input pattern. In case of unsupervised learning there are no trainers, the desired output for a given input instance is not known, and consequently the learner has to adapt its parameters autonomously. The third type called the reinforcement learning bridges a gap between supervised and unsupervised categories. In reinforcement learning, the learner does not explicitly know the input-output instances, but it receives some form of feedback from its environment. The feedback signals help the learner to decide whether its action on the environment is rewarding or punishable.

1.2 Data mining

It has been observed that the amount of data is doubling every three years and as more data is collected the need for extracting meaningful and useful information is growing. Data mining is the process of extracting hidden patterns from data and it is becoming an increasingly important tool to transform this data into information. It is commonly used in a wide range of profiling practices, such as marketing, surveillance, fraud detection and scientific discovery.

Data mining commonly involves four classes of task:

- Classification – It refers to arranging the data into predefined groups, for example an email program might attempt to classify an email as legitimate or spam. Common algorithms used for classification include nearest neighbor, neural network, etc. Classification requires supervised learning, i.e., the training data has to specify what we are trying to learn (the classes).

- Clustering - Is like classification but the groups are not predefined, so the algorithm will try to group similar items together. No predefined classification is required in clustering of the data. The task is to learn a classification from the data. Clustering algorithms divide a data set into natural groups (clusters). Instances in the same cluster are similar to each other, they share certain

properties clustering is an unsupervised task, i.e., the training data doesn't specify what we are trying to learn (the clusters).

- Regression - Attempts to find a function which models the data with the least error. A common method is to use Genetic Programming.

- Association rule learning - Searches for relationships between variables.

1.3 Clustering

A *cluster* is therefore a collection of objects which are "similar" between them and are "dissimilar" to the objects belonging to other clusters [9]. The goal of clustering is to determine the intrinsic grouping in a set of unlabeled data. Clustering is an **unsupervised learning method.**

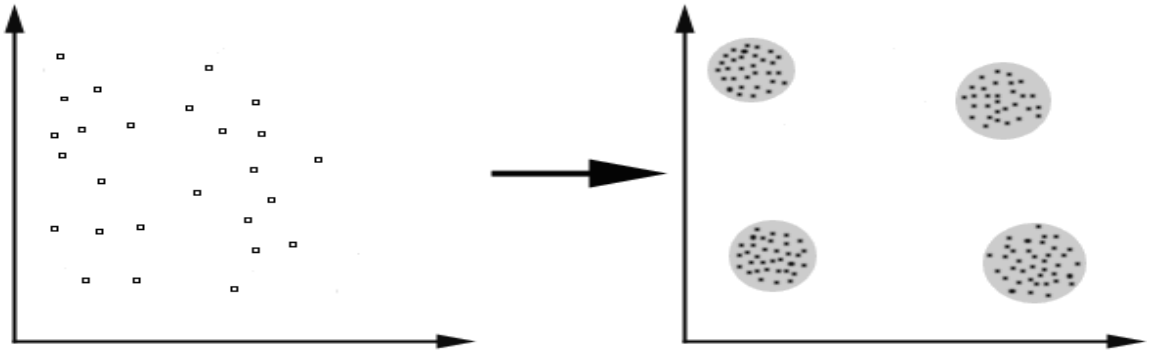


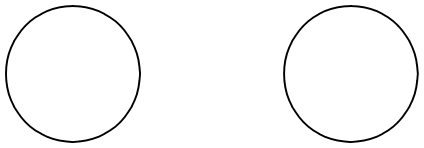
Figure 1.1 : Clustering.

Figure1.1 shows how four clusters are formed such that objects of one cluster are similar to each other whereas objects of different clusters are dissimilar. One often wants the objects to be as similar to objects in the same cluster and as dissimilar to objects from other clusters as possible. In text clustering the objects are texts or documents. Text clustering can for instance be applied to the documents retrieved by a search engine, so that they can be presented in groups according to content. Cluster analysis groups data objects based on information found in the data that describes the objects and their relationship. The goal is that the objects within group be similar or related to one another

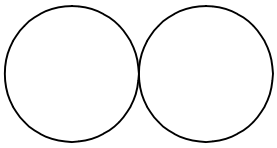
and different from the objects in the other groups. The greater the similarity within a group and greater the difference between groups, the better or more distinct is the clustering.

Different types of clusters:-

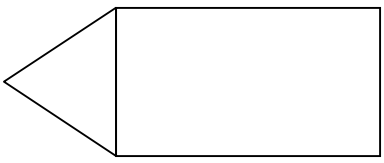
- Well separated: - A cluster is a set of objects in which each object is closer to every other object in the cluster then to any object not in the cluster. Figure 1.2(a) gives an example of well separated clusters that consist of two points in different groups is larger than the distance between any two points within group.
- Prototype based: - A cluster in which each object is closer to prototype that defines the cluster than to the prototype of any other cluster. For data with continuous attributes, the prototype of a cluster is often a centriod that is average of all the points in the cluster. When a centriod is not meaningful, such as when the data has categorical attributes, the prototype is medoid that is the most representative point of a cluster. For much type of data, the prototype can be regarded as the most central point and such prototype based clusters as center-based clusters. Figure 1.2(b) shows an example of center-based clusters.

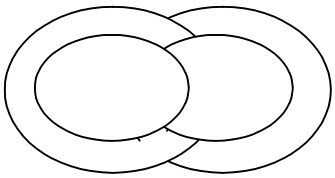


(a) Well-separated clusters. Each point Is closer to all of the points in its cluster Than to any point in another cluster.



(b) Center- based clusters Each point is closer to center of its cluster than to center of any other cluster.





(c) Conceptual clusters. Points in a cluster share some general property that derives from the entire set of points. Points in the intersection of the circles belong to both.

Figure 1.2: Different types of clusters.

- Shared-Property (Conceptual Clusters): - A cluster as a set of objects that share some property. This definition encompasses all the previous definitions of a cluster; for example, objects in a center-based cluster share the property that they are all closest to the same centriod or mediod. However, the shared property approach also includes new types of clusters. Considering the clusters shown in Figure 1.2(c), a triangular area (cluster) is adjacent to a rectangular one, and there are two intertwined circles (clusters). In both cases, a clustering algorithm would need a very specific concept of a cluster to successfully detect these clusters. The process of finding such clusters is called conceptual clustering.

- Density- Based: - A cluster is a dense region of objects that is surrounded by region of low density.

1.3.1 Clustering Algorithms

A clustering algorithm finds a partition of a set of objects that fulfills some criterion based on these conditions. Clustering algorithms are classified into three categories that are hierarchical clustering, partitioning clustering, and overlapping clustering. Hierarchical clustering is a method of cluster analysis which seeks to build a hierarchy of clusters. Hierarchical algorithms produce a hierarchy of clusters. Hierarchical clustering is a commonly used statistical tool for exploring relationships in statistical data. It clusters data based on a user defined measure called "distance", "Similarities", "correlation", are sometimes used in place of "distances", because users' definition of "distance" is related to "similarities" or "correlation" [19]. There are a large number of variants of hierarchical clustering. The differences are in the way distances are defined and computations (e.g., average-linkage, top-down) are implemented. While partitioning algorithms gives a flat partition of the set. The k-means algorithm is an evolutionary algorithm that gains its name from its method of operation. The algorithm clusters observations into k groups, where k is provided as an input parameter. It then assigns each observation to clusters based upon the observation’s proximity to the mean of the cluster [20]. The overlapping clustering uses fuzzy sets to cluster data, so that each point

may belong to two or more clusters with different degrees of membership. In this case, data will be associated to an appropriate membership value. In a hard clustering each object belongs to only one cluster for example hierarchal and k-means clustering. While soft clustering each object can belong to multiple clusters are usually inefficient. In fuzzy clustering objects belong to more than one cluster usually with a degree of membership.

1.4 Conceptual clustering

Common clustering techniques have the disadvantage that they do not provide intentional descriptions of the clusters obtained. Conceptual Clustering techniques, on the other hand, provide such descriptions, but are known to be rather slow.

Conceptual clustering is a machine learning paradigm for unsupervised classification developed mainly during the 1980s. It is distinguished from ordinary data clustering by generating a **concept description** for each generated class. Most conceptual clustering methods are capable of generating hierarchical category structures; A hierarchy is an arrangement of items (objects, names, values, categories, etc.), in which the items are represented as being "above," "below," or "at the same level as" one another. A hierarchy can link entities either directly or indirectly, and either vertically or horizontally. The only direct links in a hierarchy are to one's immediate superior or to one of one's subordinates, although a system that is largely hierarchical can also incorporate other organizational patterns. Indirect hierarchical links can extend "vertically" upwards or downwards via multiple links in the same direction. Conceptual clustering is closely related to formal concept analysis (FCA), decision tree learning, and mixture model learning.

Conceptual clustering vs. Data clustering: - Conceptual clustering is obviously closely related to data clustering; however, in conceptual clustering it is not only the inherent structure of the data that drives cluster formation, but also the description language which is available to the learner. Thus, a statistically strong grouping in the data may fail to be

extracted by the learner if the prevailing concept description language is incapable of describing that particular *regularity*.

Concepts are necessary for expressing human knowledge. Therefore, the process of discovering knowledge in databases benefits from a comprehensive formalization of concepts which can be activated to communicatively represent knowledge coded in databases. *Formal Concept Analysis* offers such formalization by mathematizing concepts that are understood as units of thought constituted by their extension and intension. **Conceptual information system (CIS)** is a system which stores information based on the concepts. Conceptual Information Systems are based on a formalization of the concept of `concept'. CIS analyze data on the basis of concepts. These are based on the Formal concept analysis and can be implemented using tool ToscanaJ.

Formal Concept Analysis (FCA) is a method mainly used for the analysis of data, i.e. for deriving implicit relationships between objects described through a set of attributes on the one hand and these attributes on the other. The data are structured into units which are formal abstractions of concepts of human thought, allowing meaningful comprehensible interpretation. Thus, FCA can be seen as a conceptual clustering technique as it also provides intentional descriptions for the abstract concepts or data units it produces. Central to FCA is the notion of a *formal context*. Formal concept analysis refers to both an unsupervised machine learning technique and, more broadly, a method of data analysis. The approach takes as input a matrix specifying a set of objects and the properties thereof, called attributes, and finds both all the "natural" clusters of attributes and all the "natural" clusters of objects in the input data, where

A "natural" *object* cluster is the set of all objects that share a common subset of attributes, and

A "natural" *property* cluster is the set of all attributes shared by one of the natural object clusters.

Natural property clusters correspond one-for-one with natural object clusters, and a **concept** is a pair containing both a natural property cluster and its corresponding natural object cluster. The family of these concepts obeys the mathematical axioms defining a lattice, and is called a **concept lattice** (in French this is called a Treillis de Galois because the relation between the sets of concepts and attributes is a Galois connection). Concept lattices are structures on concepts which are expressed as *intent* (the distinguishing features of the concept) and *extent* (the individuals which instantiate the concept). The features in the intent are called *attributes* and the individuals used re named *objects*.

1.5 Thesis Outline

This thesis is divided into six chapters. In this chapter, a brief introduction of the work was provided.

Chapter 2 – A brief review of literature studied in the area of clustering, CIS and FCA are provided.

Chapter 3 – This chapter concentrates on problem statement after going through literature survey.

Chapter 4 – The general concepts used in the area of conceptual clustering, Formal Concept Analysis and tools used for the implementation of FCA are discussed in brief.. This chapter serves as theoretical background for the future work.

Chapter 5 – An evaluation model of this thesis is described. The experiments performed and the results evaluated are discussed.

Chapter 6 - The conclusion of this thesis and a summary of all experiences in this work is described. Finally, the future work is stated.

At the end of this thesis, the appendix and references of this thesis are presented.

Chapter 2

LITERATURE REVIEW

Cluster analysis divides data into groups (clusters) that are meaningful and useful. If meaningful groups are the goal, then the clusters should capture the natural structure of the data. In some cases, however, cluster analysis is only a useful starting point for other purpose, such as data summarization. Whether for understanding or utility, cluster analysis has long played an important role in a wide variety of fields: psychology and other social science, biology, statistics, pattern recognition, information retrieval, machine learning and data mining [14].

The term *cluster analysis* (first used by Tryon, 1939) encompasses a number of different algorithm and methods for grouping objects of similar kind into respective categories. A general question facing researchers in many areas of inquiry is how to *organize* observed data into meaningful structures, that is, to develop taxonomies. In other words cluster analysis is an exploratory data analysis tool which aims at sorting different objects into groups in a way that the degree of association between two objects is maximal if they belong to the same group and minimal otherwise. Cluster analysis can be used to discover structures in data without providing an explanation/interpretation. In other words, cluster analysis simply discovers structures in data without explaining why they exist [13].

Clustering methods can be divided into two basic types: hierarchical and partitioned clustering. Within each of the types there exists a wealth of subtypes and different algorithms for finding the clusters. Hierarchical clustering proceeds successively by either merging smaller clusters into larger ones, or by splitting larger clusters. Partition based clustering, on the other hand, attempts to directly decompose the data set into a set of disjoint clusters. The criterion function that the clustering algorithm tries to minimize may emphasize the local structure of the data, as by assigning clusters to peaks in the probability density function, or the global structure. Typically the global criteria involve

minimizing some measure of dissimilarity in the samples within each cluster, while maximizing the dissimilarity of different clusters [16].

Clustering technique so far used is based on partitioning algorithm and an alternative approach that has gained in interest recently is to apply the concept lattice defined in Formal Concept Analysis commonly referred to as FCA. Formal Concept Analysis (FCA) is a mathematical approach used for conceptual data analysis and knowledge processing. These clusters, known as formal concepts, are characterized by a set of objects and a set of attributes or set of documents and set of terms when applied to text retrieval. The advantage of FCA is that users can refine their query by browsing through well defined clusters in the form of a graph [12].

In Formal Concept Analysis, the elements of one type are called “formal objects”, the elements of the other type are called “formal attributes”. The adjective “formal” is used to emphasize that these are formal notions. “Formal objects” need not be “objects” in any kind of common sense meaning of “object”. But the use of “object” and “attribute” is indicative because in many applications it may be useful to choose object-like items as formal objects and to choose their features or characteristics as formal attributes [11].

A few years ago, there was no open-source FCA software available. The availability of several Java-based open source tools since 2003, such as ConExp (sourceforge.net/projects/conexp) and ToscanaJ is probably another contributing factor for the recent growth of interest in FCA. These tools are cross-platform compatible, easy to install and fairly easy to use [11].

For over a decade, work on Formal Concept Analysis has been accompanied by the development of the Toscana software. Toscana was implemented to realize the idea of Conceptual Information Systems which allow the analysis of data using concept-oriented methods. Over the years, many ideas from Formal Concept Analysis have been tested in Toscana systems while the real-world problems encountered led to new theoretical research. After ten years of development, the ToscanaJ project was initiated to solve some outstanding problems of the older Toscana versions. The ToscanaJ suite provides

programs for creating and using Conceptual Information Systems. The experience with older Toscana implementations has been applied to the design of the programs. A workflow that developed through many Toscana projects has now been integrated into the tools to make them easier to use. Implemented as an Open-Source project and embedded into the larger Tockit project, ToscanaJ is also a starting point for creating a common base for software development for Formal Concept Analysis [6].

After going through the literature related to FCA we decided to work on the Toscana J software to find the viability of using this concept to textual data and further in the area of semantic web.

Chapter 3

PROBLEM STATEMENT

During the literature survey it was found that there are various techniques for text clustering with major ones being hierarchical or partitioning clustering. Hierarchical clustering is a method of cluster analysis which seeks to build a hierarchy of clusters. Partitioning algorithms gives a flat partition of the set. The k-means algorithm is an example of partition algorithm. Another technique is the concept based clustering that is FCA (formal concept analysis). Formal Concept Analysis (FCA) is a method mainly used for the analysis of data, i.e. for deriving implicit relationships between objects described through a set of attributes on the one hand and these attributes on the other. The data are structured into units which are formal abstractions of concepts of human thought, allowing meaningful comprehensible interpretation. Thus, FCA can be seen as a conceptual clustering technique as it also provides intentional descriptions for the abstract concepts or data units it produces. Formal Concept Analysis (FCA) is a learning technique for discovering conceptual structures in a large amount of data.

After going through few paper and material on FCA it was further analyzed that information organization and user's query retrieval can be solved more logically and conceptually using Formal Concept Analysis (FCA) compared to classic document clustering. Advantages of FCA over standard document clustering algorithms are:

a) FCA provides an intentional description of each document cluster that can be used for query modification or refinement, making groups more interpretable;
b) The clustering organization is a lattice, rather than a hierarchy, which is more natural when multiple classifications are possible, and facilitates recovering from bad decisions while browsing the lattice to find relevant information.

We will take some raw data from an organization, arrange it in conceptual form and then identify how concept lattice can be derived. We plan to use ToscanaJ, a open source tool to derive the conceptual lattice. This work can be further extended for conceptual clustering of Web pages.

Chapter 4
FORMAL CONCEPT ANALYSIS

4.1 Formal Concept Analysis

4.1.1 History

FCA was invented by Rudolf Wille in the early 80s (Wille, 1982). FCA was developed mainly by a small group of researchers and Wille's students in Germany. For first 10 years, FCA was implemented in several larger-scale applications for example implementation of a knowledge exploration system for civil engineering. During the last 10 years, FCA has grown into an international research community with applications in many disciplines, such as linguistics, software engineering, psychology, AI and information retrieval. Formal Concept Analysis (FCA) was introduced for modeling the concept 'concept' in terms of lattice theory [11].

4.1.2 Introduction

Formal Concept Analysis (FCA) is a method for data analysis, knowledge representation and information management.

Definition: *A formal context is a triple K: = (G; M; I), where G is a set of objects, M is a set of attributes, and I is a binary relation between G and M (i. e. I ⊆ G X M), (g;m) € I is read "object g has attribute m" [1].*

A concept is a unit of thoughts consisting of two parts, the extension and the intension. The extension covers all objects belonging to this concept and the intension comprises all attributes valid for all those objects. Hence objects and attributes play a prominent role together with several relations like e.g. the hierarchical "sub-concept – super-concept"

relation between concepts and the incidence relation "an object has an attribute". Attributes “are units of thought which are gained by abstraction, and hence they are also concepts. For building concepts, one always needs other concepts, which then play the role of attributes”. Concepts are necessary for expressing human knowledge. Therefore, the process of discovering knowledge in databases benefits from a comprehensive formalization of concepts which can be activated to communicatively represent knowledge coded in databases. *Formal Concept Analysis* offers such formalization by mathematizing concepts that are understood as units of thought constituted by their extension and intension. A *conceptual information system* consists of a (relational) database and a collection of formal contexts, called *conceptual scales*, together with line diagrams of their concept lattices; such systems can be implemented with the management system TOSCANA. For a chosen conceptual scale, TOSCANA presents a line diagram of the corresponding concept lattice indicating all objects stored in the database in their relationships to the attributes of the scale.

Example

A concept lattice for objects consisting of the integers from 1 to 10, and attributes composite (c), square (s), even (e), odd (o) and prime (p). Data is shown in table 4.1.

	composite	**even**	**odd**	**prime**	**square**
1			√		√
2		√		√	
3			√	√	
4	√	√			√
5			√	√	
6	√	√			
7			√	√	
8	√	√			
9	√		√		√
10	√	√			

Table 4.1: Context table for ten integers [3].

The lattice is drawn as a line diagram. Consider O = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}, and A = {composite, even, odd, prime, square}. The smallest concept including the number 3 is the one with objects {3, 5, 7}, and attributes {odd, prime}, for 3 has both of those attributes and {3, 5, 7} is the set of objects having that set of attributes. The largest concept involving the attribute of being square is the one with objects {1,4,9} and attributes {square}, for 1, 4 and 9 are all the square numbers and all three of them have that set of attributes. The full set of concepts for these objects and attributes is shown in the illustration. It includes a concept for each of the original attributes: the composite numbers, square numbers, even numbers, odd numbers, and prime numbers. Additionally it includes concepts for the even composite numbers, composite square numbers (that is, all square numbers except 1), even composite squares, odd squares, odd composite squares, even primes, and odd primes.

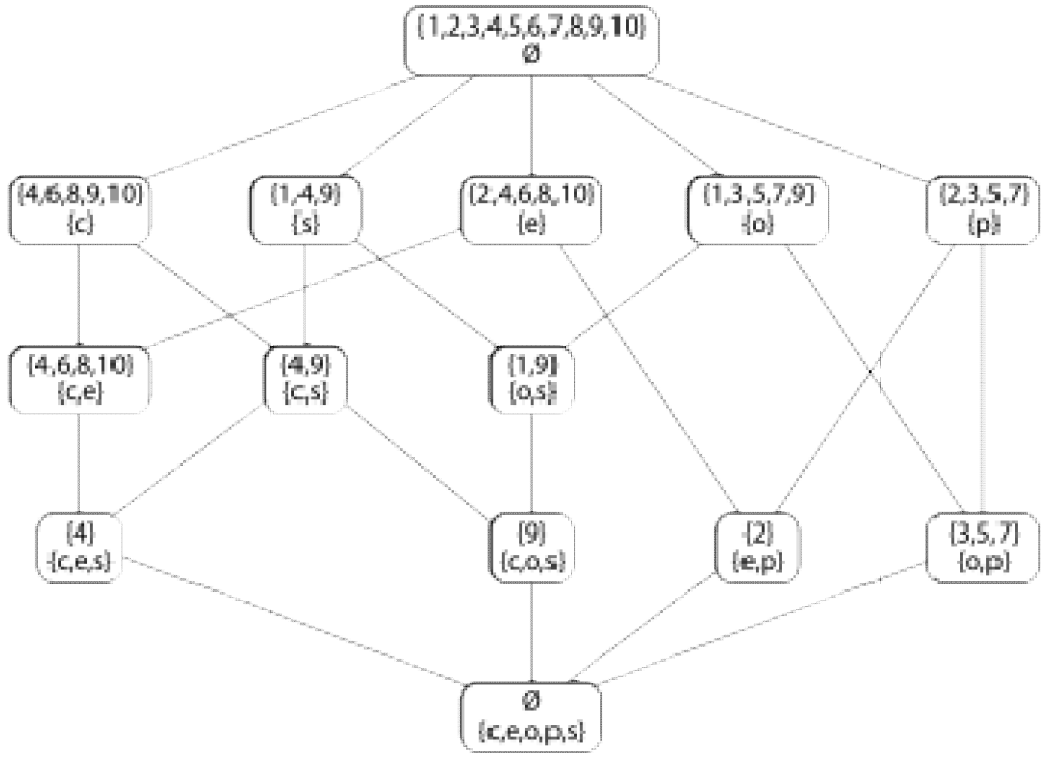


Figure 4.1: Line Diagram for context table 4.1[3].

4.1.3 Contexts and concepts

A *(formal) context* consists of a set of objects O, a set of attributes A, and an indication of which objects have which attributes. Formally it can be regarded as a bipartite graph $I \subseteq O \times A$.

A *(formal) concept* for a context is defined to be a pair (O_i, A_i) such that

1. $O_i \subseteq O$
2. $A_i \subseteq A$
3. every object in O_i has every attribute in A_i
4. for every object in O that is not in O_i, there is an attribute in A_i that the object does not have
5. for every attribute in A that is not in A_i, there is an object in O_i that does not have that attribute

O_i is called the extent of the concept, A_i the intent. A context may be described as a table, with the objects corresponding to the rows of the table, the attributes corresponding to the columns of the table, and a Boolean value (in the example represented graphically as a checkmark) in cell (x, y) whenever object x has value y. A concept, in this representation, forms a maximal sub array (not necessarily contiguous) such that all cells within the sub array are checked. For instance, the concept highlighted with a different background color in the example table is the one describing the odd prime numbers, and forms a 3×2 sub array in which all cells are checked.

Concept lattice of a context

The concepts (O_i, A_i) defined above can be partially ordered by inclusion: if (O_i, A_i) and (O_j, A_j) are concepts, we define a partial order $\leq$ by saying that $(O_i, A_i) \leq (O_j, A_j)$ whenever $O_i \subseteq O_j$. Equivalently, $(O_i, A_i) \leq (O_j, A_j)$ whenever $A_j \subseteq A_i$. Every pair of concepts in this partial order has a unique greatest lower bound (meet). The greatest lower bound of (O_i, A_i) and (O_j, A_j) is the concept with objects $O_i \cap O_j$; it has as its attributes the union of A_i, A_j, and any additional attributes held by all objects in $O_i \cap O_j$.

Symmetrically, every pair of concepts in this partial order has a unique least upper bound (join). The least upper bound of (O_i, A_i) and (O_j, A_j) is the concept with attributes $A_i \cap A_j$; it has as its objects the union of O_i, O_j, and any additional objects that have all attributes in $A_i \cap A_j$.

These meet and join operations satisfy the axioms defining a lattice. In fact, by considering infinite meets and joins, analogously to the binary meets and joins defined above, one sees that this is a complete lattice. Conversely, any finite lattice may be generated as the concept lattice for some context. For, let L be a finite lattice, and form a context in which the objects and the attributes both correspond to elements of L. In this context, let object x have attribute y exactly when x and y are ordered as $x \leq y$ in the lattice. Then, the concept lattice of this context is isomorphic to L itself.

Recovering the context from the line diagram

The line diagram of the concept lattice (also called, in formal concept analysis, a *line diagram*), encodes enough information to recover the original context from which it was formed. Each object of the context corresponds to a lattice element, the element with the minimal object set that contains that object, and with an attribute set consisting of all attributes of the object. Symmetrically, each attribute of the context corresponds to a lattice element, the one with the minimal attribute set containing that attribute, and with an object set consisting of all objects with that attribute. We may label the nodes of the line diagram with the objects and attributes they correspond to; with this labeling, object x has attribute y if and only if there exists a monotonic path from x to y in the diagram. An important advantage of FCA is that the Galois connections and the sets of formal concepts can be visualized.

In Table 4.2 a formal context is given for example animal. The elements on the left side are formal objects; the elements at the top are formal attributes; and the relation between them is represented by the crosses. In this example, the formal objects are animals that are famous in certain parts of the world: the cartoon characters.

ANIMAL	Cartoon	Real	Tortoise	dog	Cat	mammal
LION	**X**				**X**	**X**
FINCH	**X**			**X**		**X**
EAGLE		**X**			**X**	**X**
HARE		**X**		**X**		**X**
OSTRICH		**X**	**X**			

Table 4.2 Context Table for example animals.

Figure 4.2 shows a line diagram of a concept lattice corresponding to the formal context in Table 4.2. A concept lattice consists of the set of concepts of a formal context and the sub-concept super-concept relation between the concepts. The nodes in figure 4.2 represent formal concepts. Formal objects are noted slightly below and formal attributes slightly above the nodes which they label.

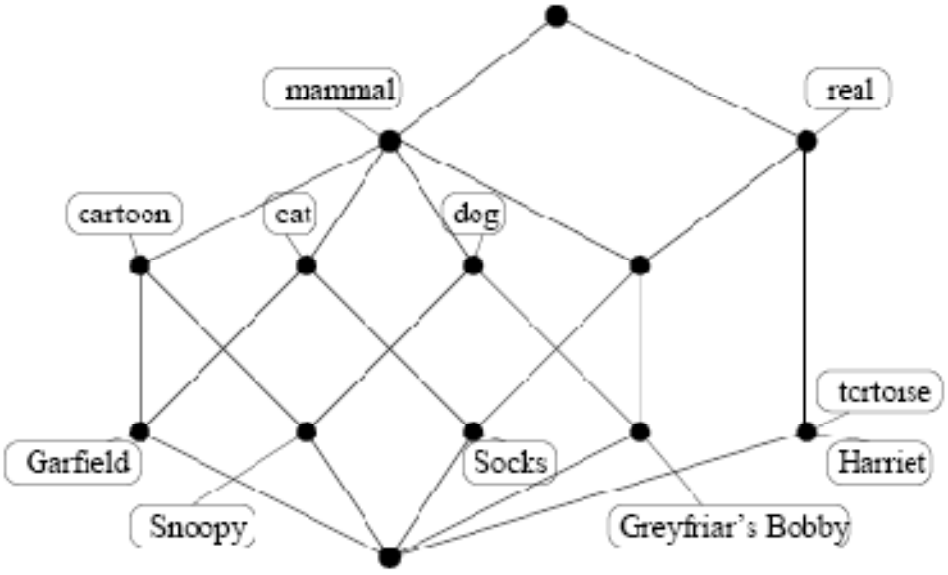


Figure 4.2: A concept lattice for the formal context in table 4.2[11].

In figure 4.2 the node on the right side which is labeled with the formal attribute "real" shall be referred to as Concept A. To retrieve the extension of a formal concept one needs

to trace all paths which lead down from the node to collect the formal objects. In this example, the formal objects of Concept A are Socks, Bobby and Harriet.

To retrieve the intension of a formal concept one needs to trace all paths which lead up in order to collect all the formal attributes. In this example, there is a node above Concept A but that node has no formal attributes attached. Thus Concept A represents the formal concept with the extension "Harriet, Bobby, Socks" and the intension as the single-element set "real". The other formal concept is the one with extension "Socks, Bobby" and intension "real, mammal". This concept is connected to Concept A by an edge (the line going down from Concept A to the left) and not labeled by any object or attribute in the line diagram in figure 4.2. This concept shall be referred to as Concept B.

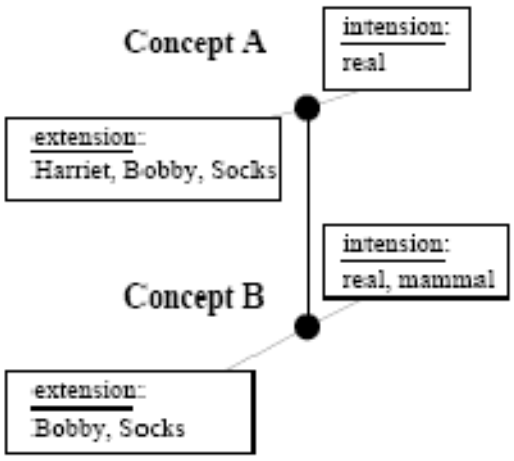


Figure 4.3: A sub concept-super concept relation [11].

Figure 4.3 summarizes the relationship between Concept A and Concept B. Concept B is a sub concept of Concept A because the extension of Concept B is a subset of the extension of Concept A and the intension of Concept B is a superset of the intension of Concept A. All edges in the line diagram of a concept lattice represent this sub concept-super concept relation.

The top and bottom concepts in a concept lattice are special. The top concept has all formal objects in its extension. Its intension is often empty but does not need to be empty.

In the example in Figure 4.2, the top concept could have a formal attribute “animal”. The bottom concept has all formal attributes in its intension. If any of the formal attributes mutually exclude each other (such as “dog” and “cat”) then the extension of the bottom concept must be empty (because no formal object can be a dog and cat at the same time). The top concept can be thought of as representing the “universal” concept and the bottom concept the “null” or “contradictory” concept of a formal context. The subconcept-superconcept relation is transitive, which means that a concept is subconcept of any concept which can be reached by traveling upwards from it. If a formal concept has a formal attribute then its attributes are inherited by all its subconcepts. This corresponds to the notion of “inheritance” used in the class libraries of object-oriented modeling.

4.1.4 SCALING: THE TRANSFORMATION OF DATA INTO CONTEXTS

Data is saved generally in the form of tables in database. Data are often given in a format of the following form Table 4.2. This Table might be a part of a questionnaire. Usually there are some missing values in the data which are indicated in Table 4.2 by the slash "/". This is many valued context table having two attributes sex and age. In general, attributes may not only be properties which are or are not related to an object, but they may allow for different values. We call such attributes, as e.g. color, sex, weight, age **many-valued attributes**.

Def.: A many-valued context (G, M, W, I) consists of sets G, M, and W and ternary relation I between G, M and W (i.e. I ⊆ G x M x W), where the following holds:

(g, m, w) € I and (g, m, v) € I imply w = v.

The elements of G are called objects, the elements of M (many-valued) attributes and the elements of W attribute values.

(g, m, w) € I is read as „attribute m has value w for object g“. [17]

	Sex	Age
ADAM	M	21
BETTY	F	50
CHRIS	/	66
DORA	f	88
EVA	F	17
FRED	M	/
GEORGE	M	90
HARRY	M	50

Table 4.3 Many Valued Data.

To implement FCA formal context is required, so there is need to transfer this data in Table 4.3 in formal contexts in Table 4.4.

	Sex		Age				
	M	F	<18	<40	<=65	>65	>=80
ADAM	X			X	X		
BETTY		X			X		
CHRIS						X	
DORA		X				X	X
EVA		X	X	X	X		
FRED	X						
GEORGE	X					X	X
HARRY	X				X		

Table 4.4 Representation of Table 4.3 as Formal Contexts.

The many-valued context is transformed by **conceptual scaling** (as described below) to a one-valued context, for which one then can compute formal concepts. Conceptual Scaling involves the human expert, as she/he has several choices how to interpret the data. For scaling, each attribute of the many-valued context is represented by a formal context, called **conceptual scale**.

In the beginning of a data exploration one should first choose a rough view to get an overview over the data. In example consider the values of the age, using some new attributes (the scale attributes) given in the second line of Table 4.3 under the header "age". The rule how to transform the two many-valued columns into the (seven) columns

of a formal context are given by the following two formal contexts, called scales. Scale S1 in Table 4.5 is for attribute sex having two scale attributes male and female and two objects that are also male and female.

S1	**M**	**F**
M	**X**	
F		**X**
/		

Table 4.5 Conceptual Scale for sex Attribute.

Similarly second scale S2 in Table 4.6 for attribute age having five scales attributes and data is divided into six objects.

S2	<18	<40	<=65	>65	>=80
17	**X**	**X**	**X**		
21		**X**	**X**		
50			**X**		
66				**X**	
88				**X**	**X**
90				**X**	**X**
/					

Table 4.6 Conceptual Scale for Age Attribute.

These scales represent the language describing the chosen view. The meaning of the scale attributes can be visualized by the Figure 4.4 for sex scale S1 and Figure 4.5 for age scale S2 line diagrams of the scales:

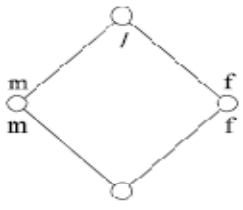


Figure4.4: Line diagram for S1.

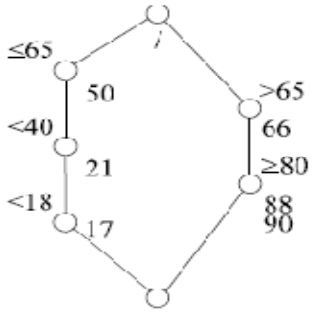


Figure4.5: Line diagram for S2.

In both scales the slash "/" as an object of the scale doesn't have any scale attribute, hence its object concept is the top concept. The line diagram of the age scale shows that the attributes of this scale divide the age values into two classes, say the young and the old ones, and each class is ordered by a chain. Therefore this scale is called a biordinal scale. Now look at the two sub contexts spanned by the scale attributes of sex and age. The corresponding concept lattices are given by the line diagrams in Figure 4.6 and Figure 4.7. These diagrams can be obtained from the scale diagrams above by replacing each value (e.g. the age value 50) by the set of all people with this value (e.g. BETTY, HARRY).

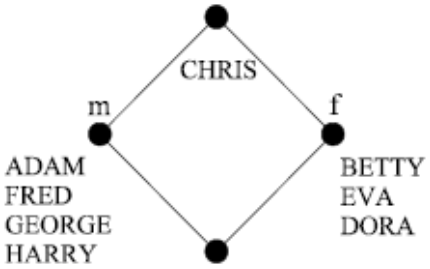


Figure 4.6: Line diagram with with objects of sex.

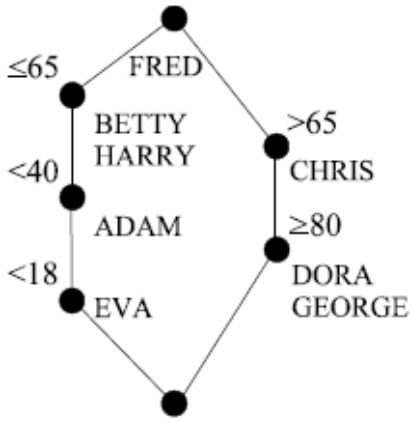


Figure 4.7: Line diagram with objects of age.

This very simple sorting procedure gives us for each many-valued attribute the distribution of the objects in the line diagram of the chosen scale. The well-known histograms for one variable arise as special cases from line diagrams.

4.2 FCA based information system

Information Retrieval (IR) has always been a major concern in Formal Concept Analysis (FCA). Indeed, an obvious analogy exists between object-attribute and document-term tables. Accordingly, formal concepts of a concept lattice may be seen as a pair (*answer*, *query*) where the *query* corresponds to the intent of the concept while the *answer*

corresponds to the extent of the concept. The subsumption relation between formal concepts can be considered as a specialization/generalization relation between such queries. Moreover, the way formal concepts are classified in a concept lattice allows an easy browsing (navigation) of the lattice structure and hence provides a second way for using concept lattices in IR, namely IR by navigation. The two forms of IR using concept lattices (by querying and by browsing) can easily be combined. Such a combination provides more precise results retrieved in a flexible way. In fact, a query can first be submitted to a lattice-based IR system to locate the formal concept containing the most precise answer. Once the answer concept is identified, additional results can be identified by browsing the concept lattice.

Using the ranked list of documents retrieved by a search engine, we generate a concept lattice to organize these search results. Lattices generated are based on a formal context $K := (G, M, I)$, where $G = \{doc1, doc2, \ldots, docn\}$ represents a subset of the retrieved documents, $M = \{desc1, desc2, \ldots, desck\}$ is a subset of document descriptors and I is the incidence relationship.

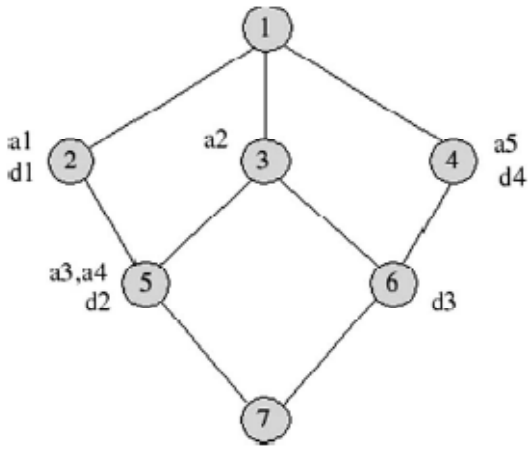


Figure 4.8: Example concept lattices [18].

This model relies on the set of concepts generated and its corresponding concept lattice, while introducing some assumptions about what concept information is going to be considered for showing, browsing or evaluation purposes. Information nodes are based on the assumption that a concept node should not display all its extent information. Working with the whole extent implies no differences between those documents which

are object concepts (i.e. they are not going to appear as extent components of lower nodes) and those documents that can be specialized. Figure 4.8 show, as an example, a concept lattice. Showing concept extent implies, for instance, that a user located at the top node of the lattice would be seeing the whole list of the documents retrieved at once. This situation would make system essentially identical to a ranked list for browsing purposes. The use of information nodes overcomes this problem, granting the document access only when no more specialization is possible. This model agrees with the access model used by most web directories (e.g. Open Directory Project ODP or Yahoo! Directory), where it is possible to find categories with no documents (i.e. categories that being very general do not completely describe any web page).

4.3. ToscanaJ

4.3.1 History

Toscana's history starts in the early 90's with a first prototypical version. The first official version, Toscana 2, was implemented by Martin Skorsky based on the Formal Concept Analysis. In 2000, Bernd Groh developed the last version of Toscana based on the original source code: Toscana 3. By abandoning some of the legacy code and functions, he was able to solve some of the major problems. Additionally, he introduced some features that accelerated the development of Conceptual Information Systems. An Open Source project was started to provide a tool that is well adapted to the workflow of Toscana systems on the user interface level, and at the same time to provide source code that is easily extensible for interested programmers. In mid 2001, the KVO workgroup4 started working on this project. One of the ideas created in the early stages of this project is to create a large, flexible framework for conceptual knowledge processing. This idea has by now been established as a separate project called Tockit. In the long run, ToscanaJ is expected to be just one of many Tockit applications. The ToscanaJ project had aims to change other aspects of the program. Right from the start of the project, one of the major aims was to create a user interface for ToscanaJ that was simpler than the existing one. In many ways small enhancements to the user interface have been made to remove clutter

and to make important features more easily accessible. This is of course not as efficient as redesigning the workflow, but still the interface of ToscanaJ is simpler without posing restrictions on the user. The ToscanaJ project also aims at interoperability, reuse and flexibility. This was the reason to choose Java as the programming language. As it was implemented with the current Java development kit, it is possible to run ToscanaJ not only on Windows, but also on UNIX/Linux and Mac OS operating systems for which the Java runtime environment 1.4 is available [6].

4.3.2 Introduction

The ToscanaJ suite, latest version of Toscana, provides programs for creating and using Conceptual Information Systems and a workflow that developed through many Toscana projects has now been integrated into the tools to make them easier to use. Implemented as an Open-Source project and embedded into the larger Tockit project, ToscanaJ is also a starting point for creating a common base for software development for Formal Concept Analysis. ToscanaJ is a browsing frontend for *Conceptual Information Systems (CIS)* in the tradition of the Windows-based Toscana tools.

Conceptual information system- a system that is able to store and present information to a user and systems based on the notion of a "concept". While "concept" seems to be an obvious notion to humans, most data-analysis software does not have this notion at all – these systems use numbers for everything. Conceptual Information Systems try to achieve a better understanding by analyzing data using the notion of "concepts". Conceptual Information Systems are based on Formal Concept Analysis (FCA), a mathematical method to structure and analyze data. Conceptual Information Systems store, process, and present information using concept-oriented representations support tasks like data analysis, information retrieval, or theory building in a human centered way [6]. The scenario for Conceptual Information Systems typically starts with a data set which a user wants to explore using concept-oriented methods. In Fig. 4.9 the principal components of a Conceptual Information System and related roles are shown.

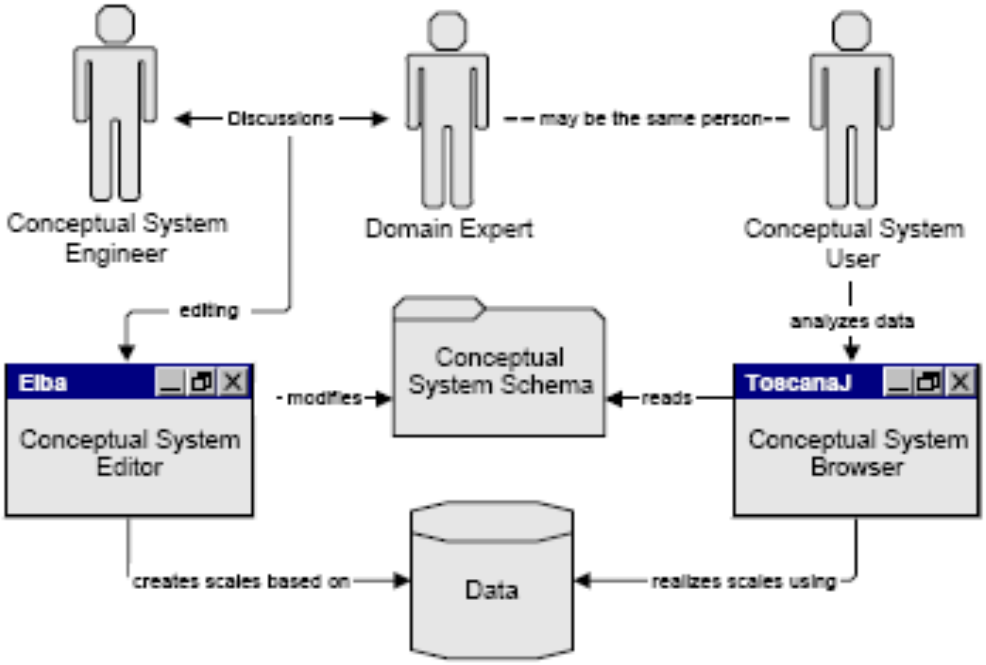


Figure 4.9: Components and Workflow of a Conceptual Information System [6].

The system is created by a conceptual system engineer in cooperation with a domain expert. They combine the engineer's knowledge about tools and theory with the expert's knowledge about the domain. Together they define the conceptual structures used to access the information in the system. These structures are understood as parts of the expert's knowledge being made explicit and therefore available to all users of the system. Using the conceptual system editor, the engineer stores the information about the conceptual structures and other information into a central repository, called a conceptual system schema. The schema is read by the conceptual system browser which allows the user to interact with the information using the conceptual structures. When implementing a Conceptual Information System using methods of Formal Concept Analysis, the data is modeled mathematically by a many-valued context and is transformed via conceptual scaling. This means that a formal context called conceptual scale is defined for each of the many-valued attributes which has the values of the attribute as objects. Structurally, there is no distinction between conceptual scales and formal contexts, but the notion of a conceptual scale does imply that the context is used to interpret an aspect of the many-valued context. If a many-valued context and a conceptual scale are given, we can derive the realized scale – a formal context which has the objects of the many-valued context as

objects and the attributes of the scale as attributes. In the realized scale, an object has an attribute if the value assigned to the object in the many valued context has the attribute in the conceptual scale.

ToscanaJ is a browsing frontend i.e. which means editing cannot be done on it. The ToscanaJ suite comes with editors for different types of CIS (with or without relational database in the backend), but editing can be a complex task and this complexity is not exposed to the user of the final system. There is no way to change the conceptual information system by using ToscanaJ, which means its users cannot break anything. The core notion in a ToscanaJ system is the notion of a line diagram. These line diagrams are a particular type of diagram, called Hasse diagrams. A line diagram in ToscanaJ represents a mathematical structure called a *concept lattice*. Concept lattices are structures on concepts which are expressed as *intent* (the distinguishing features of the concept) and *extent* (the individuals which instantiate the concept). The features in the intent are called *attributes* and the individuals used re named objects. These notions are modeled mathematically and the structures are calculated by the ToscanaJ software.

The ToscanaJ tools support connections to the embedded database engine, which is able to read data from standard SQL scripts, to JDBC and ODBC connected databases or directly to MS Access databases as files. After choosing one type, the user enters the necessary information for connecting to the specific database. Elba will then try to connect to the specified database. Tosconaj use this csx file, crested by Elba, to display the concept lattice that is line diagram.

Very early in the ToscanaJ project it was decided to create a suite of three components for creating and viewing Conceptual Schemas in the style of the old TOSCANA:

- ToscanaJ as viewer component optimized for access by inexperienced users
- Elba as creation tool for conceptual information system
- Siena as creation tool for standalone systems

As the conceptual scales and the line diagrams have to be defined beforehand, the life-cycle of a Toscana information system can be separated into two phases, namely the creation and the usage phase. In the creation phase, the conceptual system engineer creates the set of scales and diagrams using Elba editor of Toscanaj. This information is stored together along with information for accessing the database and optionally with additional visualization information in the conceptual system schema. Secondly, in the usage phase, users are able to analyze and explore the data by re-using those diagrams and choosing views from the manifold of possible composite conceptual scales. ToscanaJ information systems aim to facilitate the second phase.

4.3.2.1 Elba

Elba is an editor used to create Conceptual Information Systems (CIS) for ToscanaJ, a database frontend and browser for CIS. Elba allows to connect to a relational database and to create a conceptual system on top of it. The whole program is database-aware and tries to support the user by supplying relevant information from the database at each step. Elba has a complete feature set to create conceptual systems. It does support most of ToscanaJ's features, only if we want to use the more advanced options like system-specific database views, there is need to edit the XML file created, otherwise the more comfortable user interface of Elba can be used.

Some features of Elba are:

- Editing of diagrams and contexts
- Database wizard helps you connecting to different databases (internal, JDBC, ODBC, MS Access files)
- Scale generators help creating different types of diagrams
- The diagram layout and manipulation is based on n-dimensional structures, making editing diagrams easy while ensuring additive drawings
- A grid can be used as guides for neat layout. This is supported by all manipulators
- A very simple XML editor allows adding the descriptions of the different elements

- An XML summary of the diagrams can be exported to analyze the realized diagrams, i.e. the coverage of the lattices by the objects in the database.

The standard workflow is starts with an examination of the data that should become the basis of the CIS. The data usually needs to be converted to a single-valued or many-valued context. Once the database has been prepared, Elba can be started. Four main steps to create a CIS are:

1. Connect Elba to the database
2. Conceptual Scaling / Creation of Diagrams
3. Create CSX file
4. Call the CSX from ToscanaJ

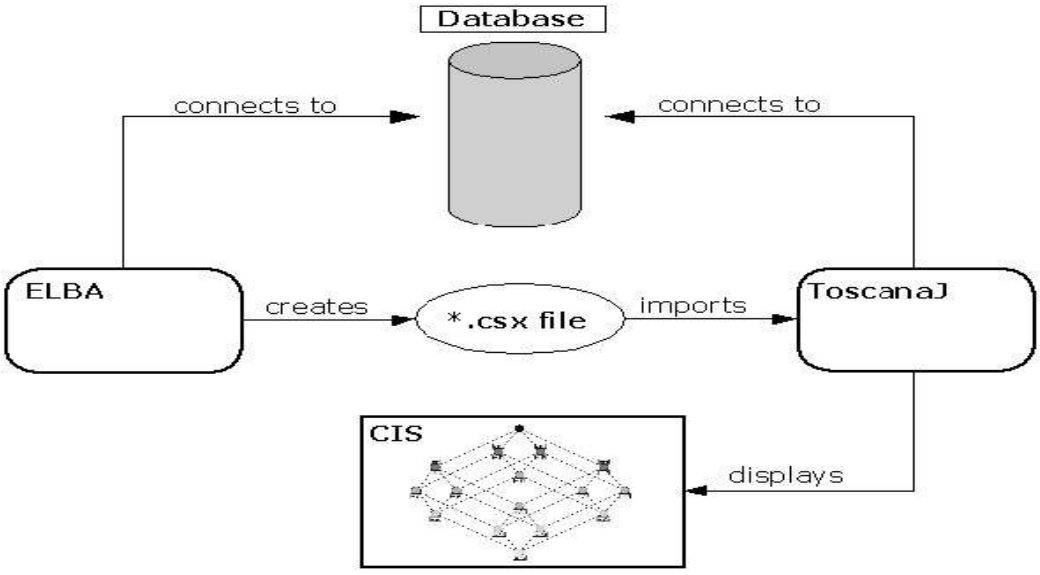


Figure 4.10: Elba and Toscanaj working [7].

Figure 4.10 gives the implementation details of Elba, used to create a CIS. First step is to Connect the database with Elba eidtor; Database of any type can be selected. Context table and diagram are designed using Elba which is .csx file. This .csx file is viewed using Toscanaj which is browsing frontend. ToscanaJ display the line diagram with

different options explained in implementation phase. Line diagrams cannot be changed in the ToscanaJ.

4.3.2.2 Siena

Siena ensures the system is suitable for appositions (as needed by ToscanaJ). This means the object sets of all contexts have to be identical, so whenever the file is saved Siena will change the object set of each diagram to the union of all objects in the system. It is assumed that objects not in an important context have none of the attributes specified in there, which means sometimes a new top concept needs to be created (whenever the old top concept had at least one attribute). This step is done only on save, not on import; it can be useful to first save the system before fine-tuning the diagram layouts. Elba is written to be database-aware and to guide the user through the workflow of creating a conceptual system on top of a relational database; Siena does similar things without the need of a database. This is achieved by a lot of shared code. Siena does not yet allow editing the data directly, it is planned to add support for editing many-valued contexts in a spreadsheet like manner in future version of Toscana.

4.3.3 Advanced Features in ToscanaJ

The features described in this section are only for advanced users. Some of the most interesting new features in ToscanaJ are:

- Color is used to indicate the size of the concept extent (number of objects belonging to the corresponding node),
- The set of nodes above and below a selected node can be highlighted by clicking on it, even in nested diagrams,
- diagrams can be exported in the SVG vector graphic format, which can be converted into other common graphic formats, e. g. Encapsulated Postscript (eps) or Portable Networks Graphic (png),

- not only simple lists or object counts can be queried from the underlying database, but also arbitrary aggregates and formatted combinations of columns, thus fulfilling certain requirements for more complex CKDD,
- And last but not least easy extensibility for new features.

In addition to this, ToscanaJ is the first TOSCANA running on multiple platforms and it can be run without a database – a feature that got lost early in the development of TOSCANA 2. Finally, the user interface has become more intuitive while offering the same relevant features.

Editing Preferences

By calling View->Preferences...one can change a number of settings in ToscanaJ. These are distributed in a number of sections, accessible through tabs. These sections are:

Diagram colors: Most colors used in the diagrams can be changed. The empty nodes and the background normally do not have a color set, in the former case this means that the empty nodes have the same color as the normal ones, setting a color will result in all empty nodes having this color. In the case of the background color it means that the background is transparent – setting a color for it will cause not only the on-screen view to use this, but also printouts and diagram exports. At the moment the colors can be unset only be restoring the full default settings.

Diagram options: One can set the default label font, the gradient type and node scaling options and some sizes for diagram drawing. The default values for the latter are 20 pixels margin on each side of the diagram, a node for a non-realized concept (i.e. an empty node) is reduced to a third (i.e. divided by 3), a highlighted line is 3 times as wide as a normal line and all not-selected parts in a highlighted diagram are 70% transparent.

Line Labels: ToscanaJ can display labels on the lines, showing the ratio of the extents of the upper and lower concept. This can be turned on here, together with two related options: one to change the width of the lines according to this ratio, another groups nodes representing the same extent with one of two visual clues (note that this is always one realized concept plus a set of super concepts which are not realized). The labels will have

a number in them, using the format given or the default, which is “0.00%” -- formats are given similar to spreadsheet applications, if the format ends on a percent symbol, it will be a percentage, otherwise a normal float value. The font size can be adjusted and the two colors used can be set: the labels use a flood effect, similar to a bar chart. Same color can be set for both colors to the same to avoid this effect.

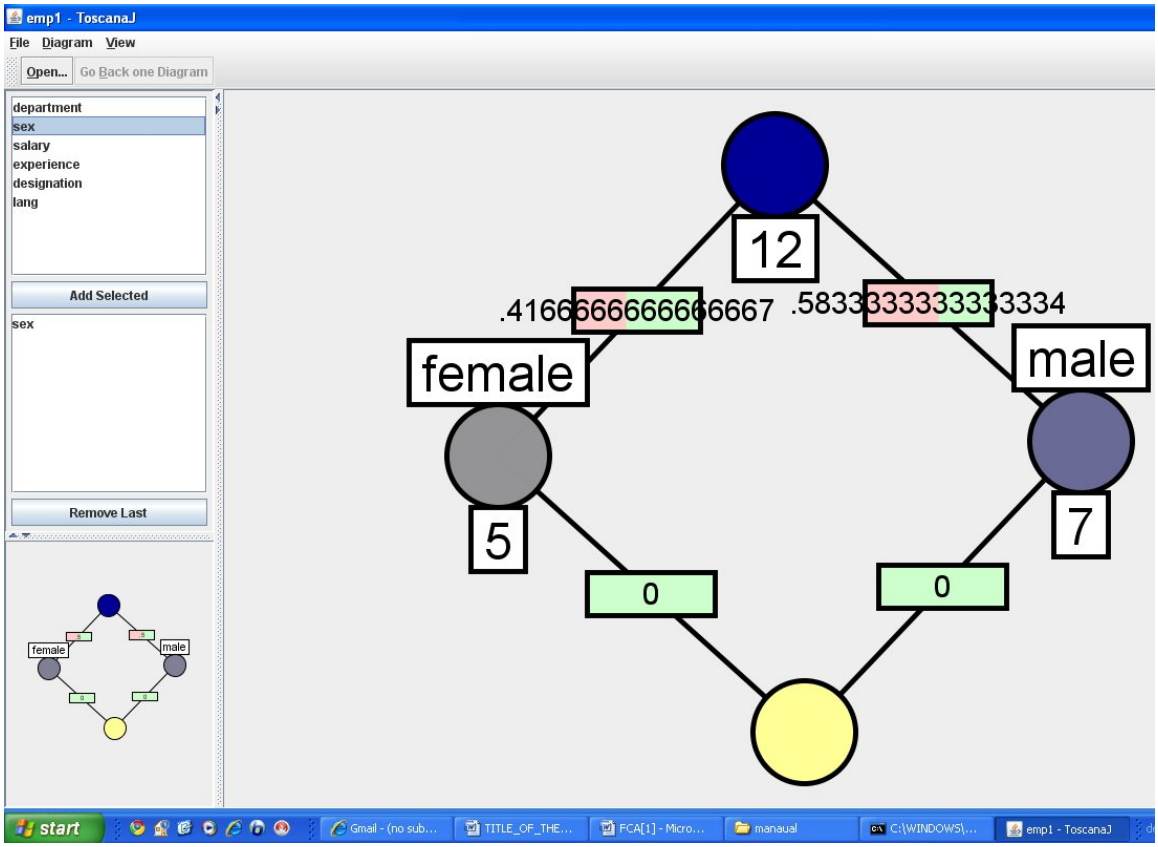


Figure 4.11: Line diagram with line label option.

CHAPTER 5
IMPLEMENTATION

Tool ToscanaJ is used to implement the FCA. As already discussed Section 4.3.2 introduction of toscanaj, Elba is used to create the conceptual information system and Toscana is used to view the diagram. Data used for clustering is given in table 5.1. SQL file is used to save the data. Srlect is used for Sr. Lect. and lect for lecturer.

Emp_name	Age	Dept.	Designation	Lang known	Salary	Exp.	Sex
DEEPAK	29	ECE	Sr. lect.	Hindi/English	25000	4	Male
PANKAJ	27	ECE	Sr. lect.	Hindi/English/Punjabi	25000	3.5	Male
SHIKHA	26	CSE	Lect.	English/Punjabi	19000	2.5	Female
MAMTA	26	CSE	Lect	Hindi/English	19000	2	Female
SANJAY	26	Mech.	Sr. lect.	English/Punjabi	22000	3	Male
SHAILJA	24	ECE	Lect.	Hindi/English/Punjabi	18000	1.7	Female
SUDESH	24	Mech.	Lect.	Hindi//English	18000	1.5	Male
KAMLESH	23	CSE	Lect.	English/Punjabi	17000	1	Female
SHALINI	32	CSE	Sr. lect.	Hindi/English/Punjabi	32000	10	Female
RAMESH	28	Mech.	Peon	Hindi/Punjabi	5000	3	Male
JAGDEEP	28	ECE	Peon	Hindi//English	4500	2.8	Male
GAGANDEEP	27	CSE	Peon	Hindi/Punjabi	4500	2.5	Male

Table 5.1 Employee table used for clustering.

5.1 Starting Elba

Steps for creating conceptual information system using Elba

1. Elba always opens the last file that was used, to create a new file, so choose “File -> New”.
2. First step is to choose the database type from available options as shown in Figure 5.1

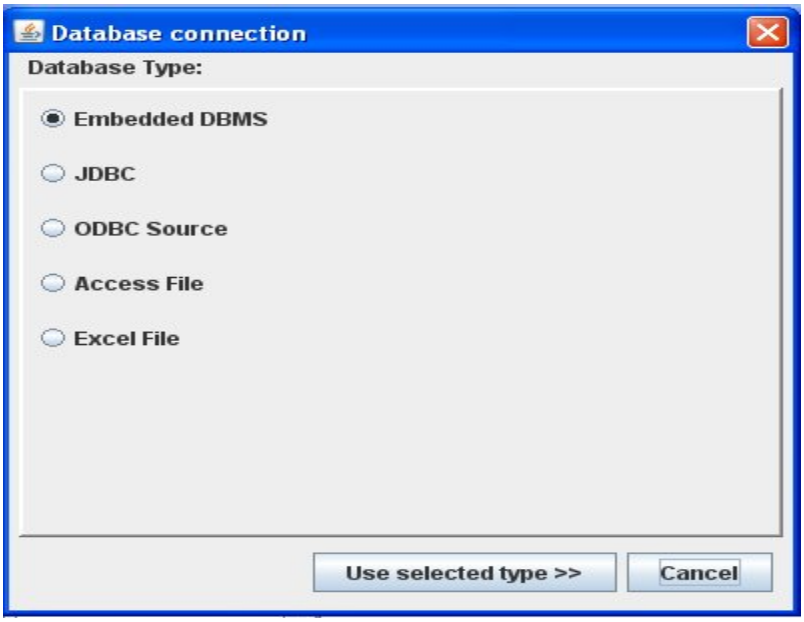


Figure 5.1: Database connections.

3. Next file of selected database is connected with Elba new project.

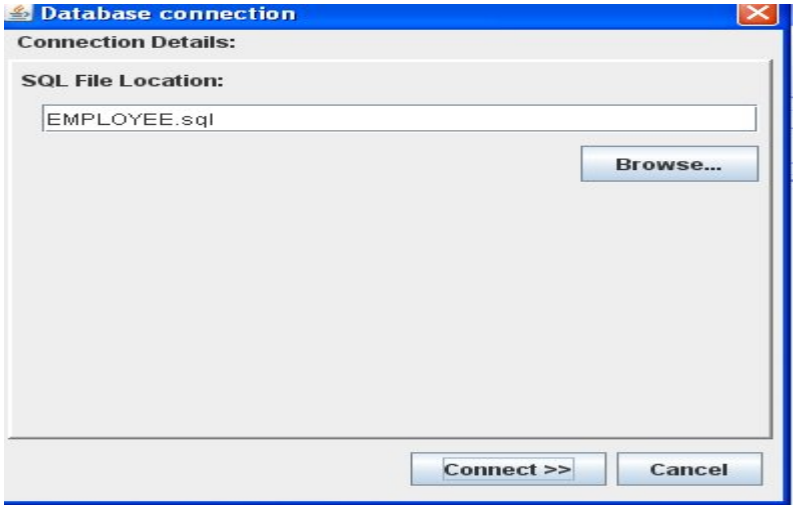


Figure 5.2: Connect database file.

4. Context Table can be created from newdiagram-context Table. Table for sex attribute is in Figure 5.3 and corresponding conceptual diagram is in Figure 5.4.

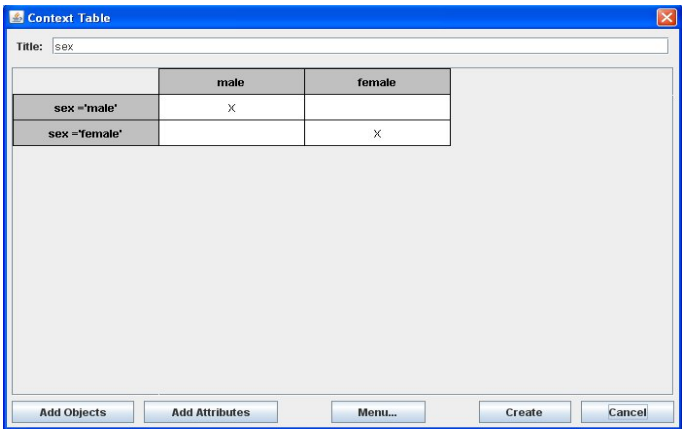


Figure 5.3: Context Table for sex attribute.

5.2 Line Diagram in ToscanaJ

Conceptual diagram can be viewed through Toscana. Diagram created in Elba cannot be edited in ToscanaJ. There are different options available to view the diagram. In Figure 5.4 objects that are male are 5, and female are 7. Total number of objects is 12.

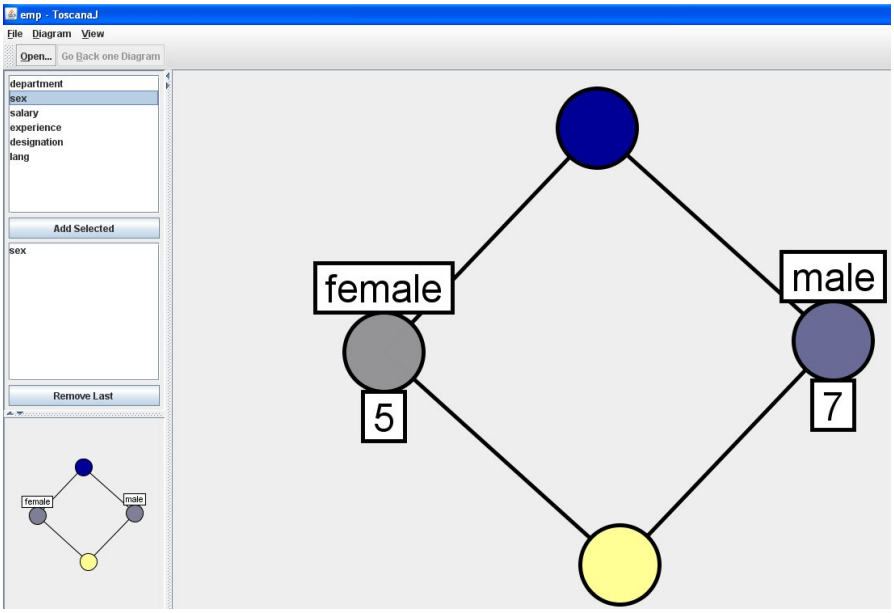


Figure 5.4: Line diagram for sex attribute.

In ToscanaJ there are three options available for displaying the extents or object contingents either:

- Count: the number of objects in the set; as in Figure 5.4

- List: a textual representation (usually an identifier) of the items in the set; as in Figure 5.6
- Distribution of Objects: the percentual distribution with respect to the current object set in the diagram; as in Figure 5.8.

In toscanaj option is there to display the exact number of matches as in Figure 5.4 and all matches as in Figure 5.10.

5.3 Clustering Based On the Different Attributes

5.3.1 Clustering on the attribute langknown

Context table for Lang known attribute in Figure 5.5. In this table there are 4 type of objects langknown= ‘h,e,p’, ‘h,e’, ‘e,p’, ‘h,p’. Three attributes Hindi, English Punjabi. ‘h,e,p’ are the objects who know the language Hindi, English, Punjabi, similarly for rest of three objects as in context table in Figure 5.5.

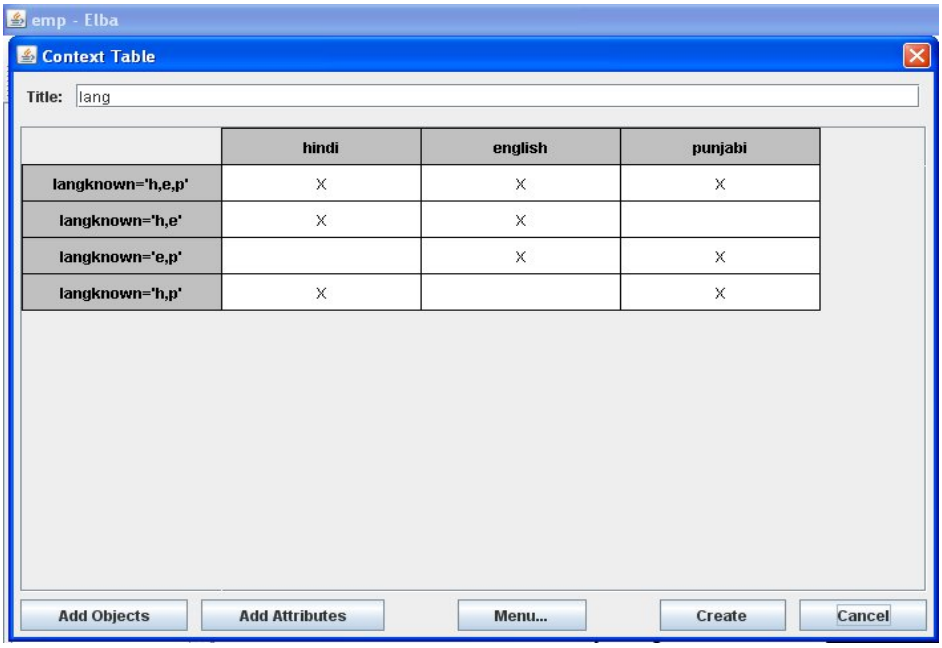


Figure 5.5: Context table for langknown attribute.

Conceptual diagram corresponding to context Table of langknown is in Figure 5.6. In this line diagram list of objects is shown as compare to number of objects as in Figure 5.4.

There are three objects PANKAJ, SHAILJA, SHALINI who know the Hindi, English, and Punjabi.

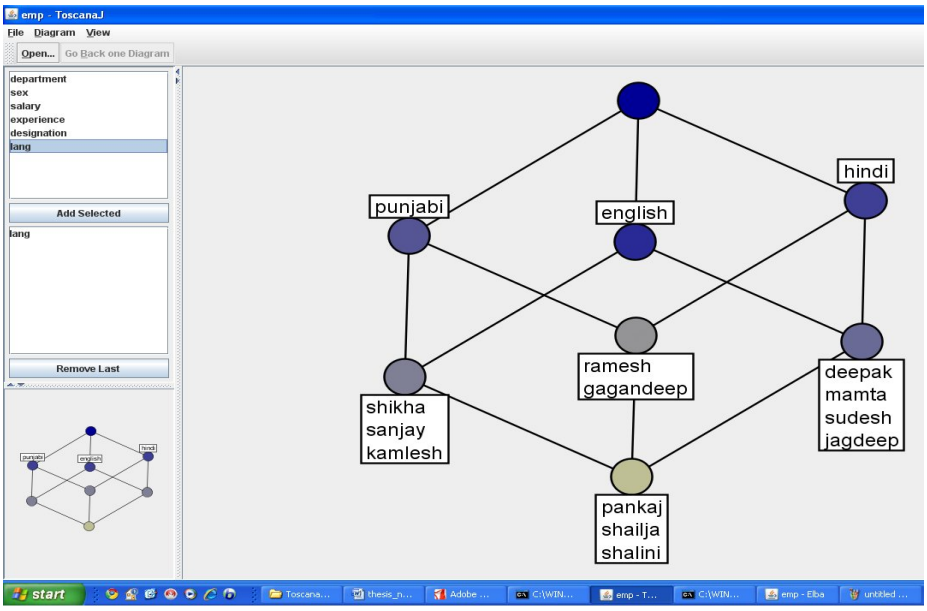


Figure 5.6: Line diagram for langknown attribute.

5.3.2 Clustering on the attribute department

Context Table for attribute department is in Figure 5.7. Three types of objects in this context table that is person belong to any of three departments CSE, ECE, and Mech.

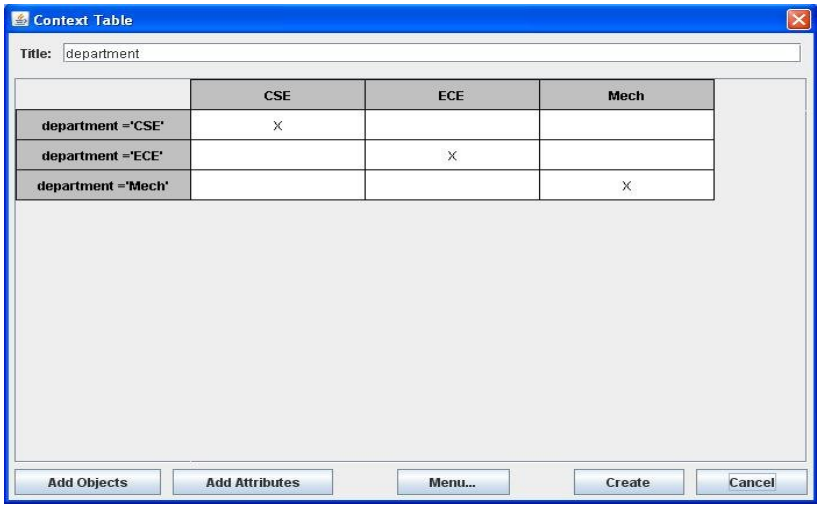


Figure 5.7: Context table for Department attribute.

Line diagram for context Table of sex is in Figure 5.8. In this line diagram percentage distribution of objects is shown as compare to number of objects in Figure 5.5. Three

clusters are there one of CSE persons with 41.67% and second Mech with 25% and third is of ECE with 33.33%.

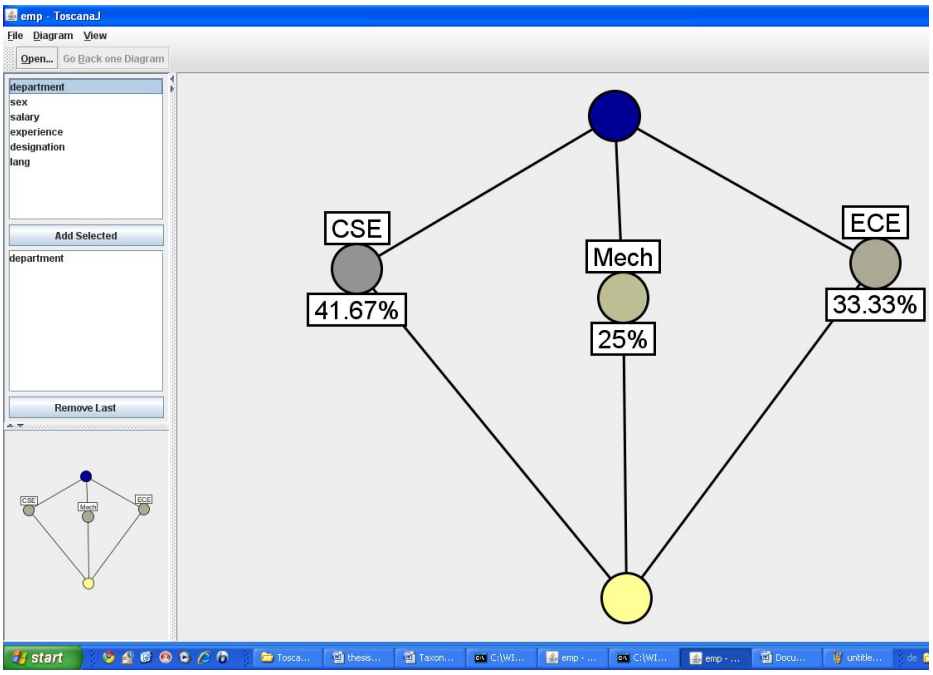


Figure 5.8: Line diagram for department attribute.

5.3.3 Clustering after adding the attribute designation to cluster department

Clustering with respect attribute designation divide the data into three clusters one with designation lecturer, second with senior lecturer and third with designation peon.

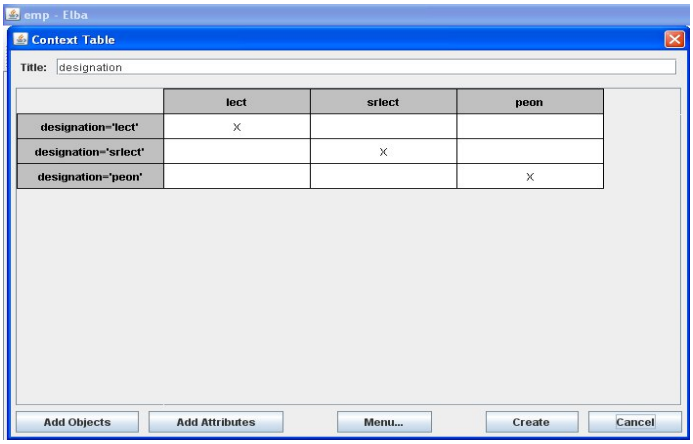


Figure 5.9: Context table for Designation attribute.

Line diagram for context table designation. In this diagram top node has extant that is number of objects 12 it shows the all matches as contrast to only exact matches in figure

5.4,5.6,5.8. Total numbers of objects in sample data are 12 out of these 4 are senior lecturer, 5 are lecturer and 3 are peon.

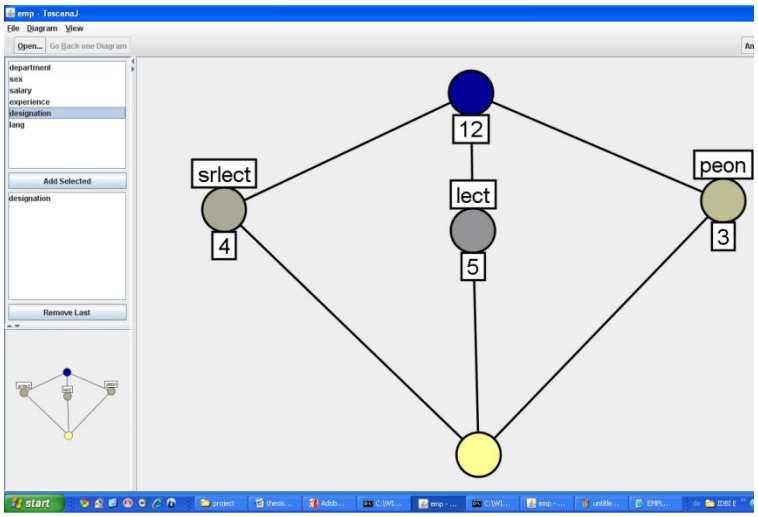


Figure 5.10: Line diagram for department attribute.

On combining both department and designation following clusters are there in line diagram- Figure 5.11 corresponding to context Table 5.2

	CSE	ECE	Mech	Lect	Srlect	Peon
Dept=CSE and designation=lect	X			X		
Dept=ECE and designation=lect		X		X		
Dept=Mech and designation=lect			X	X		
Dept=CSE and designation=srlect	X				X	
Dept=ECE and designation=srlect		X			X	
Dept=Mech and designation=srlect			X		X	
Dept=CSE and designation=peon	X					X
Dept=ECE and designation=peon		X				X
Dept=Mech and designation=peon			X			X

Table 5.2: Context Table for Department and Designation.

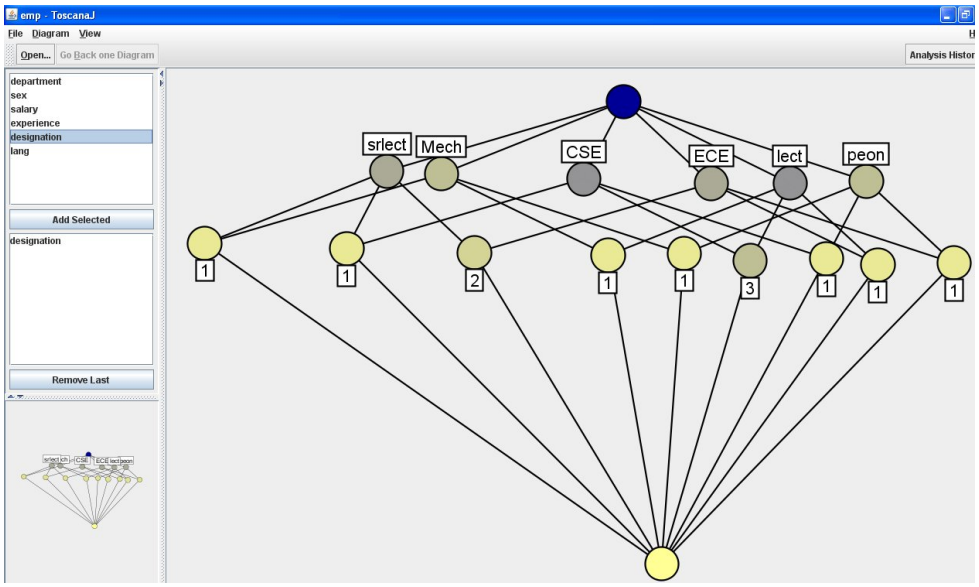


Figure 5.11: Line Diagram of Context Table 5.2.

5.3.4 Nesting line diagram

ToscanaJ allows *nesting* one diagram (the *inner diagram*) into another diagram (the *outer diagram*). Figure 5.12 shows designation nested into sex line diagram. Each of the inner nodes of this diagram represents a combination of attributes from the outer and the inner diagram.

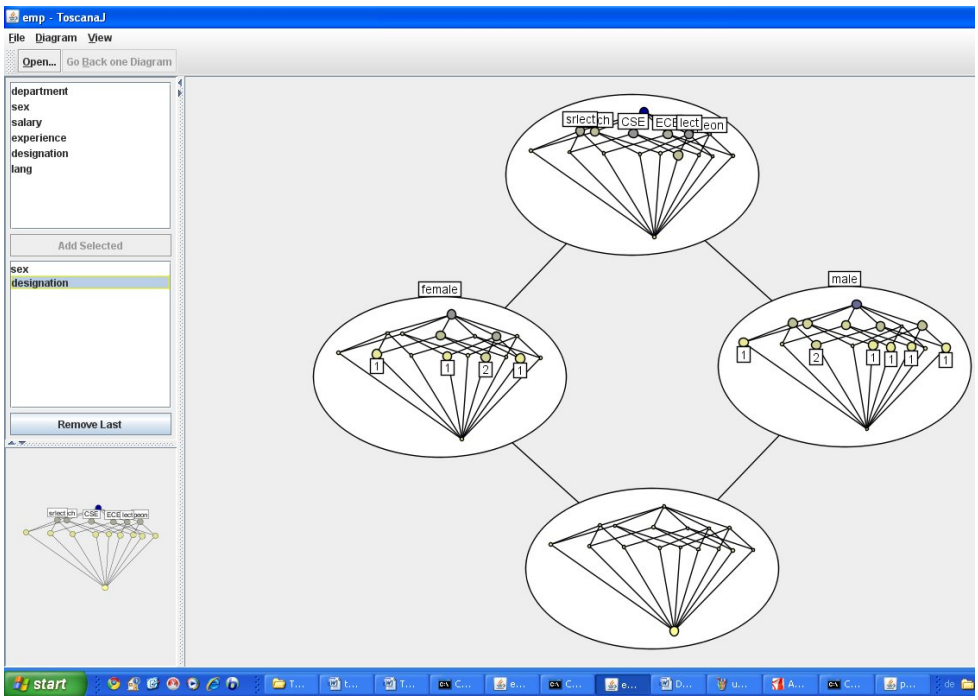


Figure 5.12: Designation diagram nested into sex diagram.

To read this type of diagram you have to follow the inner and the outer edges. Each node in an inner diagram is considered connected to the same node in nodes above in the outer diagram. The result of clicking on the left most node is shown in Figure5. 13. One can see that there is one female who is srlect and belong to CSE department. Nesting is quite helpful to see how the attributes of one diagram affect the occurrence of attributes of another diagram.

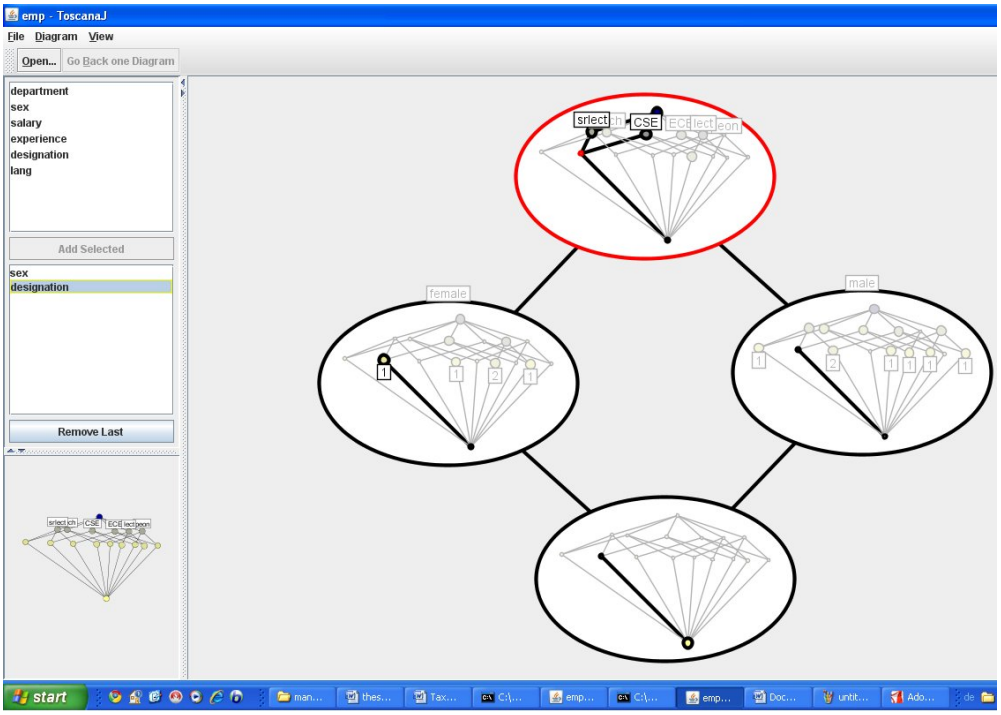


Figure 5.13: Highlighted Nested Diagram.

5.3.5 Clustering after adding the attribute salary

Clustering after adding attribute salary to the context Table of department and designation in Table 5.2 is shown in Figure 5.14. New attribute salary is added for making clustering more specific. If salary>15000 and <=20000 then object is lecturer, for senior lecturer salary is >20000 and for peon it is in between 3000 and 5000. After clustering there are nine types of clusters. For example there are three objects who belong to CSE department and their designation is lecturer and salary is between 15000 and 20000. Six intermediate nodes are shown which specifie the attributes.

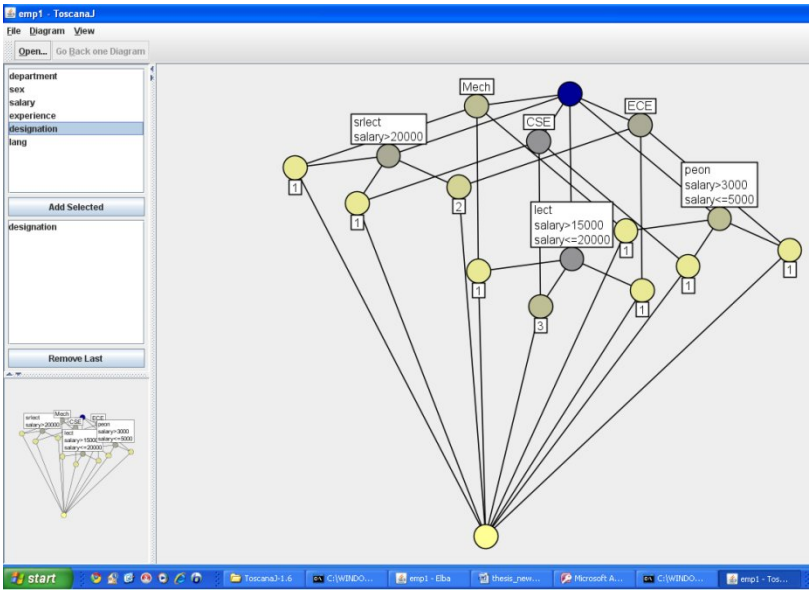


Figure 5.14: Line diagram after combining salary attribute with diagram 5.11.

Figure 5.15 shows nesting of line diagram in Figure 5.14 in sex line diagram. Highlighted lines specify that one female object is there who is lecturer in ECE department and salary >15000 and salary<=20000.

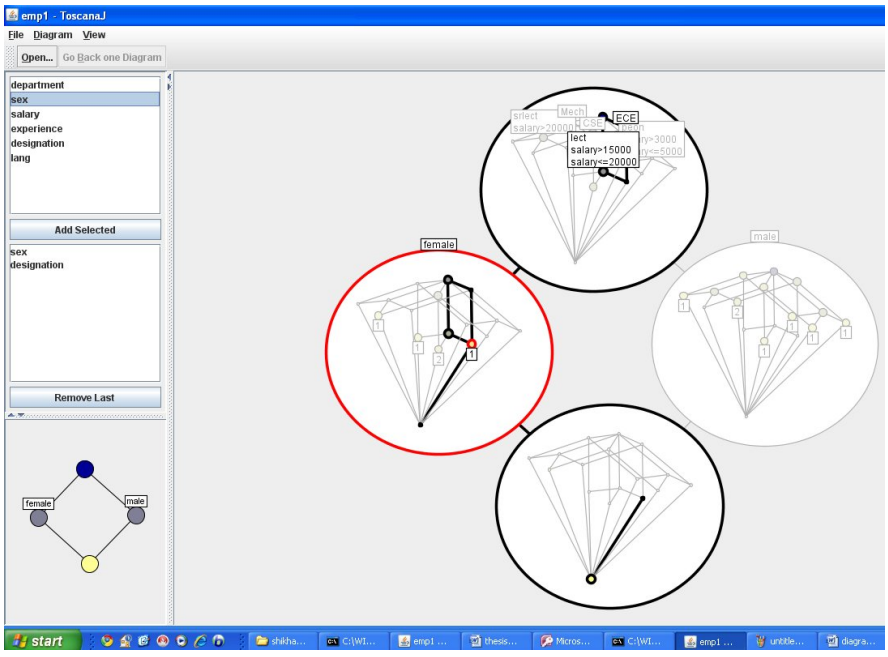


Figure 5.15: Highlighted line diagram 5.14 nested in Sex Diagram.

5.3.6 Clustering on the attribute experience

Clustering on the basis of experience is similar to salary. Table 5.3 is context table for experience attribute and corresponding line diagram is shown in Figure 5.16.

Exp	>=1	>=2	>=3	>=4	>=10	<1	<2	<3	<4	<10
Exp>=10	X	X	X	X	X					
Exp>=1 and <2	X						X	X	X	X
Exp>=2 and <3	X	X						X	X	X
Exp>3 and <4	X	X	X						X	X
Exp>4 and <10	X	X	X	X						x
Exp<1						X	X	X	X	X

Table 5.3 Context Table for Experience.

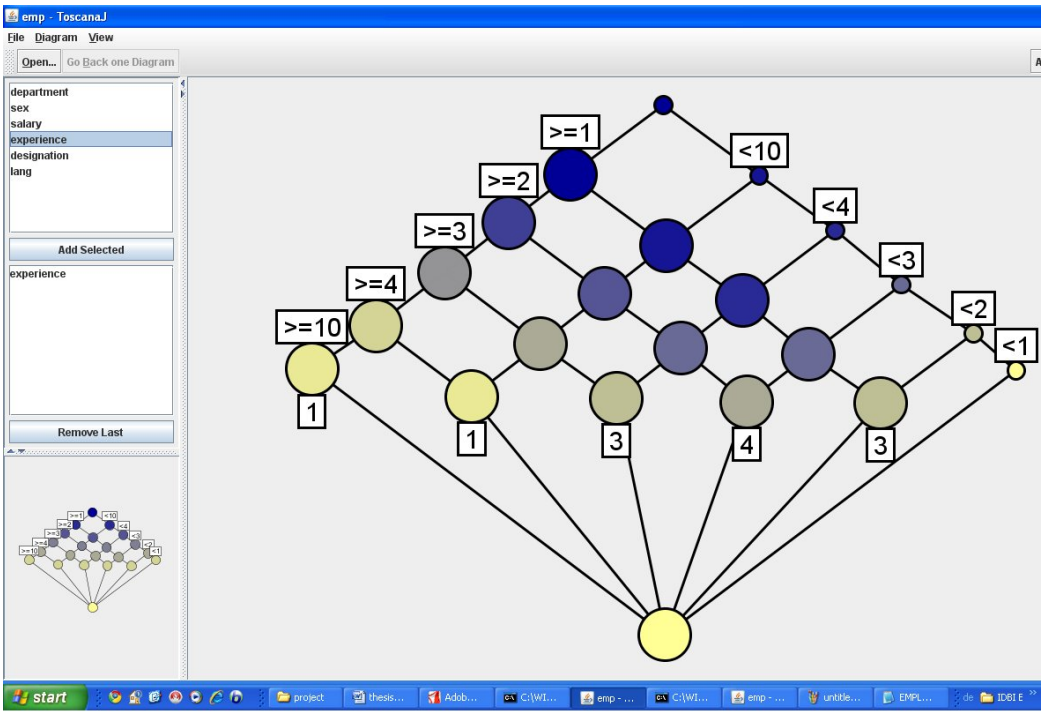


Figure 5.16: Line Diagram For experience attributes.

CHAPTER 6

CONCLUSION AND FUTURE SCOPE OF WORK

Conclusion

Statistical forecasting finds keywords and then calculates the frequency and distance of these keywords. Statistical forecasting tools include many techniques for predictive forecasting, most often using inference theory. The frequency and distribution of words has some general value in understanding content. But, cannot understand the meaning of words or sentences; or provide context. These tools are still limited by keyword constraints; and can only infer simplistic meaning from the frequency and distribution of words. As we are preparing to shift from syntactic Web to semantic Web it is necessary to have a through conceptual analysis of every document and categorize it accordingly.

The major problem we foresee in implementation FCA on XML and HTML pages is that in implementations of FCA the base of data in tabular format, but all the data on web is not stored in the form of tables, instead it is generally in XML files.

Although we have not implemented the ToscanaJ tool used for FCA directly to the Web pages or documents we have tried to implement it on a simple database table and we conclude that by implementing FCA on ToscanaJ, clusters are available in the form of concept lattice rather than a simple hierarchy or flat partition as in other clustering techniques which can definitely serve as a base for conceptual clustering of semantic related attributes.

Future Scope of Work

Semantic Web architecture is the automated conversion and storage of unstructured text sources in a semantic web database.
The ToscanaJ project is an ongoing effort and many additional features are planned. This includes more features for database connectivity, more complete editors, new editing workflows and better interoperability with other FCA tools (e. g. ConExp or Galicia) as well as standard data formats, such as CSV and similar formats or generic XML.

FCA can serve as a base for implementing Semantic Web as Semantic Web applications automatically extract and process the concepts and context in the database in a range of highly flexible tools.

REFERNCES

[1] Joshua Zhexue Huang & Michael Ng. & Liping Jing, "Text Clustering: Algorithms, Semantics and Systems", The University of Hong Kong , Hong Kong Baptist University April 9, 2006 PAKDD06 Tutorial Singapore

[2] 2 CIS 526: Machine Learning Lecture 1 (Sep 02, 2003) http://www.ist.temple.edu/~vucetic/cis526fall2003/lecture1.pdf

[3] Brian T. Luke: "K-Means Clustering" http://home.dei.polimi.it/matteucc/Clustering/tutorial_html/kmeans.html

[4] Karl Erich Wolff, "A First Course In Formal Concept Analysis How To Understand Line Diagrams", In: Faulbaum, F. (ed.) SoftStat'93 Advances in Statistical Software 4, 429-438.

[5] S. C. Johnson (1967): "Hierarchical Clustering Schemes" *Psychometrika*, 2:241-254

[6] Peter Becker1 and Joachim Hereth Correia, "The ToscanaJ Suite for Implementing Conceptual Information Systems",

[7] Elba User Manual *Bastian Wormuth* Version 0.1

[8] Joshua Zhexue Huang & Michael Ng. & Liping Jing, "Text Clustering: Algorithms, Semantics and Systems", April 9, 2006 PAKDD06 Tutorial Singapore

[9] Algirdas Laukaitis, Olegas Vasilecas, "Formal concept analysis and information systems modeling", International Conference on Computer Systems and Technologies - *CompSysTech'07.*

[10] Frank Keller, "Introduction to Machine Learning Connectionist and Statistical Language Processing",.

[11] Formal Concept Analysis in Information Science.

[12] Thesis Submitted by Gordon Kwok Tung Tam, "FOCAS - Formal Concept Analysis and Text Similarity By".

[13] Cluster analysis, http://statsoft.eu/uk/textbook/stcluan.html

[14] Cluster analysis: basic concepts and algorithm, http://www-users.cs.umn.edu/~kumar/dmbook/ch8.pdf

[15] Vladimir Estivill-Castro, "Why so many clustering algorithms: a position paper", ACM, NY, USA

[16] Clustering methods, http://www.cis.hut.fi/sami/thesis/node9.html

[17] http://www.aifb.uni-karlsruhe.de/WBS/gst/FBA03/3_conceptual_scaling_en.pdf

[18] Juan M. Cigarran, Anselmo Penas, Julio Gonzalo, and Felisa Verdejo, "Automatic Selection of Noun Phrases as Document Descriptors in an FCA-Based Information Retrieval System".

[19] Hierarchical clustering, http://www.changbioscience.com/res/res/rHierarchicalscluster.htm

[20] Clustering by mike chapel, http://www.databases.about.com/od/datamining/g/clustering.htm

LIST OF PUBLICATION

Papers Published

Shikha and Shalini Batra, "Implementation of Formal Concept Analysis using ToscanaJ", published in National Journal "PIMT Journal of Research, GOBINDGGARH" Vol. 2, No. 1, March-August-2009 (page number 73-76).

Papers Communicated

Shikha and Shalini Batra, "Using FCA for Implementing Semantic Web" CiiT International Journal of Data Mining and knowledge Engineering. ISSN 0974-9675, http://ciitresearch.org/