Analysis Guided Visual Exploration of Multivariate Data

by

Di Yang

A Thesis

Submitted to the Faculty

of the

WORCESTER POLYTECHNIC INSTITUTE

In partial fulfillment of the requirements for the

Degree of Master of Science

in

Computer Science

by

April 2007

APPROVED:

Professor Elke A. Rundensteiner, Thesis Advisor

Professor Matthew O. Ward, Thesis Advisor

Professor Carolina Ruiz, Thesis Reader

Professor Michael Gennert, Head of Department

Abstract

Visualization systems traditionally focus on building graphical depictions of relationships among information in a human comprehensible format. They tend not to provide integrated analytical services that could aid users in tackling complex knowledge discovery tasks. Users' exploration in such traditional visualization environments is usually impeded due to several problems: 1) Valuable information is hard to discover, when too much data is visualized on the screen. 2) They have to manage and organize their discoveries off line, due to the lack of any systematic discovery management mechanism provided as part of visualization system. 3) Their discoveries based on visual exploration alone may lack accuracy, because perceptual power of human beings is subjective and may be insensitive to some of the characteristics of the information. 4) They have no convenient access to the important knowledge learned by other users. To tackle these problems and provide improved exploration-support service, it has been recognized that analytical tools must be introduced into visualization systems.

In this thesis, we present a novel analysis-guided exploration system, called the Nugget Management System (NMS) that aims to tackle the these shortcomings. NMS leverages the collaborative effort of human comprehensibility and machine computations to facilitate users' visual exploration process. Specifically, NMS first extracts the valuable information (nuggets) hidden in datasets based on the interests of users. Given that similar nuggets may be re-discovered by different users, NMS consolidates the nugget candidate set by clustering based on their semantic similarity. To solve the problem of inaccurate discoveries, data mining techniques are applied to refine the nuggets to best represent the patterns existing in datasets. Lastly, the resulting well-organized nugget pool is used to guide users' exploration. Among the five stages of NMS framework, we pay our main attention on solving the technical challenges existed in nugget combination and refinement. A critical issue that makes nugget combination difficult is the distance metrics

between nugget (how can we know whether two nuggets are similar or not). For nugget refinement, trying to understand what a user is looking for when a nugget was generated is a difficult job which requires effective ”match” heuristics. In this thesis, we present solutions to both of these two challenges, and we have conducted user study to carefully compare the performances of different distance metrics between nuggets. Thus, besides presenting the general framework of NMS, the contributions of this thesis also include the novel solutions to nugget combination and refinement.

To evaluate the effectiveness of NMS, we integrated NMS into XmdvTool, a freeware multivariate visualization system that had not offered analytical services. User studies were performed to compare the users' efficiency and accuracy of finishing tasks on real datasets, with and without the help of NMS. Our preliminary evaluations indicate that NMS may greatly improve users time efficiency and accuracy when solving knowledge discovery tasks and NMS works in a stable manner during explorations by a sequence of users.

Acknowledgements

I would like to take this opportunity to thank my whole family who has supported and encouraged me forever, especially, my parents Zhiming Yang and Caiyun Wang, and my dearest grandmother Qinglian Yu.

I also would like to express my greatest gratitude to my advisors, Prof. Elke A. Rundensteiner ,Prof. Matthew O. Ward and Prof. Murali Mani and my thesis reader, Prof. Carolina Ruiz for their guidance and invaluable contributions to this work.

I thank Jing Yang, Qingguang Cui, Zaixian Xie, Charudatta Wad, Do Quyen Nguyen, and other members of the XMDV team at WPI for the development of XMDVtool, which formed the testbed for this work. I also thank members of DSRG at WPI for listening to talks and providing input on this work.

I thank to the research assistant support from NSF grant IIS-0119276.

Contents

List of Figures

Chapter 1

Introduction

Visualization systems traditionally focus on building graphical depictions of relationships among information in a human comprehensible format. By doing so, they aim to help their users to better understand the information. This means the users can either learn some facts that are not easy to discover without the graphical depiction, or the users' knowledge to some facts can become deeper or more precise. The usefulness of visualization systems has been well established [64, 55, 56, 23].

Recently, visual analytics [61] has been employed to solve complex knowledge discovery tasks in many important fields of human society, ranging from homeland security, credit fraud detection to financial market analysis. Solving such tasks usually requires analysts to perform complicated and iterative sense-making processes [21, 51]. Thus, it has been recognized that relying on analysts' perceptual power alone to conduct visual exploration may not always be the most effective method to solve these problems.

To fully support visual analytics, visualization systems have to be improved by tackling some key challenges. **1) Overloaded Displays:** When too much information is visualized on the screen, effective knowledge discovery is difficult. For example, as shown in Figure 1.1, when a dataset, even with modest numbers of records and dimensions, is visu-

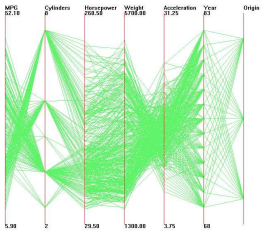

Figure 1.1: "Cars" dataset visualized with Parallel Coordinates

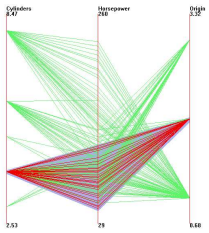

Figure 1.2: A complete cluster among three dimensions of "Cars"

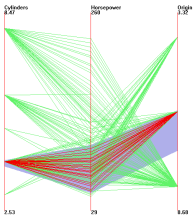

Figure 1.3: One "partial cluster" found by users

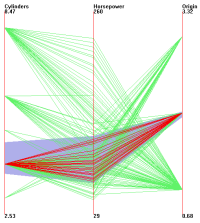

Figure 1.4: Another similar yet not identical "partial cluster"

alized, overloaded displays make knowledge discovery, such as pattern detection, a time-consuming process. **2) Disorganized Discoveries:** Since to date there is no systematical discovery management mechanism provided by visualization systems, users have to manage and organize their discoveries off line on their own. For example, some users, either due to rich domain knowledge or after a long time of exploration, may be able to identify some patterns (e.g., the cluster highlighted in red in Figure 1.2). But unfortunately, she will not be able to store it in the system nor to easily retrieve it for future exploration. Even if the systems provide some simple recording functionality, since a pattern may be repeatedly visited by a single user or even multiple users, redundant recordings may be generated (e.g., the clusters in Figures 1.3 and 1.4 are very similar). Such redundancy causes information overload that may hinder the future use of those recordings. **3) Inaccurate Discoveries:** Discoveries found by using their perceptual power alone may be inaccurate, because perceptual power of human beings is subjective and may be insensitive to some of the characteristics of the information. For example, the "clusters" found by users in Figures 1.3 and 1.4 are actually subparts of a complete cluster depicted in

Figure 1.2. Such inaccurate discoveries may lead to low-quality decision making (i.e., this user may miscount the population of the whole cluster, if she works on the "partial cluster" in Figure 1.3). **4) Isolated Knowledge:** Even if valuable knowledge may have already been uncovered, there is no convenient mechanism for users to access and share it, not to mention conduct collaborative visual analytics. For example, a user interested in "clusters" in the dataset may spend a lot of time to find the one mentioned in Figure 1.2, although it may have already been previously discovered by other users.

Previous efforts to tackle some of these problems can be roughly classified into two categories. 1) User-driven: In this category, while the knowledge discovery process still relies on users' perceptual power, a variety of visual interaction mechanisms, such as zooming, filtering, color coding and dynamic querying, are offered by the visualization systems to facilitate exploration [4, 64]. These techniques mainly help to relieve overloaded display. Our framework applies these techniques to allow users to best use their perceptual power during visual exploration. 2) Data-driven: Data-driven techniques aim to expediate knowledge discovery with the help of the analytical power of machines. Data mining algorithms [30, 69, 38], which detect useful patterns or rules in large datasets, fulfill an important role here. These techniques will be employed in our framework to improve the accuracy of discoveries.

More recently, some initial efforts have emerged to take advantage of both human perceptual abilities and computational power of computers to deal with the challenging process of knowledge discovery [61]. Visual data mining (VDM) [22, 34] involves users in the mining process itself, rather than being carried out completely by machines. In VDM, visualizations are utilized to support a specific mining task or display the results of a mining algorithm, such as association rule mining, and thus enhance user comprehension of the results. However, considering the single-task-driven style of VDM (e.g., detecting all the patterns of a certain type existing in a dataset), it usually does not support

an iterative and comprehensive sense-making process. [21] proposed interactive tools to manage both the existing information and the synthesis of new analytic knowledge for sense- making in visualization systems. This work so far has not paid much attention on how to consolidate the users' discoveries. Collaborative visual analytics [51] introduced computational power into the sense-making process with focus on supporting the exchange of information among team members.

In this work, we design, implement and evaluate a novel analysis-guided exploration system, called the Nuggets Management System (NMS), which leverages the collaborative effort of human intuition and computational analysis to facilitate the process of visual analytics. Specifically, NMS first extracts nuggets based on both the explicit and implicit indication of users' interest. To eliminate possible redundancy among the collected nuggets, NMS combines similar nuggets by conducting nugget clustering. Then, data mining techniques are applied to refine the nuggets and thus improve their accuracy in capturing patterns present in the datasets. We also provide a rich set of functionalities to manage the nuggets. With them, nuggets can be maintained automatically (i.e., out-of-date nuggets can be pruned by the system) or by the users (i.e., users can attach annotations [42] to nuggets to facilitate nugget retrieval and sharing). Lastly, the well-organized nugget pool will be used to guide users' visual exploration in both user- and system-initiated manners.

As a general framework for analysis-guided exploration of multivariate data, NMS can be incorporated into any multivariate visualization system. To verify the feasibility of NMS, we have integrated it into XmdvTool [64], a freeware tool developed at WPI for visual exploration and analysis of multivariate data sets. The main contributions of this thesis are:

- We introduce a novel framework of analysis-guided visual exploration, which facilitates visual analytics of multivariate data.

- We present a nugget combination solution that effectively reduces the potential redundancy among nuggets. We design a novel distance metric which effectively capture the distances between nuggets, and our user study shows that it matches well with users' intuition.

- We present a nugget refinement solution, which utilizes data analysis techniques to improve accuracy of the nuggets in capturing patterns in datasets. This is a novel approach that leverages the advantages of both human intuition and computational analysis. It not only improves the accuracy of users discoveries, but also avoids expensive global data mining process.

- We develop tools for the management and support of visual exploration based on a learned nugget pool.

- We apply the above techniques of NMS to XmdvTool, a freeware multivariate data visualization tool.

- We describe user studies evaluating the effectiveness of NMS. The user study demonstrates that NMS is able to enhance both the efficiency and accuracy of knowledge discovery tasks.

The remainder of this thesis is organized as follow: In Chapter 2, we will introduce the overall NMS framework for analysis guided visual exploration, including nugget extraction, combination, refinement, maintenance and nugget-guided exploration. In the later chapters, we will carefully discuss the technical details and experimental results of nugget combination and refinement. Specifically, in Chapter 3, we will discuss the distance metrics used in nugget combination (clustering) in details. A user study comparing different distance metrics will also be described in this chapter. The specific clustering algorithm we developed for nugget clustering is presented in Chapter 4. Evaluations to

this algorithm appears in chapter as well. Chapter 5 will discuss the ideas and techniques utilized in nugget refinement. Chapter 6 shows our user study assessing the overall functionality of NMS. In Chapter 7, we will introduce the related work. Finally, in Chapter 8, we draw conclusions and envision our future work.

Chapter 2

NMS Framework for Analysis Guided Visual Exploration

In this chapter, we introduce the overall NMS framework for analysis guided visual exploration, including an overview of its different components and brief introductions to key ideas used in each component. The specific components are: nugget extraction, combination, refinement, maintenance and nugget-guided exploration.

2.1 Nugget Extraction

2.1.1 Definition of Nuggets

Before introducing nugget extraction techniques, we define our notion of what we mean by the term nuggets. Generally, a nugget is some piece of valuable information extracted from the dataset, typically, some subpart of the whole dataset. A nugget could be a representative of clusters, outliers, association or any other type of patterns. Additional attributes of a nugget, such as a name and annotations, can be attached to a nugget as well. For the purpose of this thesis, a nugget N is defined by a range query Q over a

particular dataset D as well as the result of this query, dataset $Q(D)$. In N-dimension space, a nugget is a (hyper-)rectangle, whose boundaries are decided the query range on each dimension. Why we choose range query as the initial type of nugget we study on is because 1) Range query is a very common query type that allows users to specifiy the subpart interested to them in a dataset. 2) An important interaction tool existing in current visualization system, which is called "brush", use the semantic of range queries. A more extensive range of nugget types will be considered in our future work.

$N = \{D, Q, Q(D)\}$; $Q =$ Select $D.A_m, D.A_l, ..., D.A_n$ From D Where $D.A_m = [A_m.b_l : A_m.b_h], D.A_l = [A_l.b_l : A_l.b_h], ..., D.A_n = [A_n.b_l : A_n.b_h]$; $\{D.A_m, D.A_l, ..., D.A_n\} \subseteq$ attributes of D, $A_x.b_l$ and $A_x.b_h$ are the lower bound and the upper bound of the query ranges on attribute A_x, $[A_x.b_l : A_x.b_h]$ means "from $A_x.b_l$ to $A_x.b_h$"; $Q(D) \subseteq D$.

As shown in the definition above, for a given dataset, a nugget is first depicted by a query that has a selective range on some (or all) of the dimensions. However, a nugget is not only defined by the range query itself, but also the results (data records) of this query.

The concept of nuggets is independent of the display methods in multivariate visualization systems, such as Parallel Coordinates, Scatterplots and Glyphs [64]. Without loss of generality, we use Parallel Coordinates [32], which is a widely used method, to demonstrate the examples in this paper. Thus visually a nugget appears as a blue band across the axes, which represents the query ranges on each dimension, and the red (highlighted) lines that indicate the selected records (result) of the query. As shown in Figures 1.2, 1.3 and 1.4, users can specify different queries by adjusting the lower and upper bounds of the blue band (selection ranges). Users can also hide some dimensions if they are not interested in them.

2.1.2 Nugget Extraction Based on User Interest

Meta-information extraction can be achieved by observing users' exploration process (user-driven) or by conducting analysis of the patterns existing in the data (data-driven). The NMS framework is compatible with the nuggets derived using either of these two methods. Data mining algorithms for pattern detection have been extensively studied in the KDD community [26, 52, 12]. Any one of these existing methods if applicable to mutlivariate datasets could be plugged into our framework.

Here in our prototype of the framework, we instead focus on nugget extraction via user-driven methods. The main benefits of user-driven methods are 1) We can bring into play the advantage of human perceptual and cognitive abilities to identify patterns in a knowledge discovery process, which is in fact one of the main reasons why people have developed visualization systems. 1) We only deal with the information that users are interested in, thus avoid unnecessary effort to produce results that users are not of interest.

Similar to other systems [21, 51], users can explicitly indicate if a particular piece of information is of interest. This is done by explicitly saving the given query and labeling it by a persistent nugget name. NMS also provides a rich set of functionalities to let users input, edit, and remove the nuggets as further discussed in Section 5. A non-intrusive alternative to explicit indication is implicit indication, a method found in intelligent systems [17, 11]. In NMS, nuggets can be extracted automatically by observing a user's exploration. "Visiting time" is one factor [17] used as the main indicator of a user's interest during visual analysis. NMS extracts a nugget if a user spends a long time visiting (querying over and looking at) a subpart of the dataset. Specifically, our system monitors users' navigation. When it finds that a user is querying over a subpart of the dataset, and spending longer than a certain amount (a threshold) of time observing this subpart, it extract a nugget based on the query the user specified. Such a extraction process can also be caused by repeatedly visitings. That means even if a subpart has never been visited for

longer than the threshold, it can also be extracted as a nugget if it is repeatedly visited, and its accumulated visiting time makes it qualified. For example, if a subpart shown in Figure 1.2 has been visited for a long time by a single user or repeatedly visited by one or more users, NMS will conclude that it is a nugget.

Problems that could be caused by such log mining, such as redundancy, inaccuracy, out-off-date nuggets, and misinterpretation of users' interests, will be tackled by nuggets combination (Chapter 2), refinement (Chapter 3) and maintenance (Chapter 4).

2.2 Nugget Combination

Relying on nuggets extraction alone may suffer from nugget redundancy. This is because as the users navigate in the datasets by moving the sliders which control the range query boundaries, rather than by explicitely specifying exact queries as typically in SQL-type query systems [44, 45], many similar nuggets with slightly different boundaries are very likely to represent the same data feature. Nugget redundancy causes two major problems to the system: 1) A large nugget pool generated during a long exploration period may make it more difficult for users to make use of individual nuggets, because searching nuggets of potential interest can be quite time-consuming. 2) Continuous growth of the nugget population may also lead to low system performance.

Here we give an example (Figure 2.1) to show that three slightly different nuggets are actually representing the same pattern in the dataset. As shown in Figure 2.1, nuggets 1, 2, and 3 are capturing a same pattern (a cluster) in the "Iris" dataset. So, an efficient method is needed to keep the nugget pool of a modest size yet with high representativeness. Several different techniques, such as sampling [10], filtering [60] and clustering [12] of nuggets may be employed to achieve this goal. After careful comparison, we chose clustering, which groups similar nuggets and generates representatives for each group.

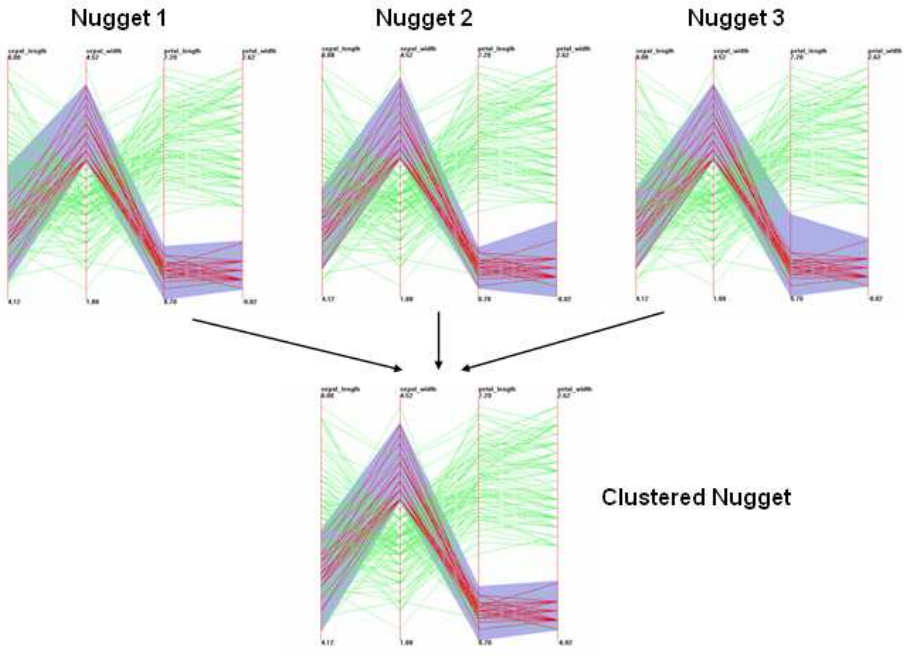

Figure 2.1: A example of clustering three similar nuggets

This is because, when constructing a representative for each group, clustering techniques consider and combine the features of all the group members, while filtering and sampling techniques tend to just pick an “important” group member as their representative. Since in many cases, we can hardly tell which nugget is surely more important than others (even if we have certain mechanism to express the importance of nuggets, nuggets with similar importance may be very common), constructing a representative which “speaks” for every nuggets in a group makes more sense than just picking one. And since we can use importance as the weight in clustering process, the representative generated will mainly reflect the feature of the dominant (supper important) nugget, if there is any. The example of forming a representative (clustered nugget) for similar nuggets is also shown in Figure 2.1.

2.2.1 Distance Metrics

Clustering aims to group objects based on their similarities [5, 6]. They require a distance measure that best expresses the domain specific similarity between objects. In our work, one of the main issues we have to tackle is thus the development of a suitable distance metrics for our multi-dimensional nuggets. To solve the problem, distance metrics are developed to effectively capture the distance between any pair of nuggets.

Query Distance

Nuggets are defined by both queries and their results. So, naturally, nuggets that are defined by similar queries should be considered to be more similar than those defined by rather different queries.

Thus our problem can be transformed into the problem of how to quantify the similarity of queries. Fortunately, previous work [65, 67] has studied this problem. The major principle utilized for measuring the query similarity (QS) can be summarized as:

$$QS(A, B) = \frac{QA \cap QB}{QA \cup QB} \qquad (1)$$

Here QS(A,B) represents the query similarity between Nugget A and Nugget B, and QA and QB are the qualifiers of these two queries. We adopt this idea as the basic principle for our query similarity measure on individual dimensions. We have also studied several important refinements to this basic idea, which make it capable to handle different types of domains (discrete, continuous, nominal) and extend it to the multi-dimensional environment. Details of this will be discussed in Chapter 3.

When we've successfully acquired normalized query similarities (between 0-1), we

can now easily calculate the query distances (QD) as shown in Formula 2.0.

$$QD(A, B) = 1 - QS(A, B) \qquad (2)$$

Data Distance

Our query distance metric effectively measures the similarity between two nuggets based on their query specification. However, nuggets are not only characterized by their queries (profile), but also by the results of the queries obtained when applying the queries to a particular dataset (content). As shown in Figures 2.2 and 2.3, two nuggets generated by

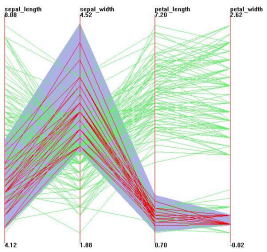

Figure 2.2: A nugget capturing a cluster in the "Iris" dataset

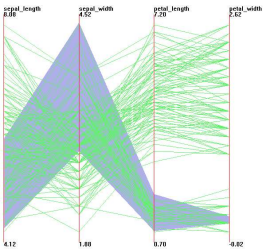

Figure 2.3: A nugget with no data record included

very similar queries may be rather different in terms of actual data content. The former contains a cluster, while the latter is empty. Clearly, we need to enhance the capability of our distance metrics by comparing the "contents" of the nuggets. Now, the problem we must solve can be viewed as a general date analysis problem. That is, given two subsets of a multi-dimensional dataset, how could we measure the distance between them. Previous works to tackle such problems [8, 20, 48, 19] can be generally classified into two main categories, statistic and transform-cost approaches. In this thesis, we introduce our proposed algorithm, Exact Transformation Measure(ETM), which is based on extending a basic transform cost algorithm. The details of this algorithm and comparisons of different data distance metrics will be discussed in Chapter 3. ETM will help us to get the Data Distance $(DD[X, Y])$, between two nuggets, where X and Y are two nuggets and A and

B are the datasets contained by them respectively.

Finally, we combine the Query Distance $(QD[X, Y])$ and Data Distance $(DD[X, Y])$ to present the Nugget Distance $(ND[X, Y])$ between any pair of nuggets X and Y.

$$ND[X, Y] = \alpha \cdot QD[X, Y] + \beta \cdot DD[X, Y] \qquad (\alpha + \beta = 1) \quad (3.0)$$

Since QD and DD are both normalized (between 0 to 1), ND will be normalized as well.

2.2.2 Nugget Clustering

Once we have learned the distances between nuggets, any generic clustering algorithm [12, 68] can be applied to conduct nugget clustering. In our system, we designed an iterative incremental clustering (IIC) algorithm, which provides real time clustering service to the nugget pool. We will discuss the details of this algorithm in Section Chapter 4.

2.3 Nugget Refinement

2.3.1 Benefit from Nugget Refinement

Data mining techniques applied to the datasets provides us further opportunities to improve the quality of nuggets. In this section, we'll introduce our solution of using data mining techniques to refine the nuggets found from users log. Such a refinement can be performed when a nugget is made because users were searching for some identifiable pattern types, such as clusters or outliers. For example, a user was searching for a cluster in the dataset, however, for some reason, she missed part of it (Figure 5.1). Then, NMS will refine the nugget to capture the complete cluster (Figure 5.2).

Nugget refinement offers several advantages over both pure log analysis or mining techniques of the data itself. They are: 1) Log analysis techniques, for example, the

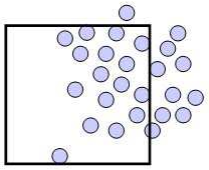

Figure 2.4: A nugget which captures the main body of a cluster bus misses part of it

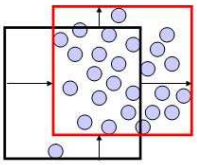

Figure 2.5: The refined nugget which captures the complete cluster

nugget extraction introduced in Section 1 of Chapter 2, rely on users' actions only, without any help from computational analysis of the datasets and their properties. Thus they may lack accuracy in nugget specification. While nugget refinement guarantees the accuracy by exploiting both of them. $2)$ Data mining techniques, such as global pattern detection algorithms, need to be told the specific type of pattern that a user is looking for. Such information may not be always available, because users may not know the exact pattern types that are important to them. While for nuggets refinement, the users' interests have already been indicated by candidate nuggets, which are usually small subparts of a whole dataset. Thus the refinement process could run different local pattern detection algorithms to figure out what users are looking for. $3)$ Even assuming the system knows the specific pattern type a user is interested in, in many cases the user is not searching for all possible patterns but only for certain patterns of this type. This makes running expensive global pattern detection algorithms not cost effective and unrelated patterns detected may even cost users more effort to isolate the useful ones.

In this work, we chose density-based clustering [26] and distance- based outlier detection [39] as our sample pattern detection algorithms, which are popular algorithms extensively studied in the literature [28, 52]. However, other search methods from the literature could equally be used.

2.3.2 Techniques for Nugget Refinement

Generally, the actual refinement is divided into two phases, called the match and the refine phases.

Match phase: In this phase, we aim to match the identified nuggets with patterns "around them" within the data space. In other words, our goal is to determine which patterns users were searching for when these specific nuggets were made. Briefly, the concept of "Match" is used to judge whether some data patterns or the major parts of these patterns primarily contribute to a nugget. If it is the case, we call the nugget and these patterns "matched". Figure 5.1 shows a good example of a "match" between a nugget and a cluster pattern in the dataset. The specific techniques utilized to calculated how much a nugget is "matched" with the patterns around it will be described in Chapter 5.

Refinement Phase: The match phase reveals to us what type of patterns that a user was searching for. With this knowledge, we can finish nugget refinement using the two steps of splitting (if necessary) and modification. These two steps will make each nugget a perfect representative of a single pattern. Details of them can be found in Chapter 5.

2.4 Nugget Maintenance

In this section, we will discuss maintenance of the nugget pool. Over the duration exploration, two potential hazards may leave "bad" nuggets in the nugget pool. 1) Out-of-date Nuggets: Some out-of-date nuggets extracted early on and no longer of interest to users may become an unnecessary burden. For example, a user who was searching for patterns in "Car" dataset mainly paid attention to luxurious cars at one month ago. Some nugget were extracted during her exploration at that time. However, as her budget shrinks, her recent exploration is only of cheap cars. Then, keeping the nuggets about the luxurious

cars will bring no help but more disturbance to her further exploration. 2) Misinterpreted Nuggets: Some nuggets may have been wrongly learned by misinterpreting users' interests.

To exclude these useless nuggets from our nugget pool, we introduce the concept of "vitality". Generally, the "vitality" of a nugget reflects the importance of this nugget. We use accumulated "visiting time" as its main indicator. A similar idea can be found in the literature [12, 2]. Specifically, each nugget obtains an initial "vitality" when it is extracted. This "vitality" fades as users' exploration period increases. A nugget can also gain "vitality" through two methods. 1) Being Directly Visited: If a nugget is retrieved by a user from the nugget pool, the time that this user spent on it counts for its "vitality" increase. 2) Being Indirectly Visited: An existing nugget is indirectly visited if a similar new nugget is combined into it. Once a new nugget is clustered into an existing nugget, this existing nugget absorbs its initial "vitality", which means the "vitality" of the existing nugget will be increased by the same amount of the initial "vitality" of the new nugget. Thus, briefly, nuggets created recently or visited frequently will have higher "vitalities", while those extracted a long time ago and never visited thereafter will have lower ones. Once the "vitality" of a nugget drops below a certain threshold, the nugget is retired from the system.

In NMS, such a natural "evolution" process of nuggets can also be controlled by users. NMS allows users to cease, quicken, or slow down the "evolution" by setting different parameters, such as initial "vitality", fading rate, and increasing rate. Besides such macro-control, users can also directly manipulate any individual nugget. For example, users can mark a nugget as crucial, indicating that it should never be expired from the system. They could also directly delete some useless nuggets. Nugget maintenance leaves many opportunities for our future work, including: 1) How to give proper rights to multiple users working on the same nugget pool. 2) How to automatically learn and modify the param-

eters controlling nugget pool "evolution" during users' exploration. These problems are important but not selected to be the topic of focus for this thesis work.

2.5 Nugget-Guided Exploration

Nugget-guided exploration makes use of the nuggets we have learned to facilitate the knowledge discovery process. Figure 2.6 shows a screen shot of our prototype system. As we mentioned earlier, nuggets, as important carriers of valuable information, can be augmented with different kinds of additional attributes. Besides "vitality", names and annotations are examples of other attributes that can enrich the meaning of nuggets. For example, when exploring a dataset about "arriving passengers", a border control officer finds a nugget that represents a cluster existing in dimensions of "nationality", "arriving time", and "criminal records", she can give the nugget a meaningful name (i.e., "Suspicious Passengers Group"), and attach an annotation about her concerns to this passenger group. Such attached information will not only make it more convenient to retrieve this nugget, but also makes her nuggets shareable with other users. Meanwhile, statistic information, such as the number of data records included, average and extreme values on each dimension, can be automatically computed and attached to the nugget.

Nugget-guided exploration can be carried out in both user- and system-initiated modes. 1) User-initiated: Within this mode, users take the initiated to search and retrieve nuggets when they desire to. NMS provides functionalities, such as sorting and querying on statistic information, key word based search on names and annotations, to help users quickly access the nuggets of interest. 2) System-initiated: NMS can take the initiative also when guiding users' exploration. Such guidance will be given based on watching the users' exploration. For example, when a user is querying a subpart of the dataset that is similar to one of the existing nuggets, NMS could inform the user that previous users have already

found a nugget similar to what she is looking for.

Other sophisticated services, which are not the key focus of this thesis work, are also critical for NMS and will be studied in our future work. They include: How to build hierarchical structures among nuggets based on their interrelation (e.g. some nuggets may be subparts of a bigger nugget). How to guide users based on their profiles using collaborative filtering techniques [15].

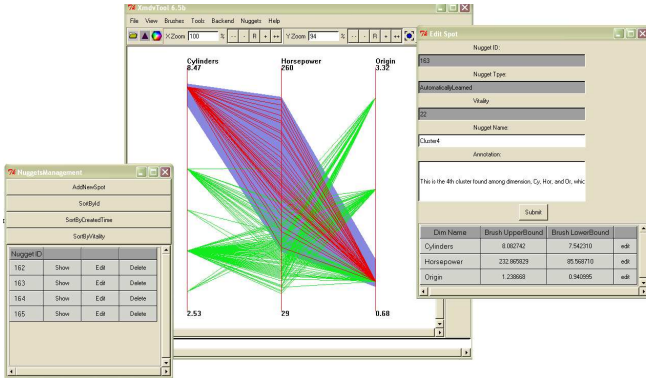

Figure 2.6: A screen shot from the NMS prototype when looking for clusters hidden in the dataset

Chapter 3

Distance Metrics Between Nuggets

As we mentioned in Chapter 2, relying on nuggets extraction alone may cause redundant nuggets. In this chapter, we introduce our solution of nugget combination, which keeps the nugget pool in modest size yet with high representativeness.

Several different techniques, such as sampling [10], filtering [60] and clustering [12] of nuggets may be employed to achieve this goal. After careful comparison, we choose nugget clustering, which groups similar nuggets together and generates representative for each group. The reason for this has been explained in Chapter 2.

Clustering aims to group objects based on their similarities. They require a distance measure that best expresses the domain specific similarity between objects. In our work, one of the main issues we have to tackle is thus the development of a suitable distance metrics for our multi-dimensional nuggets. To solve the problem, distance metrics are developed to effectively capture the distance between any pair of nuggets.

3.1 Query Distance

Nuggets are defined by both queries and their results. So, naturally, nuggets that are defined by similar queries should be considered to be more similar than those defined by rather different queries.

Thus our problem can be transformed into the problem of how to quantify the similarity of queries. Fortunately, previous work [65, 67] has studied this problem. The major principle utilized for measuring the query similarity (QS) can be summarized as Formula 1,

$$QS(A, B) = \frac{QA \cap QB}{QA \cup QB} \qquad (1)$$

where QS(A,B) represents the query similarity between Nugget A and Nugget B, and QA and QB are the qualifiers of these two queries. We adopt this basic idea as starting point for the design of our similarity measure. However, several issues have to be refined. First, we focus our attention on metrics for query similarity on a single dimension. Two main types of domains are considered:

- *Discrete Domains* : A discrete domain composed of nominal values is easy to handle. Because of the discrete property, a direct use of Formula 1 indeed solves the problem. For example, given two queries over the nominal domain, QA: select * from X where X.origin=$\{Japan, US, Germany\}$, QB: select * from X where X.origin=$\{Japan, US, Italy\}$, we just need to count the number of elements that fall into the intersection and the union of these two sets and then we get $|QA \cap QB| = |Japan, US| = 2, |QA \cup QB| = |Japan, US, Germany, Italy| = 4$, and thus QS(A,B)=2/4=0.5. Clearly, this strategy of counting key words can also be used in numeric discrete domains.

- *Continuous Domains* : Intuitively, a straightforward variant of the previous "count method" can also be used for continuous domains. The intersection and union of two range queries are no longer expressed by a count of the elements, but rather by the "length" of overlap and total coverage. For example, given QA: select * from X where X.height =[5.25:5.85], QB: select * from X where X.height=[5.45:6.15], then we have $QA \cap QB = 5.85 - 5.45 = 0.40, QA \cup QB = 6.15 - 5.25 = 0.9$, QS=0.4/0.9=0.44.

  However, although the major principle of Formula 4.0 still holds for continuous domains, a more careful consideration regarding the continuity of the domain may be needed. A problem rises as that in a domain of size from 0 to 1000, if we decide that two range queries over [1.00:2.00] and [1.50:2.50] respectively have some similarities, should we assert that two queries over [1.00:2.00] and [2.00:2.50] are totally dissimilar just because they do not happen to overlap each other? An example will illustrate this concern better.

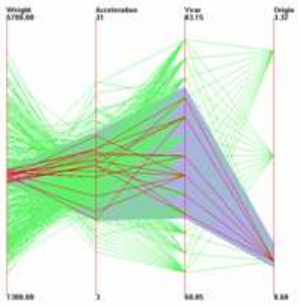

Figure 3.1: Query X

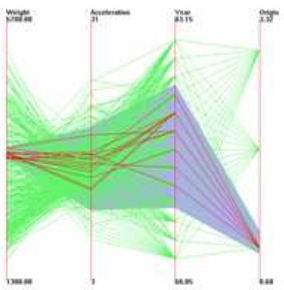

Figure 3.2: Query Y

As shown in Figures 3.1 and 3.2, queries X and Y on dimension "Weight" are [3051.73:3318.68] and [3327.02:3527.23] respectively. We note that even though they do not overlap, visually the nuggets defined by them are quite similar. So, in order for our metric to capture the broader semantics of similarity, we have developed a more general algorithm that handles both types of domains, while still

keeping the essence of Formula 1. In this algorithm, the domain will be divided into discrete bins. If some part of a query falls into a bin, we call the bin an "occupied bin (ob)" of the query. Finally, we utilize the "occupied bin count strategy" (obcs) when comparing two queries.

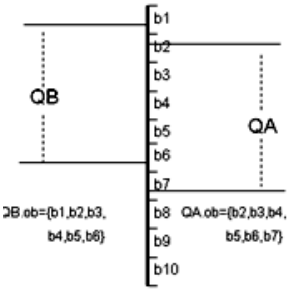

Figure 3.3: Overlap case in "Occupied Bin Count Strategy"

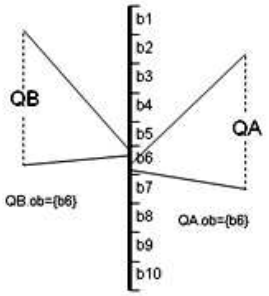

Figure 3.4: non-overlap case in "Occupied Bin Count Strategy"

As demonstrated in Figures 3.3 and 3.4, now both overlap and non-overlap cases are handled by our new algorithm. In Figure 3.3, $QA \cap QB = |QA.ob \cap QB.ob| = |\{b2, b3, b4, b5, b6\}| QA \cup QB = |QA.ob \cup QB.ob| = |\{b1, b2, b3, b4, b5, b6, b7\}| =$ 7,and QS=5/7=7.1. In the Figure 3.4, $QA \cap QB = |QA.ob \cap QB.ob| = |\{b6\}| = 1, QA \cup QB = |QA.ob \cup QB.ob| = |\{b6\}| = 1,$ QS=1/1=1. In practice, we could also set QS less then 1 for non-overlap cases, because after all they are not perfect matches.

Actually, as we've seen, the "occupied bin count strategy" can be used as a uniform query similarity metric for range queries on a single dimension.

$$QS(A, B) = \frac{QA.ob \cap QB.ob}{QA.ob \cup QB.ob} \qquad (3)$$

The difference between discrete and continuous domain cases is that the former uses each discrete value as its bin, while the latter divides the continuous domain into bins first.

Nonetheless, in most cases, datasets are multi-dimensional, and so are the queries defining our nuggets. Thus, we have to extend the previous metric defined for a single dimension to now be applicable for multiple dimensions. In this work, we adopt minimum single–dimensional query similarity among all the dimensions of two multi-dimensional queries to represent the query similarity between them. To guarantee the "visual similarity" of two nuggets, we choose the minimum but not other combination methods, such as Manhattan Distance or Euclidean Distance [1]. The later may compromise to large differences on single dimensions if exist some highly matched ones. To better explain our choice, we use a concrete example to demonstrate it. As shown in Figures 3.5 and 3.6, although two queries L and K have completely same selective range on 13 dimensions (the second to the fourteenth dimension from left), they are very different on 1 dimension, which is the first dimension. In this case, if we use Manhattan Distance or Euclidean Distance, since the query distances on 13 dimensions are all 0, even if the distance on the first dimension is huge, the final query distance between these two queries will still be very small. However, as we can see, these two queries are actually very different because they are about different "Types" of products. That means the users who specified them were interested about the different things and we should not cluster their queries together. Further more, if our distance metric beleives these two queries are similar and eventually cluster them together, another problem may arise. That is the selective range of the representative on the first dimension will reflect the "average" of these two query, and thus may be the "Type" in the middle which never appeared in both of them. Then, what we store in the system is a representative that is very different from both of the original queries, and the real interesting queries for users are lost. In contrast, when we choose "minimum", our distance metric will give small similarity for these two queries, because the minimum similarity among all the dimensions is that on the first dimension, and it is obviously small.

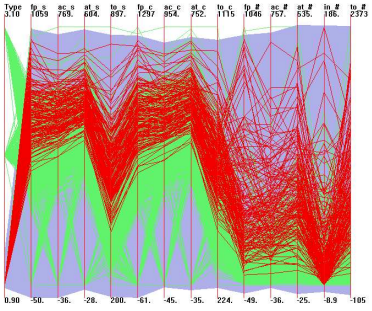

Figure 3.5: Query L

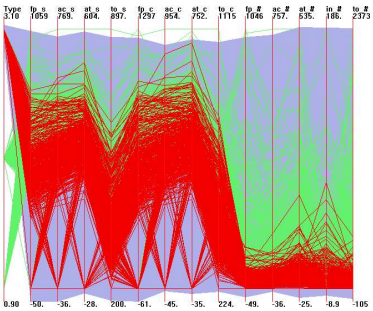

Figure 3.6: Query K

Finally, when we've successfully acquired normalized query similarities (between 0-1), we can now easily calculate the query distances (QD) as shown in Formula 2.

$$QD(A, B) = 1 - QS(A, B) \qquad (2)$$

3.2 Data Distance

Our query distance metric effectively measures the similarity between two nuggets based on their query specification. However, nuggets are not only characterized by their queries (profile), but also by the results of the queries obtained when applying the queries to a particular dataset (content).

As shown in Figures 2.2 and 2.3, two nuggets generated by very similar queries may be rather different in terms of actual data content. The former contains a cluster , while the latter is empty. Clearly, we need to enhance the capability of our distance metrics by comparing the "contents" of the nuggets. Now, the problem we must solve can be viewed as a general date analysis problem. That is, given two subsets of a multi-dimensional dataset, how could we measure the distance between them. Previous works to tackle such problems [8, 20, 48, 19] can be generally classified into two main categories, statistic and transform-cost approaches. Below, we will explain why we choose the latter, and then introduce a proposed algorithm based on extending a basic transform cost algorithm.

3.2.1 Statistic Approach

Since traditional statistic methods, such as average and deviation, cannot fully capture the characteristics of two subsets, a more sophisticated method, histograms has been developed, which is called Histogram Difference Measures(HDM). HDM based on the average relative error [8] of aggregation is used in data abstraction quality measure [20], approximate query processing of databases as well as image similarity measures [47, 59]. It relies on comparing the histograms of two sets of data, meaning, it focuses on the distributions of data points. When measuring multidimensional datasets, NHM can be carried out by either an integration of single-dimensional histograms or by a single multi-dimensional histogram.

However, both histogram methods tend to suffer from different but not ignorable disadvantages. For multi-dimensional histograms, the number of bins grows exponentially when the number of dimensions increases, thus the complexity can easily reach an unaffordable level even with a modest number of bins and dimensions. For example, if we have 10 dimensions and divide each dimension into 10 bins, we need 10^{10} comparisons. On the other hand, the integration of single-dimensional histograms first compares histograms on each dimension separately and then integrates the results into a normalized result. This is similar to what have done for the query distance. It has a linear complexity O(b*k) (with b: number of bins on each dimension, and the k: the number of dimensions). But such integration cannot truly reflect the distribution of data points in many cases. For example, dataset $A\{a1(length = 1, width1), a2(length = 10, width = 10)\}$ and dataset $B\{b1(length = 1, width = 10), b2(length = 10, width = 1)\}$ will be measured to be exactly the same by this method, since they have the same distribution on each individual dimension. Even though, these two datasets actually have very different data records.

3.2.2 Transform Cost Approach

As a general notion, Transform Cost has been shown to be effective in a wide range of different areas, such as "Edit Distance" in string matching [48], and "DIFF" in change detection to HTML and XML files [19]. In Transform Cost Approach, distance between two objects is expressed as the minimum cost of transforming one object to another. A well known algorithm that relies on Transform Cost is Nearest Neighbor Measure (NNM). NNM is used in measuring data abstraction quality [20] and image quality [54]. When comparing two datasets, NNM aims to move each data point (record) in one set to its nearest neighbor in the other set. It then calculates the accumulative distance that all the data points have moved. Generally, it is more precise than the Statistic Approach, because it deals with each data point rather than only general statistic information of datasets. But unfortunately, NNM appears to work better for measuring the quality of representativeness due its n to 1 mapping strategy. Let us see an example that shows the deficiency of this method. Given two dataset: dataset $A\{a1(length = 1), a2(length = 100), a3(length = 100), ..., a99(length = 100), a100(length = 100)\}$ and dataset $B\{b1(length = 1), b2(length = 100)\}$ would be measured to be exactly same, for each element in set A finds a 0 distance nearest neighbor in set B. In short, NNM is a population-insensitive algorithm. It may lead to bad comparison results in our case, because comparing nuggets with different populations is going to be the norm in our work . To solve this problem, we propose a new algorithm called Exact Transformation Measure (ETM), which not only overcomes the population-insensitivity but also is more effective in capturing visual similarity of two datasets. Before discussing the specific algorithm, let us first formulate the problem:

Given dataset $O, |O| = m$, and datasets A and B, $A \subseteq O, B \subseteq O, |A| = a, |B| = b, 0 \leq a \leq b \leq m, |A \cap B| = l, |B| - |A \cap B| = n$, data points in O can be viewed as geometrically distributed in the value space based on their values in different dimensions,

we transform A to be exactly equal to B with minimum cost.

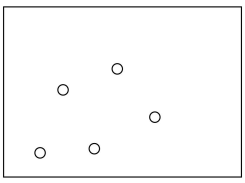

Figure 3.7: Dataset A

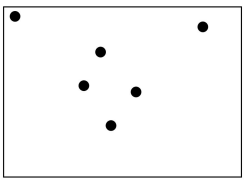

Figure 3.8: Dataset B

To solve such a problem, simply moving data points in A to their nearest neighbors in B will fail in many cases, because it is neither globally optimal nor sensitive to population. Thus, in order to achieve the transformation with minimum cost, we define three types of operations:

- *Move(x, y): given $x \in A, y \in B$, move x to the position where y lies.*
- *Add(x, y): given $y \in B$, add a new data point x to A at the same position where y lies.*

By using “Move” and “Add”, we are guaranteed to always be able to transform A to B, since A always has a smaller or equal sized population to that of B. However, simply relying on “Move” and ”Add” will impose “forced matches”, which may not always lead to capture of the real distance between two datasets. Figure 3.9 shows an example of two 2-dimensional datasets where moving and adding are not sufficient to make a cost effective transformation plan.

Given dataset A (Figure 3.7) and B (Figure 3.8) as shown in Figure 3.9, by using “Move” and “Add” only, we have to match some data points in A with data points in B that are far away from them. While the “Delete” operation helps would us to achieve a more cost-effective transformation, as shown in Figure 3.10.

In the worst case, the existence of a few “outlier” data points that do not have a “near neighbor” close to them will deprive opportunities for many of other data points to be

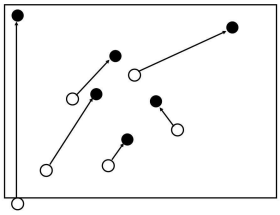

Figure 3.9: Trasforming A to B with moving and adding operation only

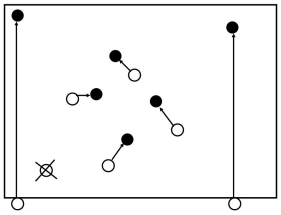

Figure 3.10: Transforming A to B with, moving, adding and deleting

matched with their real nearest neighbors. To deal with this disadvantage of "forced matches", we introduce another type of operation, namely, "Delete".

- *Delete(x)* $x \in A$*, delete x from A.*

With the help of the "Delete" operation, we no longer need to suffer from "forced matches", because for a given data point in A, "Move" is no longer the only option for it. We can choose to "Delete", if moving it will bring too much global cost. However, how to make an optimal transformation plan, which has the minimum cost, is still a complex problem. In order to tackle this problem, we need to study the cost model of each operation first.

- *Cost of Move(x,y)* –$Cost(M[x, y])$

  Cost of moving a data point x to y is equal to the distance between x and y. Here, we adopt the Euclidean Distance(normalized, between 0-1), which is the most widely used distance measure between two objects in a multi-dimensional space.

- *Cost of Add(x,y)* –$Cost(A[x, y])$

  Since Cost(A[x,y]) is usually an estimated value rather than any physical distance, in most of the Transform Cost works, a single COA (cost of adding, which is independent from the position where the point will be added) is used for each transformation. In this work, we adopt this single COA strategy, while developing a

new method of estimation. Considering that a point is directly added to a certain position, the adding process is composed by two steps: a generation (generating a point at a random position) and a moving (moving the point to a certain position). Thus, COA can be expressed in the following way:

$$COA = GC + MC \qquad (5)$$

a) Generating Cost (GC): As mentioned in [63], COA grows as the size of the original set (dataset A, in our case) decreases. It is not hard to see that generating a new element for A would cause greater a "mutation" to it, when A is small. For instance, when A is an empty set ($|A| = 0$), generating a new data point for A will thoroughly change it, while if a=100,000, such a generation can hardly make a noticeable difference. So, we correlate GC with the cardinality of A:

$$GM = \frac{MPD}{a+1} \qquad (5)$$

With MPD: the maximum possible distance between two datasets is equal to 1. We add 1 to the divider to handle the case that a=0.)

b) Moving Cost (MC): When a new data point is generated for A, it has a random position. Thus, since we cannot truly calculate its distance from the position it should be moved to, we use the average distance between two datasets (centroid to centorid) to estimate the MC needed for moving it to this certain position.

Generally, COA as an estimated value has a positive association with the average distance between two datasets and negative association with the cardinality of A. It should be more expensive than most of the Cost(M[x,y]) in a transformation. For normalization reason, we set the upper limit 1 to it.

- *Cost of Delete(x)–* $Cost(D[x])$

  Similar to GC, the change cost of deleting is associated with the cardinality of A and unrelated to the position where the deleted data point lies. The difference here is that we do not need to handle the cases where a=0, because we can delete a data point only if it exists. So, we use Cost Of Delete (COD) to express all the Cost [D(x)] in a transformation:

$$COD = \frac{MPD}{a} \tag{6}$$

Having defined the cost models of all our transfer operations, we now establish our solution for finding an optimal (most cost-effective) transformation plan. We note that making such an optimal transformation plan is non-trivial. Fortunately, the Hungarian Assignment [62, 46] which was designed for finding minimum cost bipartite matches, provides a good approach to solve this NP- hard-like problem in polynomial time. The algorithm takes a $n \times n$ matrix as input. Each row in the matrix represents a data point in A, and each column represents a data point in B. Then each entry is filled with the distance between the row and the column it belongs to. The algorithm returns a minimum cost match in $O(n^3)$ time.

Let us see a simple example of how it works. Given a 2D dataset $A\{a_1(0,1), a_2(0,4), a_3(0,7)\}$, a dataset $B\{b_1(0,3), b_2(0,6), b_3(9,9)\}$. We know the domain for both dimensions is (0-10), then the input matrix will be as shown in Figure 3.11. After a series of matrix manipulations, the output matrix will have exact one "0" in each row and each column, which stands for the "match" of two data points. For example, in the output matrix below, since there's a "0" appearing at the entry[a_1, b_1], data point a1 should be moved to b_1 (Figure 3.12).

The details of Hungarian Assignment Method can be found in [62, 46].

	b1	b2	b3
a1	0.14	0.35	0.85
a2	0.07	0.14	0.72
a3	0.28	0.07	0.65

Figure 3.11: Input matrix

	b1	b2	b3
a1	0		
a2		0	
a3			0

Figure 3.12: Output matrix

As mentioned before, we encountered several issues to be addressed. The first one is that two subsets to be compared do not necessarily have the same population. Also, we need to incorporate the adding and deleting actions into the transformation plan. To achieve these goals and thus complete the design of our global transformation plan, two modifications to the input matrix are needed.

- *Dummy Points*

  When two subsets have different numbers of members ($a < b$), an input matrix with distances between points only would not be a squared matrix required by Hungarian Assignment Method as input. To deal with this, we add dummy points to A to produce a squared input matrix. The distance between a dummy point d_i and any real data point in B should be equal to COA, because when the algorithm eventually makes a match between d_i and b_i, then this means a new point will be added to A at the same position where b_i lies, and thus it costs COA. For example, if we add one more point b4 (0, 0) to the above dataset B, then the input matrix will be as depicted in Figure 3.13.

- *Incorporating Adding and Deleting*

  Recall that adding and deleting actions are essential for avoiding a "forced match". Thus, we need to consider them comprehensively with moving actions when making the transformation plan. Specifically, we have to incorporate COA and COD into the input matrix properly. The key idea here is that when moving a_i to b_i is

	b1	b2	b3	b4
a1	0.14	0.35	0.85	0.07
a2	0.07	0.14	0.72	0.28
a3	0.28	0.07	0.65	0.49
d1	0.58	0.58	0.58	0.58

Figure 3.13: Input matrix with dummy points (COA=0.58)

	b1	b2	b3	b4
a1	0.14	0.35	0.83	0.07
a2	0.07	0.14	0.72	0.28
a3	0.28	0.07	0.65	0.49
d1	0.58	0.58	0.58	0.58

Figure 3.14: Input matrix incorporated with adding and deleting. Cost(M[a1,b3])is replaced by COA+COD=0.83

even more expensive then deleting it and adding a new data point to where bi lies, we choose the later "deleting + adding" strategy instead of moving. Thus, in the input matrix, if the original value of an entry $Cost(M[a_i, b_i]) > COA + COD$, we use COA+COD to replace the original value. The example is shown in Figure 3.14.

Now we've discussed all the techniques needed to make a proper input matrix that can lead to an output matrix representing the optimal matches. Once the output matrix has been produced, by simply summing all the values in the input matrix entries, which match entry location with a "0" in its output matrix, and dividing the sum by population of B, we get the Data Distance $(DD[X, Y])$, between two nuggets, where X and Y are two nuggets and A and B are the datasets contained by them respectively.

Finally, we combine the Query Distance $(QD[X, Y])$ and Data Distance $(DD[X, Y])$ to present the Nugget Distance $(ND[X, Y])$ between any pair of nuggets X and Y.

$$ND[X, Y] = \alpha \cdot QD[X, Y] + \beta \cdot DD[X, Y] \qquad (\alpha + \beta = 1) \qquad (6)$$

Since QD and DD are both normalized (between 0 to 1), ND will be normalized as well.

3.3 Evaluation of Distance Metrics: User Study

Now, we discuss several experimental studies we have conducted to compare the effectiveness of different distance metrics introduced in this chapter.

3.3.1 Experimental Setup:

Experimental environment: This user study was carried out in a web-based environment. A web page which carried the instructions and all the questions was posted on the Internet.

Users: Although the user study was posted on the Internet and accessible for all Internet users, it was only advertised to WPI community by e-mail. So, generally, the users engaged in this user study were volunteers that are WPI students, faculties or staff.

Datasets: Three real datasets are employed in our user study. They are the "Iris" dataset (4 dimensions, 150 records); the "Cars" dataset (7 dimensions, 392 records); and the "Aaup" dataset (14 dimensions, 1161 records).

Nuggets: We have designed twenty pairs of synthetic nuggets which are based on the three real datasets we mentioned above. In particular, seven nuggets each are based on "Iris" and "Cars", and the other six are extracted from "Aaup". These synthetic nuggets are examples of the real nuggets which users could make in their navigations, because they covered all the pattern types we discussed in this work and have different sizes. Specifically, the smallest nugget we used in this user study was based on "Iris" dataset. It had very short selective range on all the four dimensions and contained only two data records. In contrast, the largest one, which was based on "Aaup" dataset had large selective range on all 14 dimensions and contained 543 data records.

Questions: We had twenty questions in the user study. Each of them requires users to give a distance between a pair of nuggets. Particularly, all the distances were scaled by

the integers from 0 to 10 which were presented by eleven radius buttons. The suggestive semantics of each integer were also shown under the radius buttons. Specifically, 0-1 means “very similar”, 2-4 means “similar”, 5-7 means “unsimilar” and 8-10 means “totally different”. The default answer for all the questions was “NULL”, which means no radius button was initially selected when the questions were shown to the users. The sequence of the questions was randomly arranged, but once it was arranged , it was kept identical for all users.

Results: All the results of the user study were automatically collected loaded into our database through the web page.

Experimental Methodology: As mentioned above, this user study was carried out through Internet. Users volunteered to participate in the user study and they could choose to either provide their personal information, including name, occupation, major (if student), and e-mail address, or not. An brief instruction for the user study was given before the specific questions were presented to the user. This instruction introduced the concept of “nugget” and asked users to answer all the questions based on their own intuition. Users were asked to answer all the questions without communication with any other. However, they were encouraged to contact us, the user study hosts, if they had any problem at any stage of the user study both by e-mail or in person. During the user study, users could go back to reanswer any previously answered question and they could answer questions in any order. However, they had to answer all the twenty questions before they could submit their answers.

When analyzing the final results, we found that 20 users participated in our user study and all of them left at least part of their personal information. From those personal information, we learned that they were all WPI students but from different majors. Five of the twenty users contacted us in person before answering the questions in order to fully understand the instruction. Our communication with them was restricted to the contents

of the instruction itself. All such sessions were shorter than 10 minutes each.

Experimental Strategy: We applied each individual distance metric (one query distance metric QD and three data distances HDM, NNM and ETM) and all the combined distance metrics (query distance + data distance: QD+HDM, QD+NNM, QD+ETM) to compute the distances between the same 20 pairs of nuggets. Finally, we compare the distances given by the users with those computed by each of the metrics. For this, we introduce a function "Dif", which is used to express the difference between the distances given by a metric versus by a user. For each user U, each distance metric M and a certain pair of nuggets N_x, N_y, we compute Dif=$|D_U(N_x, N_y) - D_M(N_x, N_y)|$. In our comparison, we first utilize a comparison strategies, which we call Accurate Credits Strategy (ACS). In ACS, for each pair of nuggets, we assign different amount of credit, called accurate credit, to each metrics based on the difference between the distances given by this metric and by the user. Concretely, we give 3 credits to a metrics if Dif=0, and we give 2 credits to it if $Dif = 1$ and 1 credit if $Dif = 2$. If $Dif > 2$, no credit will be given to the metric, meaning that the distance metric fails to match with the user's intuition. For all 20 pairs of nuggets and all 20 users (totally 400 distances given), we calculate these accumulative credits for each of the metrics. Besides ACS, we use pie graphs, which we call "Dif Distribution Graphs" to observe and compare the Dif distribution of each metric. For each metric, it will show us the exact number and percentage of each "match category" (Dif=0, Dif=1, and so on).

3.3.2 Experimental Results:

Figure 3.15 shows the accurate credits earned by each metric.

Since the distances given by users were based on their own intution, they may be different for a same pair of nuggets. Actually, we found that for all the questions, our twenty users gave more than one answers. However, as ACS counts all the "matches"

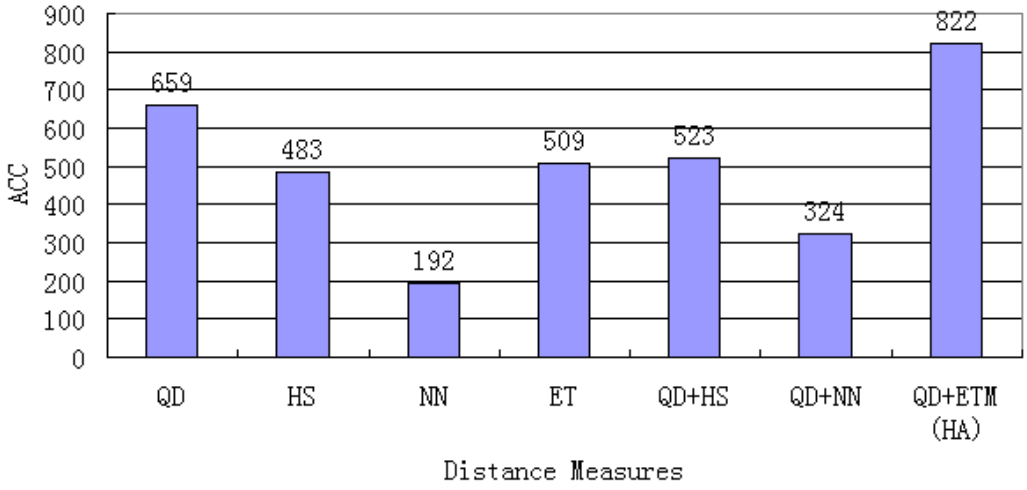

Figure 3.15: Accumulative credits earned by each distance metrics for all 400 cases

and sums up the Accurate Credit earned by each distance metric for all the 400 cases, it is a fair comparison for all the metrics. Generally, a metric that matches well with more users in more questions will earn higher accumulative credit. As shown in Figure 3.15, QD+ETM earns much higher accumulative credit than any other metric. This indicates that it matches the users intuition best among all the distance metrics.

Figures 3.16 to 3.22 show the distribution of "Dif"s for each distance metrics. From them, we can observe that QD+ETM has 128 (32%) "perfect matches" ($Dif = 0$) with users' ratings for the 400 distances. It also has 196 (49%) $Dif = 1$ matches, 44 (11%) $Dif = 2$ matches, while only 32 (8%) "non- matches" ($Dif > 2$). It is much better than any other distance metrics in terms of more "good matches" and less "non-matches", even when compared with the second best one, QD only, which has 88 (22%) $Dif = 0$ matches, 132 (32%) $Dif = 1$ matches, 136 (34%) $Dif = 2$ matches, and 44 (11%) "non=matches" ($Dif > 2$). Based on the comparison results above, we learn that QD+ETM captures the distances between nuggets best among all the metrics we discussed in this work.

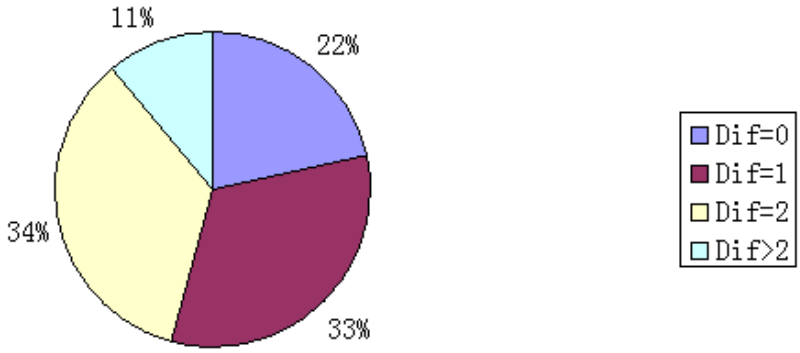

Figure 3.16: Dif distribution of QD

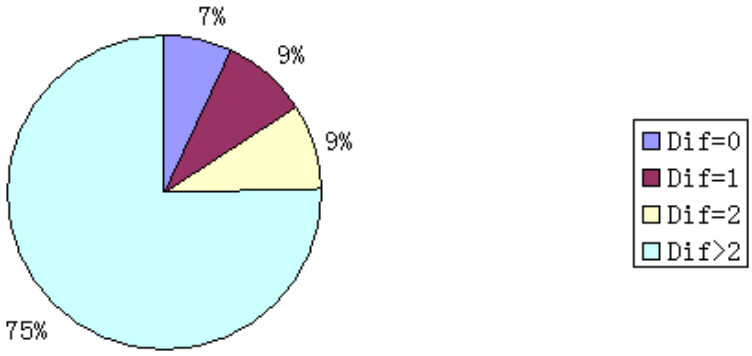

Figure 3.17: Dif distribution of NNM

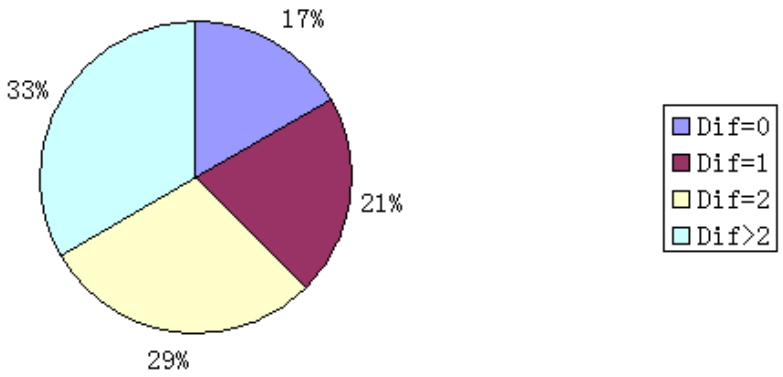

Figure 3.18: Dif distribution of HDM

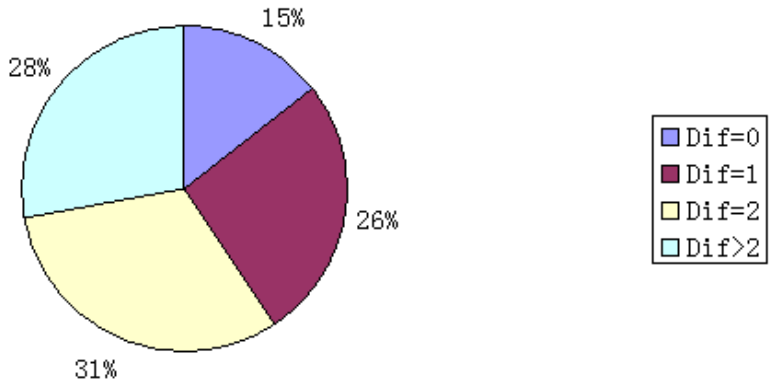

Figure 3.19: Dif distribution of ETM

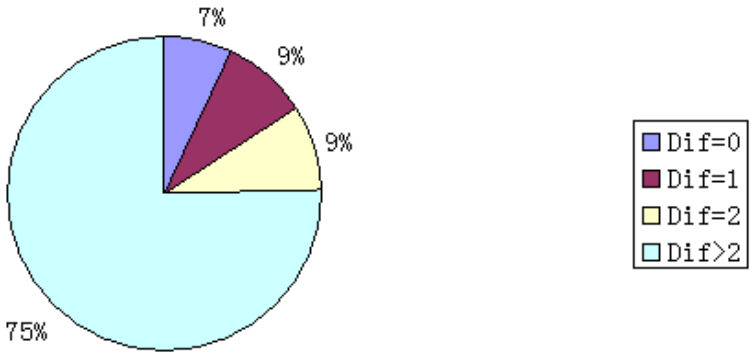

Figure 3.20: Dif distribution of QD+NNM

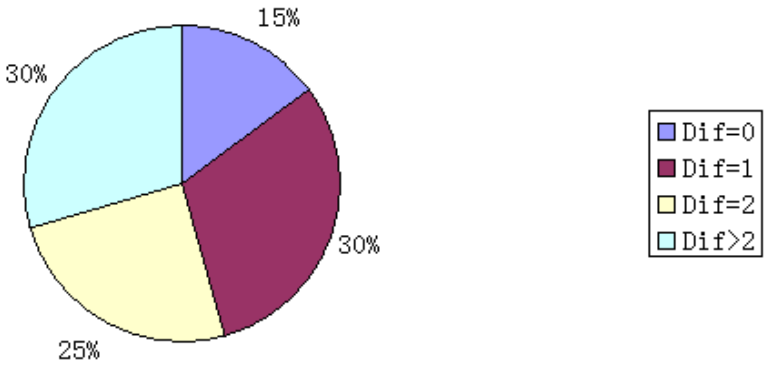

Figure 3.21: Dif distribution of QD+HDM

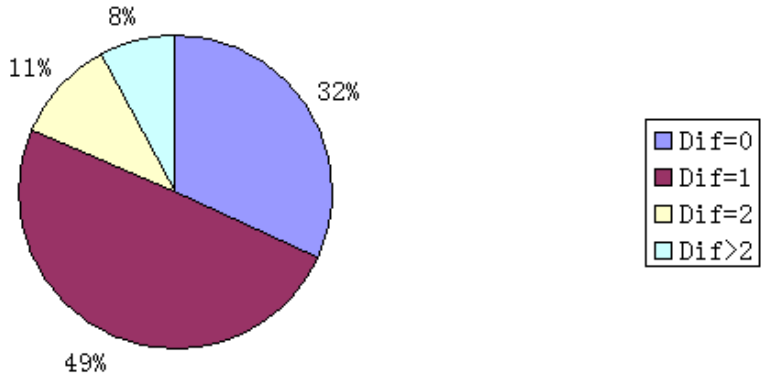

Figure 3.22: Dif distribution of QD+ETM

3.3.3 Approximation to Hungarian Assignment Algorithm

However, the best quality usually comes at the price the highest cost. Since the Hungarian Assignment (HA) Algorithm used in ETM has $O(n^3)$ complexity, where n is the number of points that appear in the larger subset but not in the smaller subset, it is not always practical performance-wise when we try to compare the nuggets with huge populations. Figure 3.23 shows the CPU time used by all the distance metrics when measuring distances for the 20 pairs of nuggets we mentioned earlier. We can see that QD+ETM has highest cost in terms of maximum, minimum and average CPU time used, which means now we have a metric that is best at capturing users' intuition but worst in terms of time efficiency.

To address this, we now propose to employ a much cheaper approximation algorithm of HA instead of the full-fledged HA algorithm. It is Coupon Collection (CC) Algorithm, which has $O(nln(m))$ complexity, where n has the same meaning with that in HA and m is the size of the original dataset. By using CC, we do not try to make global optimal

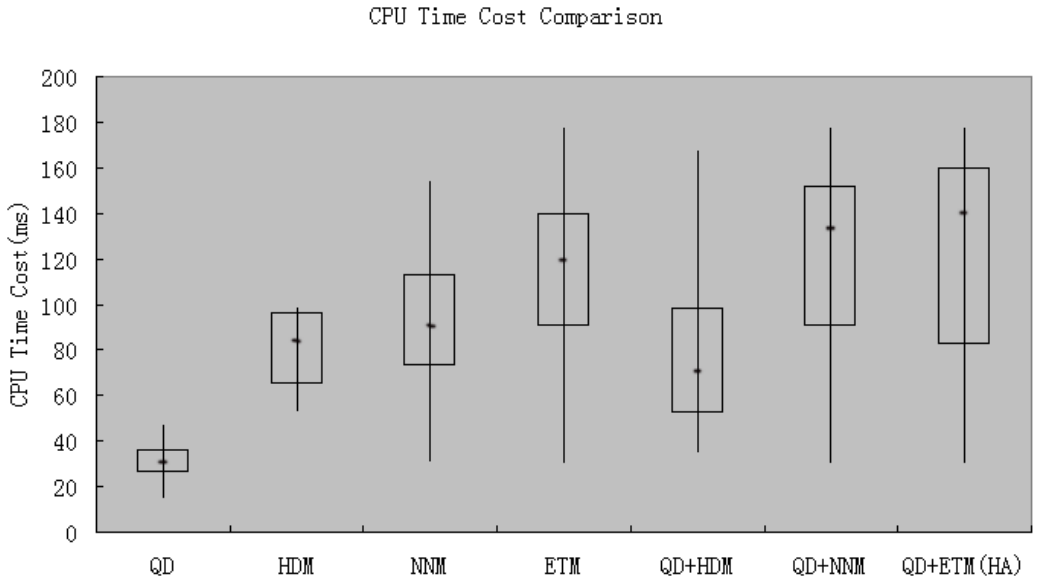

Figure 3.23: CPU time cost by different distance metrics

transformation plan by conducting complicated matrix operations as we did with HA. Instead, we “move” each non-overlaping data point in one nugget to its nearest neighbor in another. Once a data point from one nugget has been moved to a data point in the other nugget, the later data point is “occupied” and can no longer “accept” moving from any other data point. Thus, if the nearest neighbor of a data point is “occupied”, this data point has to be moved to its second nearest neighbor. And if the second nearest neighbor is “occupied” also, then we move it to its third nearest neighbor. This yielding strategy continues until a data point find an unoccpupied neighbor, or the data point has to be “deleted” in the transformation plan.

To study the performance and quality of ETM using CC, we run QD+ETM (with two different implementations of ETM: CC and HA) against 3000 pairs of synthetic nuggets extracted from 3 different datasets (1000 pairs per dataset). Since, theoretically, HA is guaranteed to give the minimum distance for ETM, we use our Dif Distribution Strategy to compare the distances by QD+ETM (CC) with those computed using QD+ETM (HA).

By doing this, we will be able to learn how often and how well the distances computed using QD+ETM (CC) match with those by QD+ETM (HA). If they have good matches in most of the cases, we can conclude that CC is a good approximation for HA used for computing ETM.

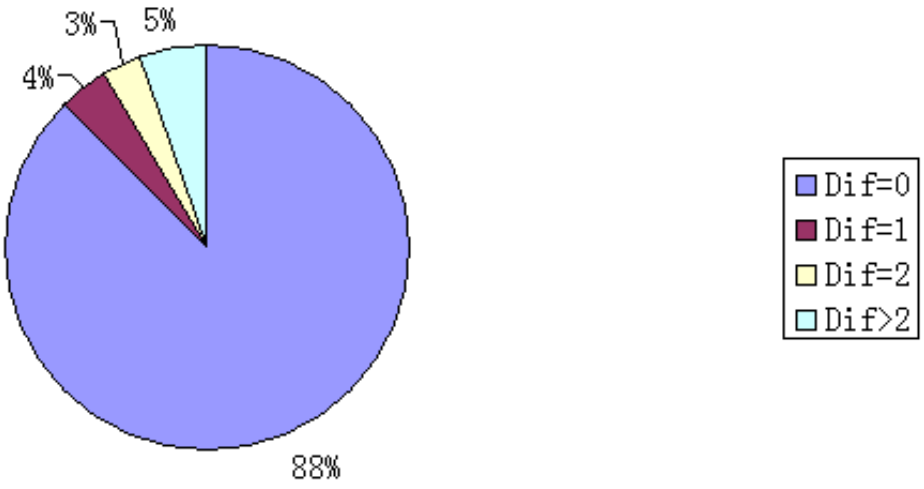

Figure 3.24: Dif distribution of CC and HA on 1000 pairs of nuggets from “Iris” dataset

As shown in Figures 3.24, 3.25 and 3.26, we observe that QD+ETM (CC) produces exactly the same answers as QD+ETM (HA) in around 90 percent of the time. And in only less than 5percent time, the Dif between them is larger than 2. These comparison results indicate that CC is a very good approximation to HA in terms of capturing the distances between nuggets.

Further more, to compare the CPU time cost of QD+ETM (CC) with the costs by other distance metrics and also its performance in terms of matching users’ intuition, we use QD+ETM(CC) to measure the distances between the same 20 pairs of nuggets we used in the earlier user study (Section 3.3).

As shown in Figure 3.27, from the maximum, minimum, average and also standard deviation of CPU time cost, we learn that QD+ETM(CC) is the second fastest distance

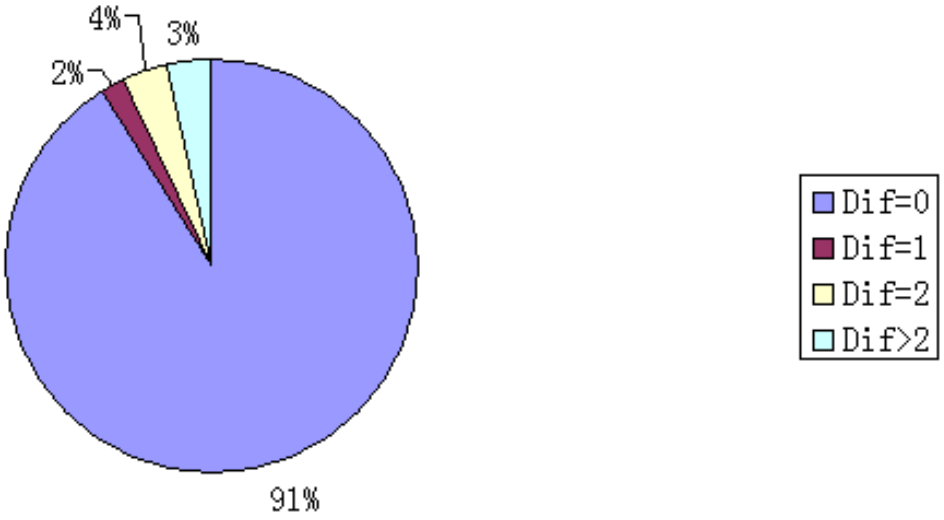

Figure 3.25: Dif distribution of CC and HA on 1000 pairs of nuggets from “Cars” dataset

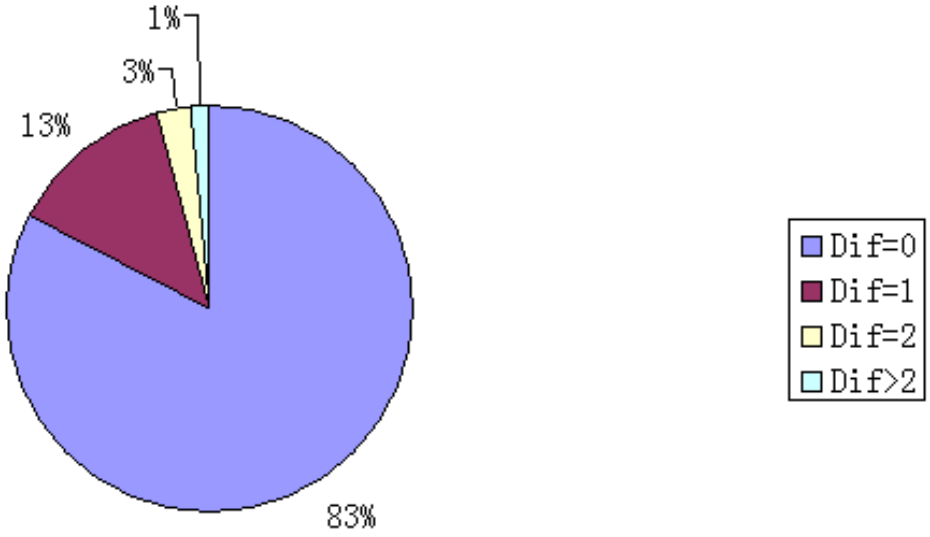

Figure 3.26: Dif distribution of CC and HA on 1000 pairs of nuggets from “Aaup” dataset

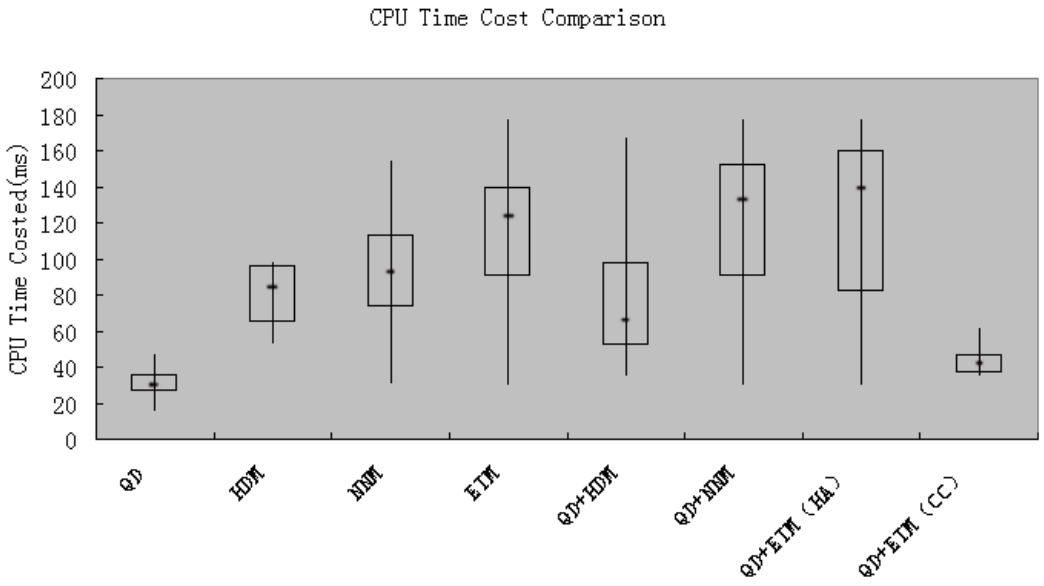

Figure 3.27: CPU time cost by different distance metrics

metric among all those we have discussed. It costs averagely around 75 percent less CPU time than QD+ETM (HA) and only slightly more than QD only, which is the cheapest metric. Here, we also need to point out that, although QD+ETM (CC) saves much CPU time in average case compared with QD+ETM (HA), QD+ETM (CC) is not guaranteed to be faster than QD+ETM (HA) in all cases. This is because the complexity of CC, $O(nln(m))$, is related to the size of the original datasets, but the complexity of HA, $O(n^3)$, is only related to the non-overlap population in the larger nugget n. CC can be slower than HA when m is extremely large, while n is extremely small, although this is not likely to happen in most of the cases. So, to be careful, we can choose to use either of these two metrics based on comparing $ln(m)$ and n^2. If the former wins, we pick QD+ETM (CC), or we pick QD+ETM (HA).

When comes to the performance of QD+ETM (CC) in terms of matching users' intuition, we found that among all the 20 pairs of nuggets we used in our user study, QD+ETM (CC) gave exactly same answer with QD+ETM (HA) in 18 pairs of them. For the re-

mained 2 pairs of nuggets, on which the distances given by QD+ETM (CC) and QD+ETM (HA) did not completely agree, one of them has a difference equal to 1 and another has a difference equal equal to 2.

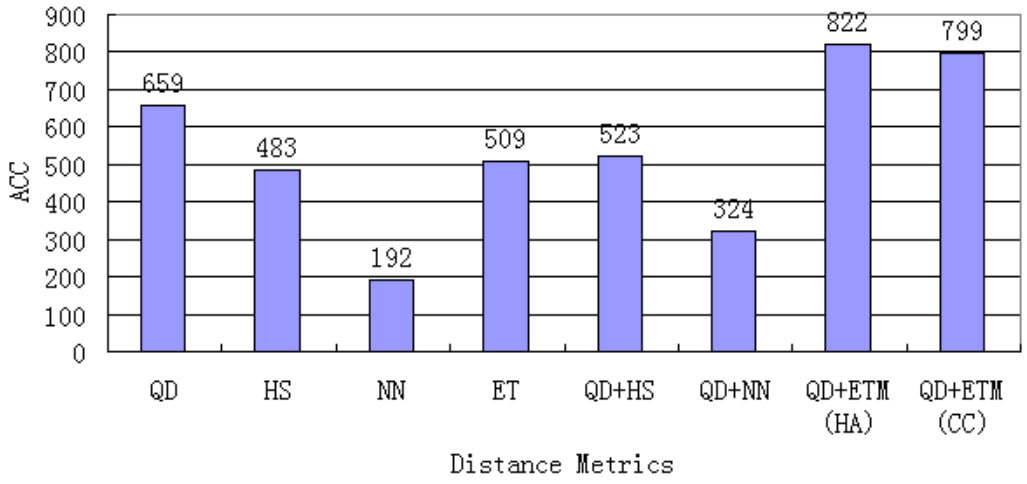

Figure 3.28: Accumulative credits earned by each distance metrics for all 400 cases

As shown in figure 3.28, the accumulative credits earned by QD+ETM (CC) are very close to those of QD+ETM (HA) and much higher than any other metrics. And from figure 3.29, we can observe that QD+ETM has 110 (28%) "perfect matches" ($Dif = 0$) with users' ratings for the 400 distances. It also has 210 (52%) $Dif = 1$ matches, 49 (12%) $Dif = 2$ matches, while only 32 (8%) "non- matches" ($Dif > 2$).

3.3.4 Conclusions on Distance Metrics

Based on the experimental studies we discussed in this chapter, we draw following conclusions:

1) QD+ETM agrees well with users' intuition on distances between nuggets. It is thus the best distance metrics among all we have discussed in this work in terms of capturing nugget distance.

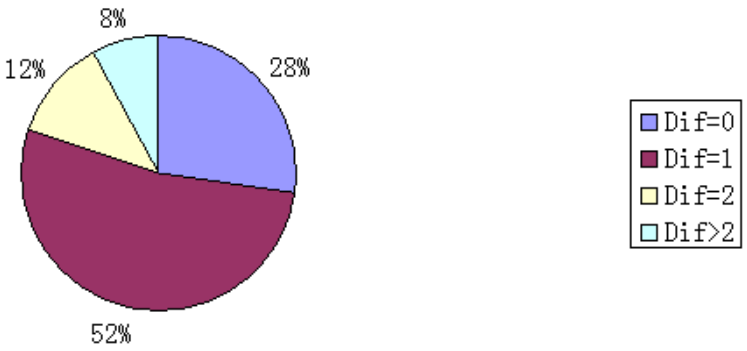

Figure 3.29: Dif distribution of QD+ETM (CC) on the 20 pairs of nuggets used in user study

2) Coupon Selection (CC) algorithm has been shown to be a good approximation to Hungarian Assignment (HA) algorithm used for computing ETM. It has almost the same performance with HA in terms of capturing distances between nuggets, and it costs averagely around 75% less CPU time than HA for the nuggets on three real datasets we employed ("Iris" dataset: 4 dimensions, 150 records; "Cars" dataset: 7 dimensions, 392 records; "Aaup" dataset: 14 dimensions, 1161 records).

3) Query Distance (QD) only, as a cheap metrics, works well in many cases.

4) When picking the distance metric, QD can always be carried out before conducting any data comparison. If two nuggets have huge query distance then the data comparison is on longer necessary, because the two nuggets will be surely dissimilar even without considering the data they contain. If two nuggets have a small query distance, we may choose to compute the data distance or not based on how "aggressive" we want to be in the process of nugget clustering. If we want to form big and rough clusters in our nugget pool with little time expense, we can just skip the data comparison and rely on the results of query comparison only. In contrast, if we need and can afford to be "careful" in nugget clustering and aim to form as precise clusters as possible, we have to consider data

distance, meaning that we need to pick QD+ETM (CC) or QD+ETM (HA). This choice can be made by comparing $ln(m)$ and n^2. If the former wins, we pick QD+ETM (CC), otherwise we pick QD+ETM (HA).

Chapter 4

Nugget Clustering

4.1 Iterative Incremental Clustering (IIC) Algorithm

As we mentioned earlier in Chapter 2, once we have an effective distance metric to capture the similarity between nuggets, any generic clustering algorithm can be applied here to conduct nugget clustering. However, we have to be careful about picking a suitable clustering algorithm, for the following reasons: 1) Considering that nuggets are generated during users' exploration, reclustering all the nuggets at the arrival of each new nugget may tent to be too expensive, especially when the nuggets pool is large. 2) Reclustering periodically (for example, once every day) will not only cost extra storage for nuggets until the nugget consolidation is under taken but also lose the opportunity to make real time modification to the nugget pool. Thus, we adopt an iterative incremental clustering (IIC) algorithm to realize high-quality clustering in real time. The main feature that distinguishes this algorithm from the traditional incremental clustering algorithms [12, 58] is that it recursively re-inserts the modified nugget back into the nuggets pool. This aims to avoid similar nuggets to be kept in the nugget pool (distances between all nuggets are larger than a threshold distance). Our proposed IIC algorithm is given in Figure 4.1.

IIC(P **,** N**):** //P is the current nugget pool
// N is the new nugget

1 AddToNuggetPool(N , P)
2 Return(P)

AddToNuggetPool(N **,** P**):**

1 $MinDistance = 1$
2 **for** each existing nugget N_j in $P\{$
3 **if (**$Distance(N_j, N) < MinDistance$**)**
4 $MinDistance$= $Distance(N_j, N)$
5 **int** $Closest$= j
6 **if (**$MinDistance < DistThreshold$**)**
7 **new nugget** N'=Combine($N, N_{Closest}$)
8 Remove $N_{Closest}$ from P
9 AddToNuggetsPool(N' , P)
10 else
11 Make N' a new nugget in nugget pool

Combine(Nx **,** Ny**):**

1 **new nugget** N
2 **for** each dimension k of N_x
//N_x and N_y have same dimensions
3 $N(k).b_l = \frac{N_x(k).b_l * N_x.vitality + N_y(k).b_l * N_y.vitality}{N_x.vitality + N_y.vitality}$
4 $N(k).b_h = \frac{N_x(k).b_h * N_x.vitality + N_y(k).b_h * N_y.vitality}{N_x.vitality + N_y.vitality}$
5 $N.vitality = N_x.vitality + N_y.vitality$
5 $N(k).b_l$ and $N(k).b_h$ are the lower bound and upper bound of nugget N on the kth dimension
6 **Return** N

Figure 4.1: Iterative-incremental clustering (IIC) algorithm

4.2 Evaluation to IIC

To learn the effectiveness of IIC, we carefully compare its performance with one of the more common non-incremental clustering algorithms, K-Means. First, we compare the theoretical characteristics of the two clustering algorithms.

	Incremental	Real Time	Parameter Required	Complexity
K-Means	No	No	K	$O(I*K*n)$
IIC	Yes	Yes	Distance Threshold	$O(K*n)$

K: number of clusters. I: number of iterations. n: number of objects

Figure 4.2: Theoretical characteristics of ICC and K-Means

As shown in Figure 4.2, ICC exhibits two noticeable advantages compared with K-Means. First, IIC is a real-time clustering algorithm, which incrementally clusters newly arriving objects into previously formed clusters. While, K-means would keep all the newly arriving objects and reclusters them whenever an output is needed. This real-time characteristics of ICC is important for our system, because it not only saves the storage for all the nuggets generated in history but also allows real-time modification of the nugget pool. Second, the complexity of the IIC is much lower than K-Means. Because K-Means relies on iterative reclustering of all objects, the number of iterations (I) may be very high in many cases. Thus, even if we give up the chance of real-time modification of the nugget pool, periodically reclustring the whole nugget pool with K-means is still quite time-consuming. For those two reasons, we choose ICC for our system.

We perform several experimental studies to compare both the efficiency and quality of ICC and K-means algorithms when they are used to cluster nuggets. We use ICC and K-Means to cluster three groups of nuggets. The 100 nuggets of the first group are extracted

from ”Iris” dataset. The second group has 500 nuggets extracted from ”Cars” dataset, and the third group is composed by 1000 nuggets extracted from ”Aaup” dataset. Our comparison strategy can be divided into three steps: **1)** For a selected group of nuggets, we run ICC first to cluster them and get the clustering results. **2)** We run K-Means against the same group of nuggets several times, but with different K (I.e, K= 1/2 k, K=k, K= 3/2 k, K= 2k, where k is the number of clusters just found by ICC). **3)** We compare both the execution time and the clustering quality of ICC and K-Means with different values of K.

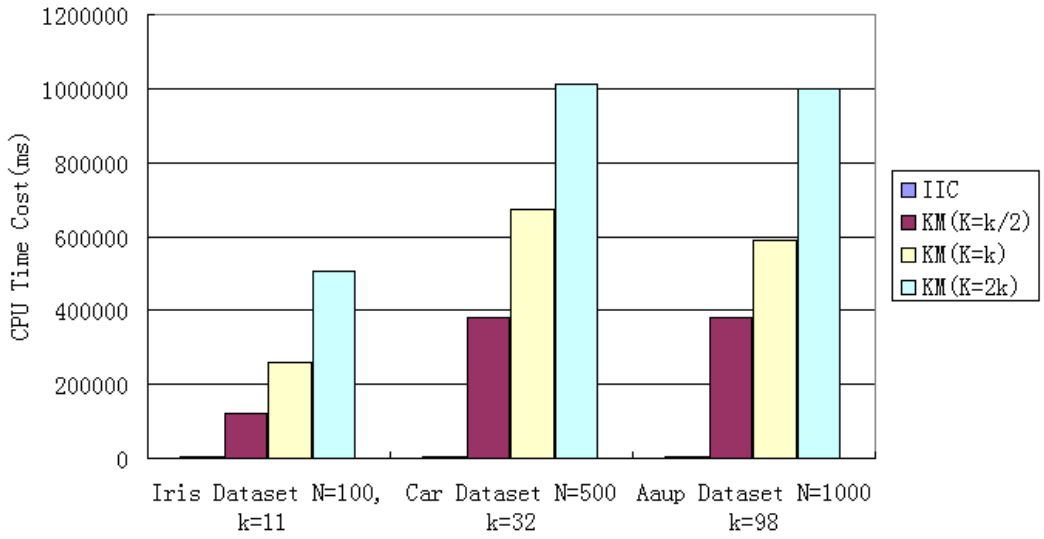

Figure 4.3: Comparison of CPU execution time by ICC and K-means with varying k

As shown in Figure 4.3, ICC is much more efficient than K-Means in terms of CPU time cost. Actually, ICC takes only 5 seconds to cluster 1000 nuggets. Thus, the time needed for clustering a newly arriving nugget is usually unnoticeable for users.

Now we discuss the clustering quality of two clustering algorithms. As mentioned in [25, 30], two major criteria are considered when measuring the quality of clusters. The first one is **“Compactness”:** This is a measure of cohesion or uniqueness of objects in an individual cluster with respect to the other objects outside the cluster, e.g., the average similarity of objects within the cluster. The greater the similarity, the greater the compact-

ness. The second one is **"Isolation":** This is a measure of distinctiveness or separation between a cluster and the rest of the world, e.g., the similarity between a cluster centriod to an object outside the cluster. The smaller the similarity, the greater the isolation. Thus, our cluster quality measure can be expressed by the sum of "compactness" and "isolation". Specifically, we use the accumulative distance between each cluster member and the cluster centroid to which it belongs to present the "compactness" of clusters. To present the "isolation", we use the "closeness penalty", which is given and accumulated if any nugget is too "close" to the cluster centroid it does not belong to. By adding up the accumulative distance and close penalty, we can express the "negative quality" of the clusters. By simply substract the negative quality from 1, we can finaly acquire the quality of clusters.

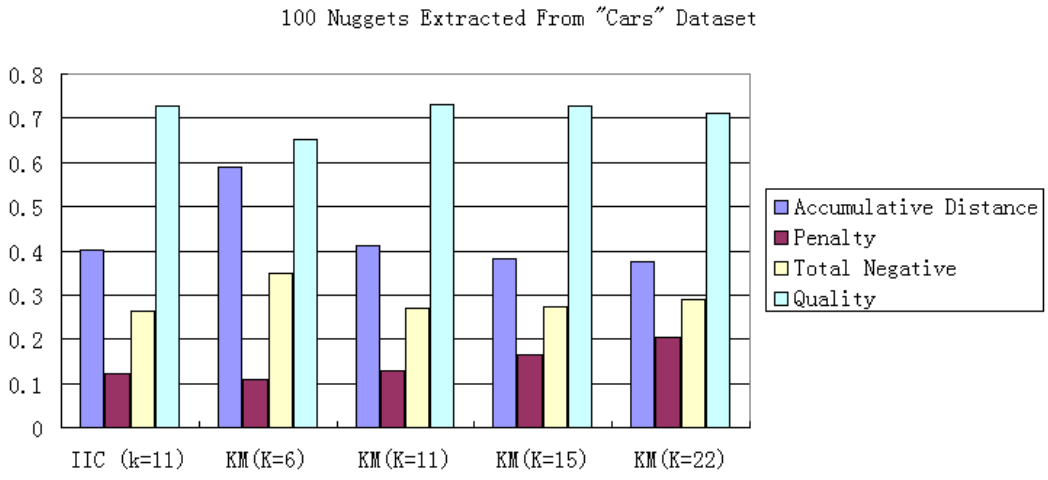

Figure 4.4: Comparison to quality of clusters (objects: 100 nuggets extracted for "Cars" dataset)

From Figures 4.4, 4.5 and 4.6, two facts can be observed: 1) When K=k, ICC has equivalent performance with K-Means in terms of quality of clusters. 2) When we decrease(half) or increase (double) the value of K, the quality of clusters degenerates. Thus, we learn that, for our observed cases, ICC can divide nuggets into proper number of clus-

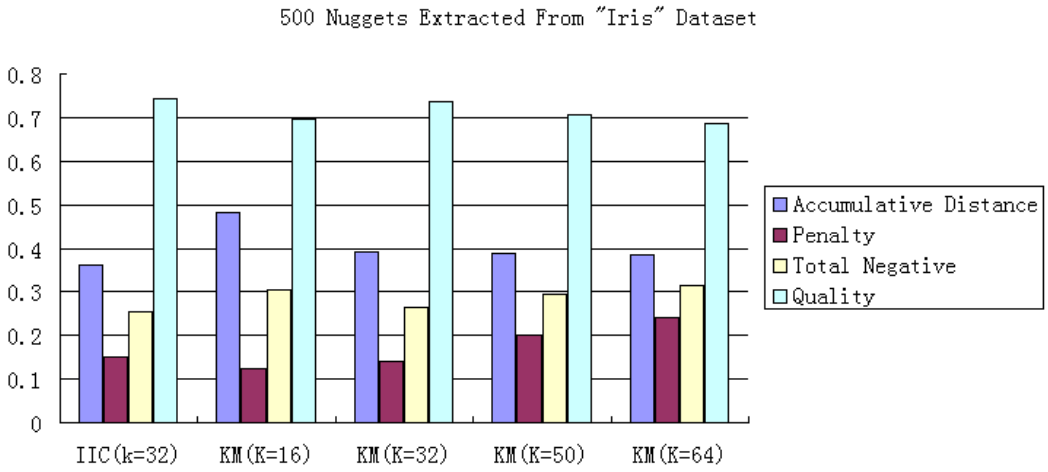

Figure 4.5: Comparison to quality of clusters (objects: 500 nuggets extracted for ”Iris” dataset)

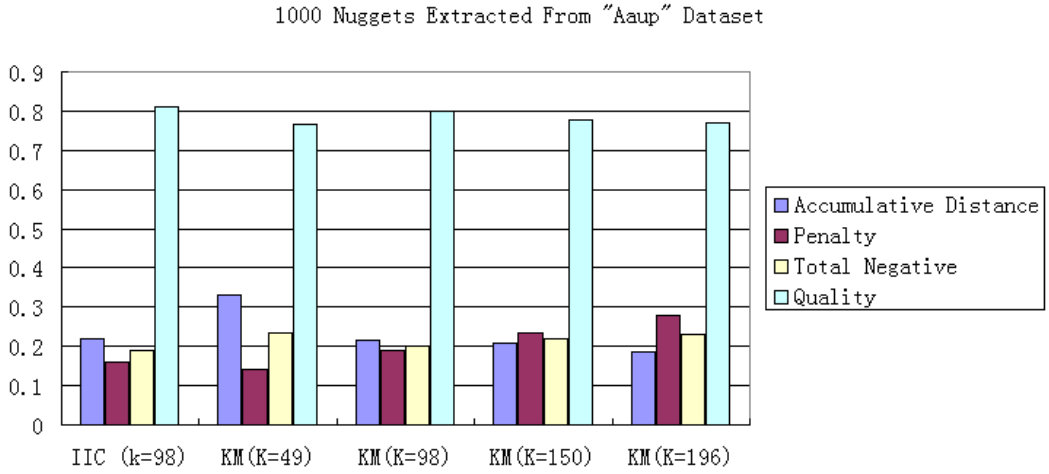

Figure 4.6: Comparison to quality of clusters (objects: 500 nuggets extracted for ”Aaup” dataset)

ters, and the quality of clusters it achieves is comparable with the K-Means. We thus conclude that the Iterative Incremental Clustering Algorithm is an effective method for nugget clustering, because it not only has the advantage of being incremental, and thus very efficient at run-time, but it also achieves comparable cluster quality with the non-incremental clustering algorithm.

Chapter 5

Nugget Refinement

5.1 Benefits from Nugget Refinement

In this section, we'll introduce our solution of using data mining techniques to refine the nuggets found from users' log. Such a refinement can be performed when a nugget was made because users were searching for some identifiable pattern types, such as clusters or outliers. For example, assume a user was searching for a cluster in the dataset, and for some reason, she missed part of it (Figure 5.1). Then, NMS will refine the nugget to capture the complete cluster (Figure 5.2).

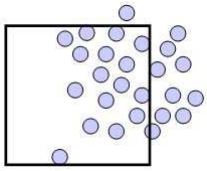

Figure 5.1: A nugget which captures the main body of a cluster bus misses part of it

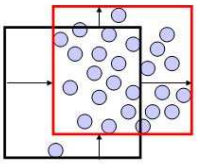

Figure 5.2: The refined nugget which captures the complete cluster

Nugget refinement offers several advantages over both pure log analysis or mining techniques of the data itself. They are: 1) Log analysis techniques, for example, the

nugget extraction introduced in Section 2, rely on users' actions only, without any help from computational analysis of the datasets and their properties. Thus they may lack accuracy in nugget specification. While nugget refinement guarantees the accuracy by exploiting both of them. 2) Data mining techniques, such as global pattern detection algorithms, need to be told the specific type of pattern that a user is looking for. Such information may not be always available, because users may not know the exact pattern types that are important to them. While for nuggets refinement, the users' interests have already been indicated by candidate nuggets, which are usually small subparts of a whole dataset. Thus the refinement process could run different local pattern detection algorithms to figure out what users were most-likely looking for. 3) Even assuming the system knows the specific pattern type a user is interested in, in many cases the user is not searching for all possible patterns but only for certain patterns of this type. This makes running expensive global pattern detection algorithms not always cost effective and unrelated patterns detected may even cost users more effort to isolate the useful ones. In this work, we focus on two important types of patterns, namely, clusers and outlies, and we chose density-based clustering [26] and distance- based outlier detection [39] as our sample pattern detection algorithms, which are popular algorithms extensively studied in the literature [28, 52]. However, other search methods from the literature could equally be used.

5.2 Techniques for Nugget Refinement

The refinement process is divided into two phases, called the match and refine phases.

Match phase

In this phase, we aim to match the identified nuggets with patterns "around them" within the data space. In other words, our goal is to determine which patterns users were searching for when these specific nuggets were made. In this work, we concentrate nuggets refinement on two important pattern types, clusters and outliers.

We first formally define the concept of "Match". The concept of "Match" is used to judge whether some data patterns or the major parts of these patterns primarily contribute to a nugget. If this is found to be the case, we call the nugget and these patterns "matched". The nuggets may be "matched" with more than one pattern. Or, put differently, a nugget may contain several patterns. Technically, to match a nugget with patterns, we have to compute two important factors that each represent one direction of the match:

- *Participation Rate* (PR) : A pattern P should be matched with a nugget N, only if most of its members, if not all, participate (are covered by) the nugget. For example, in Figure 5.1, for the cluster at the left side, data points 2, 3, 4, 5, 6 are covered by the nugget. So, we use PR to present how much of a pattern P is covered by a nugget N.

$$PR(N, P) = \frac{P.population \cap N.population}{P.population} \qquad (4)$$

- *Contribution Rate* (CR) : Since "match" is two-directional, while PR just expresses one direction, namely, nugget to pattern, we introduce CR to capture the opposite direction, from pattern to nugget. This shows how much a pattern or a partial pattern contributes to the nugget. Moreover, because a nugget is decided by a query and the results of this query (selected data), the notion of "contribution" here has a broader meaning than "covering" of population. Similar to the problem of measuring the distances between nuggets, we consider both the selected area and data population of the pattern and the nugget when calculating CR.

$$CR(N,P) = \frac{P.area \cap N.area}{2 * N.area} + \frac{P.population \cap N.population}{2 * N.population} (5)$$

Next we show a specific example of how to calculate PR and CR between a nugget and a cluster (the cluster on the left side in Figure 5.3).

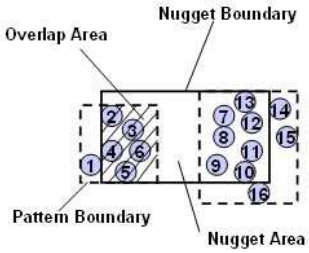

Figure 5.3: A nugget which captures the main bodies of two clusters

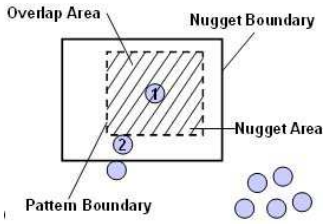

Figure 5.4: A nugget which includes an outlier (data point 1) and noise (data point 2)

The covered pattern population ($P.population \cap N.population$) equals 5 (containing data points 2, 3, 4, 5, 6), and the pattern population ($P.population$) equals 6. So $PR = 5/6 = 0.83$. The Nugget Area ($N.area$) in this example is the area denoted by the Nugget Boundary. The Pattern Area ($P.area$) is indicated by the Pattern Boundary. Overlap Area ($P.area \cap N.area$) is the overlap area depicted by the shaded area in the figure. Let's assume Overlap Area/Nugget Area=0.3. The concept of "Area" here extends to hypervolume when number of dimension increases. We also know that the Nugget Population equals 12. So CR= (0.3+5/12)/2=0.39

Now we use PR and CR to match a nugget with the patterns around it. We use $MatchRate(P,N)$ to express the result of a match between a nugget N and all patterns of type P that are covered or partially covered by N. Based on the match results, we classify nuggets into 3 different categories.

- *Clusters*

$$MatchRate(C,N) = \sum_{1 \le i \le n} PR(C_i,N) * CR(C_i,N) > T \ (6)$$

Where C_i's are all the cluster patterns covered or partially convered by the nugget. T is a threshold which decides whether the nugget and the patterns match. In this case, a nugget is matched with one or more clusters. In other words, the main components of this nugget are clusters.

- *Outliers*

  Although we still follow the notion of PR and CR as we did in the cluster cases, the way we calculate them is a little different. First, since an outlier pattern only has one data member, the PR for an outlier pattern is always 1. So we omit it from Formula 7. Second, the way we present pattern boundaries of outlier patterns is different also. As shown in Figure 5.4, the pattern area of an outlier is a (hyper) square area, where the distance from it to any boundary equals the maximum distance (among all the dimensions) between it to its nearest neighbor. All the other calculation processes remain the same as those for the cluster cases.

$$MatchRate(O, N) = \sum_{1 \leq i \leq n} CR(O_i, N) > T \qquad (7)$$

  Where O_i's are all the outlier patterns covered by the nugget. T is the same threshold we use in Formula 6. In this case, a nugget is matched with one or more outliers. In other words, the main components of this nugget are outliers.

- *No Specific Pattern*

  It is possible that a nugget will be matched with neither clusters nor outliers. In this case, the nugget belongs to the "No Specific Pattern" category. Expanding the library of recognized patterns is an important part of our future work.

An main assumption in this nugget-pattern-match solution is that Minimum Bounding Rectangle (MBR) is a suitable method to express the "spatial" characteristics of patterns.

Although MBR has been used in many previous work [50, 49] to capture the "spatial" characteristics of objects in multi-dimensional space, this may be a dangerous assumption when the patterns are in highly irregular shapes and the number of dimensions goes extremely high.

Refinement Phase

The match phase reveals to us what type of patterns that a user was searching for. If a nugget is classified into the first two categories mentioned above, we finish nugget refinement using the following two steps, called splitting (if necessary) and modification.

Splitting: If a nugget is judged to be "matched" with a certain type of patterns and it is composed of more than one pattern of this type, we could split it into several new nuggets, each representing one pattern of this type only. This process is straightforward, because we already know all patterns the users was searching for based on the knowledge learned from the match phase. The specific splitting process can be finished by putting all the members of each "qualified" pattern in the original nugget into a new nugget. The only thing we need to be careful about is that we only make new nuggets for those "qualified" patterns but not all those (partially) covered by original nugget. To be a "qualified pattern" in the original nugget N, a pattern P has to have $MR(P, N)$ greater than a certain threshold, which means the major population of this pattern has to be covered by the original nugget.

Modification: For the nuggets representing a single pattern only, the modification process becomes simple also, because we just need to make the nugget boundaries exactly the same as the pattern boundaries. Then the nuggets will become perfect representatives of the patterns. In Figures 5.5 and 5.6, we show the new nuggets after nugget refinement. Each now represents one pattern only.

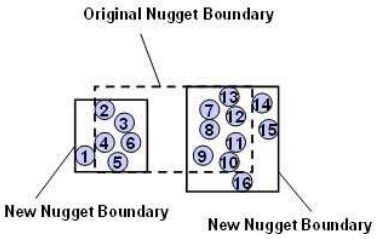

Figure 5.5: The refined nuggets which each capture a complete cluster

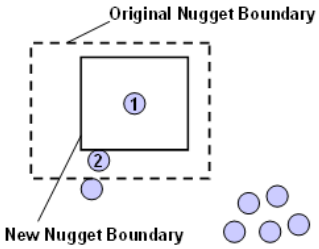

Figure 5.6: The refined nugget which includes one outlier only

Chapter 6

Users Study On Overall Functionalities of NMS

In order to show the effectiveness of NMS, we have performed user studies to compare users efficiency and accuracy when solving tasks with and without the help of NMS. Also, we have observed and analyzed the stability of NMS through our user studies.

6.1 Experimental Setup

Experimenal Environment: For all the user studies, we use an HP Pavilion laptop computer with 1.6MHz CPU and 512M memory. NMS is integrated in XmdvTool 7.0, which is the latest version of this multivariate visualization system [64].

Users: 17 subjects, all WPI students from various majors, participated in our user studies. All users came through a uniform training process. Those in the NMS group were also given the basic idea of how NMS works and made familiar with the interfaces of NMS.

Datasets: Three real datasets are employed in our user studies. They are the “Iris”

dataset (4 dimensions, 150 records); the "Cars" dataset (7 dimensions, 392 records); and the "Aaup" dataset (14 dimensions, 1161 records).

Tasks: Users are asked to finish five knowledge discovery tasks. Each of the tasks requires users to study a dataset and answer a question about the dataset. The specific tasks are:

Task 1: How many main clusters exist in the "car" dataset on four dimensions: MPG Cylinders, Horsepower and Origin?

Task 2: How many records in "Detroit" dataset satisfy the query: $ft_police < 274$, $unemp < 5.5$, $500 < manu - wrkrs < 600$, $handgun - lcs < 700$.

Task 3: In the Detroit dataset, for the record, which has highest value on $hundgan - lcs$, what are its corresponding values on dimension $ft - police$, $unemp$, and $manu - wrkrs$.

Task 4: In the Iris dataset, two subsets: subpart A: $4 < sepal - length < 6$, $2.8 < sepal - width < 4.5$, $1 < petal - length < 2$, and subpart B: $4 < sepal - length < 6$, $2.8 < sepal - width < 4.5$, $5 < petal - length < 6$, Which subparts has more records?

Task5: In the car dataset, which origin has highest average MPG?

All those five tasks were printed out on a single-page task sheet and given to users when they were ready to solve the tasks.

6.2 Experimental Methodology

In this user study, we appointed time with each individual user based on our user grouping results, which will be discussed later in the section. During each appointed time, one user participated in our user study. Users were not allowed to communicate information about the user study through any other channel except NMS at any time before, during, or after the user study. This is to make sure that users can only solve the tasks based on their

own exploration and the help from NMS (if available). A uniform training process was designed to give the basic idea of how NMS works and made familiar with the interfaces of NMS. This training process was given by a same user study host, namely, the author of this thesis, to each individual user before they started to solve the tasks. Although this training process was not given once to all the users in a lecture manner, since it was well prepared and carried out by a same user study host, we believe that it was helpful and fair for every user. The training process took 10 to 15 minutes for different users, including question-raising and -answering time. Once the users felt ready to start solving the tasks, we gave them the task sheet and started timing for their task-solving process. By then, they were no longer allowed to communicate with the user study host. All the users were encouraged to finish the tasks as quickly and correctly as possible, but there's no time limit for them to finish the tasks. Users were asksed to finish the tasks in the same sequence as the tasks appeared on the task sheet. They handed up when finishing each of the individual task and wrote down the answer on an answer sheet. The user study host collected all the answer sheets and all the time records for the final analysis.

6.3 Users' Time Efficiency

To compare users' efficiency of finishing this given set of tasks with and without help of NMS, we randomly divide the twelve users into four groups, three per group. Each user is asked to finish the same five tasks. Among these 4 groups, users of group 1 were asked to finish the tasks without NMS, while the other three groups (2- 4) were supported by NMS. Members of each group were randomly given a sequence number ranging from 1 to 3, which represents the user's sequence of solving the problems in his/her group. For example, once a user from group 1 receives the sequence number 2, he/she will be the second one in group 1 to finish the tasks. Group members were encouraged to use

the functionalities of NMS (if available) to manage and share their discoveries with later users. Figure 6.1 shows the time used by each user and group to finish the tasks. As

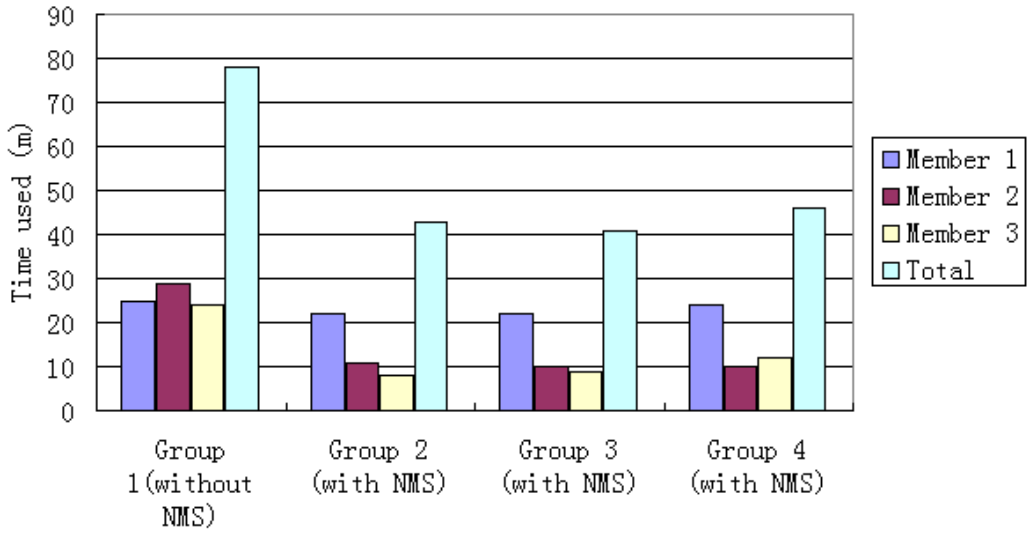

Figure 6.1: Comparison of users' efficiency in different groups

shown in Figure 6.1, groups 2, 3, and 4 (with NMS) spent noticeable less time (around 50 percent) than group 1 (without NMS). Such time savings due to the second and the third users, given that the first users all worked from scratch. Although NMS did facilitate their job, managing discoveries needed time. However, once the nuggets were extracted during the exploration by the first users, the exploration processes of the second and the third users largely benefited from the nugget pool.

To better support our analysis, we compared the time used by six users working from scratch (three members of group 1, and three first users of each other groups) and by the other six users working with guidance of the nugget pool (the second and third users of groups 2,3 and 4). Figure 6.2 shows that the later six users with guidance of nugget pool were working much more efficiently. Specifically, the minimum, maximum, and the average time spent by these users are all much less than those who worked from scratch.

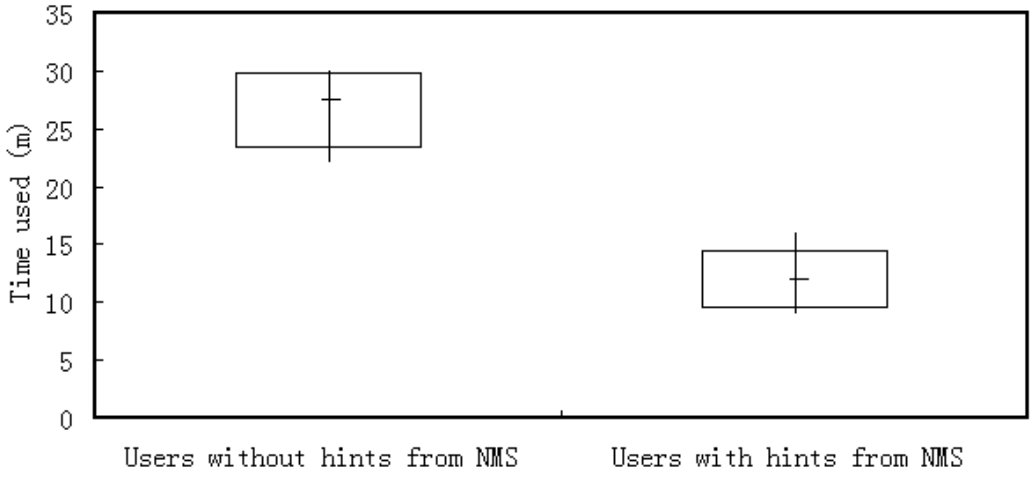

Figure 6.2: Comparison of users' efficiency with and without guidance from nugget pool

The standard deviation is lower too.

6.4 Accuracy of Accomplished Tasks

We also studied the effect of NMS on the accuracy of the accomplished tasks. Among the five tasks we mentioned earlier, tasks 2 to 5 are straightforward problems. Users gave correct answers, although spending different amounts of time on them. However, task 1 is a common but complex knowledge discovery problem. Since not all the clusters can be easily found, in our user study the answer provided by the users varied. Figure 6.3 shows the number of clusters found by each user and also the number of clusters that actually exist. Two facts can be observed from Figure 6.3: 1) The number of clusters correctly found by users working with NMS are generally closer to the number of actual clusters. 2) The later users in each group are more likely to find all existing nuggets compared to the earlier ones. These two facts show the promise of NMS indeed improving the accuracy of the tasks accomplished by the users.

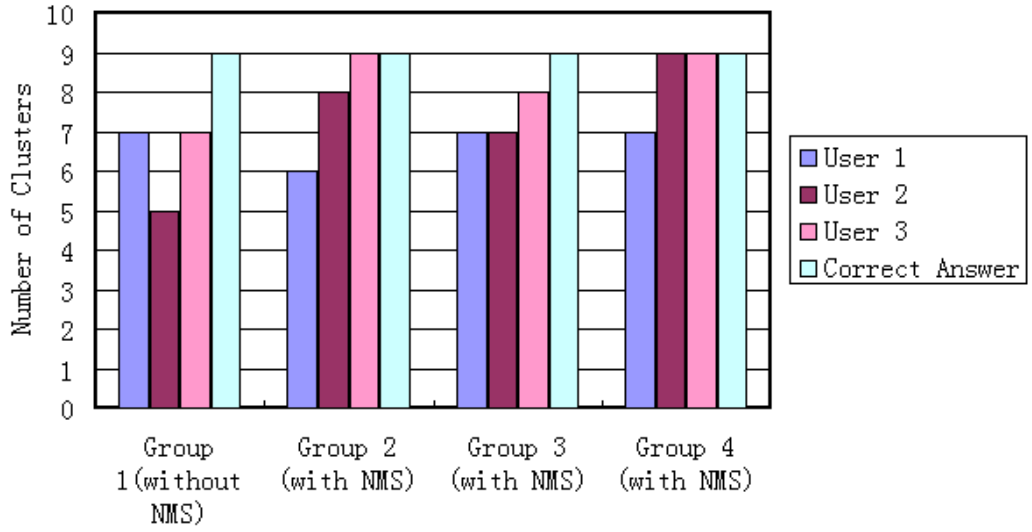

Figure 6.3: Comparison of users' accuracy of finishing complex task with and without help from NMS

6.5 Stability of NMS

Lastly, we consider NMS's stability, meaning how well it performs after long-term use. To give a preliminary evaluation of this, we asked a 7-user group to solve the five tasks mentioned earlier. We analyzed the population of the nugget pool after use by each user. The change of the nugget pool population is shown in Figure 6.4. The comparison of average number of nuggets identified by the 7 users for each task and the estimated number of nuggets needed for each task is shown in Figure 6.3. The "Estimated" number of nuggets is given based on our own experience of how many nuggets are needed for each task. From Figures 6.3 and 6.4, we observe two facts: One, the average number of nuggets formed by each user for each question is generally a little higher than estimated (Figure 6.5). Two, the populations of the nugget pools are relatively stable during users' exploration (Figure 6.4). Thus although some useless nuggets may be generated during exploration, they are usually only small portions of the whole nugget pool. Thus, this is

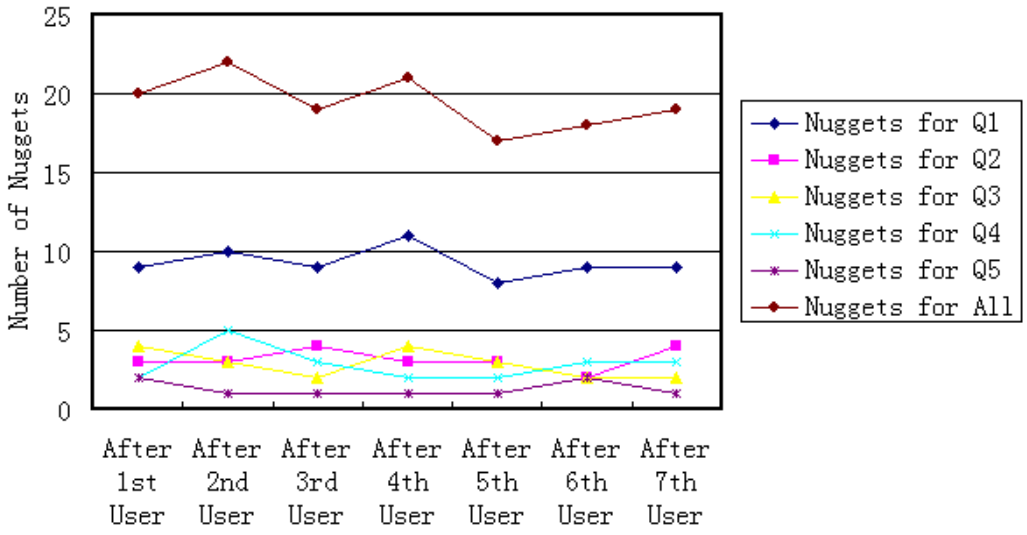

Figure 6.4: Evolution of nugget populations over time

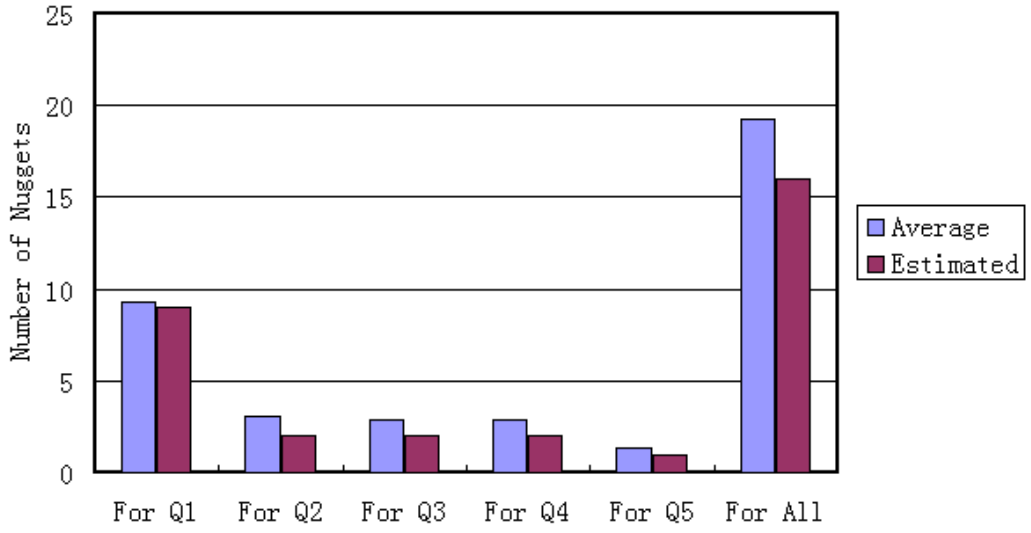

Figure 6.5: Comparsion of numbers of nuggets generated by users and those are estimated

indication that the population of the nugget pool is not likely to degenerate dramatically during a long-term use.

Chapter 7

Related Work

Illuminating the Path [61], the research and development agenda for visual analytics published by National Visualization and Analytics Center, points out that visual analytics is the science of analytical reasoning facilitated by interactive visual interfaces. People use visual analytics tools and techniques to synthesize information and derive insights from different types of data; provide understandable assessments; and communicate assessment effectively for action. This agenda also lists the four major focus areas of visual analytics, which are: 1) Analytical Reasoning Techniques; 2) Visual Presentations and Interaction Techniques; 3) Data Representations and Transformations and 4) Production, Presentation, and Dissemination. As a analysis-guided framework for visual exploration, our NMS framework mainly falls into the first and the second area, which are Analytical Reasoning Techniques and Visual Presentations and Interaction Techniques. And it is also related to other two areas, where many important ideas can be borrowed while also expanded to serve our general framework.

7.1 Analytical Reasoning Techniques

As described in [14, 40, 16], analytical reasoning usually follows a sense-making loop, which is iterative and time-consuming. Thus, two of the key goals need to be achieved by analytical reasoning techniques are: 1) Enabling users to isolate valuable information from massive amount of raw data. 2) Providing discovery management mechanisms which allow users to organize their temporary achievements and finally develop insights that directly support assessment, planning, and decision making. The efforts to achieve the first goal can be classified into three categories, namely, User-Driven, Data-Driven and Hybrid.

In the User-Driven category, the knowledge discovery process relies on users perceptual power, while a variety of visual interaction mechanisms [4, 64, 18, 3] such as brushing, and dynamic querying, are offered by the visualization systems to help them isolate the valuable information. Such techniques work well in many cases, compared with those explorations without any interaction. While they still suffer from two disadvantages. First, they may be inefficient, because users' effort to locate valuable information generally follows a"try-and-see" manner. When the information items need to be isolated have a huge population or are not easy to identify by human vision, using such techniques may lead to large investment of users' time. Second, because the whole isolation process relies on users' perceptual abilities only, the valuable information found by users may lack accuracy. This problem has been discussed in the Chapter 5 of this thesis. Thus, in our framework, we have applied these techniques to allow users to best use their perceptual power during visual exploration, while we have introduced computational analysis in the different stages to improve the accuracy of users' discoveries.

Data-driven techniques aim to expediate knowledge discovery with the help of the analytical power of machines. Data mining algorithms [29, 68, 38, 26, 52, 12], which detect

useful patterns or rules in large datasets, fulfill an important role here. These techniques are independent from the visual environment and thus do not bring into play the power of the human intuition in the pattern detection process. The main limitation of these techniques is that they usually aim to find all the members of a certain pattern type and thus ignore the interests of users. In many cases the user is not searching for all possible patterns of a certain type but only for a subset of them. For example, for a dataset with 100 dimensions, a user may not be interested in all the clusters existing among all these 100 dimensions. Rather, she may just care about the clusters existing in a small subset of dimensions and even small subranges on these dimensions. This makes running expensive global pattern detection algorithms not cost effective and unrelated patterns detected may even cost users more effort to isolate the useful ones.

The third category is hybrid. The initial effort in this category is called visual data mining (VDM) [35, 22, 34]. It involves users in the mining process itself, rather than being carried out completely by machines. In VDM, visualizations are utilized to support a specific mining task or display the results of a mining algorithm, such as association rule mining, and thus enhance user comprehension of the results. However, these techniques still suffer from a major problem. That is they still require users to provide the specific pattern types they are looking for before the detection process can begin. Such information, however, may not be always available, because users may not know the exact pattern types that are important to them. In our NMS framework, we address this problem by introducing a "match" phase in our nugget refinement stage.

Providing a management mechanism to users' discoveries is equally important to the isolation process. Although general knowledge management mechanisms have been studied in the literature [9, 41, 36], few attention has been paid to those closely integrated into visualization systems. [31] provided some recording functions to its users during the KDD process, while its main focus is on visualizing the KDD process itself. [21] pro-

posed interactive tools to manage both the existing information and the synthesis of new analytic knowledge for sense- making in visualization systems. This work so far has not paid much attention on how to consolidate the users discoveries. However, consolidation to users' discovery by computational power is one of main contributions of my work. Collaborative visual analytics [51] introduced computational power into the sense-making process with focus on supporting the exchange of information among team members.

7.2 Visual Presentations and Interaction Techniques

Visualization systems traditionally focus on building graphical depictions of relationships among information in a human comprehensible format. By doing so, they usually help their users to better understand the information. This means the users can either learn some facts that are not easy to discover without the graphical depiction, or the users' knowledge to some facts can become deeper or more precise. The usefulness of visualization systems has been well established [64, 55, 56, 23]. Although visual presentation techniques are not the focus of this work, as a framework that aims to provide useful guidance to users, NMS exploits interaction [13, 14] techniques between system and users. Our framework are using two important types of interaction techniques.

Filtering and navigation techniques [4, 64, 18, 3] allow users to search and focus their attention on the information items that are of their interests. So, they are the basis of our nugget extraction. More sophisticated filtering and navigation techniques will provide us new opportunities to extend the range of nugget types in our future work.

As we will use the nugget pool to guide users' exploration, when and how to present our guidance information will also be a challenging problem for our future work. Previous work looked at this problem mainly belong to recommendation systems [37, 24] and HCI community [57, 13]. Recommendation systems cluster users into different groups based

on their profile. Then within the same groups, recommendation systems recommend a user the information items that have received attention from other uses but not from herself yet. The key challenge of using this basic idea to our framework is that, traditional recommendation systems are mainly working in discrete domain, which mains judging the similarity between the information items is usually straightforward. However in our system, judging the similarity between the discoveries found users is a very challenging job. Although we have given initial solution to this problem in Chapter 4 of this work, as the new types of nuggets must appears in future, we need to learn more general solutions to scale the distances between the nuggets. HCI itself as a broad research direction focus on effective information exchange between humans and computers. By borrowing and developing the ideas from these areas, we expect to build guidance mechanisms based the nugget pool, which effectively communicates useful information with users while brings as few unnecessary interruptions as possible.

7.3 Data Representations and Transformation

As the result of knowledge crystallization, nuggets are important carriers of user's insights to the datasets. How to best encapsulate different types of information about a same pattern into a nugget is an important Data Representations and Transformation problem. [27] studied on the metadata which provide structures for a global information space that lends context to support multi-type analysis. Besides the individual nuggets, interrelations among nuggets is another important aspect that we have to explore. These interrelations can be hierarchical structures among nuggets, for example, some nuggets may be subparts of a bigger nugget. Interrelations among nuggets can also be parallel, for example, several nuggets may all be the evidences for a single knowledge.

7.4 Production, Presentation and Dissemination

These techniques support production, presentation and dissemination of the results of an analysis to communicate information in the appropriate context to a variety of audiences. Same as other analytical systems, the ultimate goal of our framework is to make the nuggets we learned available and understandable to users with different background. So we have to keep our eyes on these important techniques also. Previous work in this area includes, [53] , which allows commanders from battalion level and higher to feed real-time situational awareness into the system and have that information available in text and graphic representation immediately by fellow commanders and operations officers at all level; [66, 33] which allow analyst to organize and work with evidence from multiple perspectives simultaneously. Besides those techniques which mainly aim to help professional analysts work better, animation techniques [43, 7] are another important aspect which lets audiences with less proficiency understand the result of analysis better. We have to study these two aspects to make both the “production” and “consumption” to the nuggets smoother.

Chapter 8

Conclusions and Future Work

8.1 Conclusion

In this paper, we introduce a framework for analysis-guided visual exploration of multivariate data. Our system (NMS) leverages the collaborative effort of human intuition and machine computations to extract, combine, refine and maintain the valuable information hidden in large datasets. Finally, a well-organized nugget pool can be used to guide users exploration. Our preliminary evaluations indicate that NMS may greatly improve users time efficiency when solving knowledge discovery tasks. It may also be able to enhance users accuracy of finishing these tasks, although more complicated tasks are needed to validate this. Lastly, NMS works in a stable manner during explorations by a sequence of users. This shows its promise of working well during long-term exploration. More comprehensive user studies which involve more users and more complex tasks will be one of the directions for our future work.

8.2 Future Work

Although we have created a general framework for analysis-guided visual exploration of multivariate data and initially studied on each of the stage in this framework, this thesis is just the beginning but not end of many important work. For the nugget extraction, a more extensive range of nugget types will be considered in our future work. This will go hand in hand with new filtering and navigation techniques. For the nugget refinement part, expanding the library of recognized patterns is an important task for our future work. Besides the extension work to these two parts, the focus of our future work will mainly be concentrated on nugget maintenance and nugget guided exploration. As we mentioned earlier in each of these two chapters, we will study how to give proper rights to multiple users working on the same nugget pool and How to automatically learn and modify the parameters controlling nugget pool evolution during users exploration, for nugget maintenance. And for the nugget guided exploration, future work includes: 1) How to build hierarchical structures among nuggets based on their interrelation. (i.e., some nuggets may be subparts of a bigger nugget) 2) How to organize a knowledge-driven nugget pool [21] .(i.e., several nuggets may all be the evidences of a single knowledge, thus should be grouped together) 3) How to guide users based on their profiles using collaborative filtering techniques. [15] 4) How to extract story lines from a dataset and use animation techniques to feed back to users. (i.e., there may be some certain visiting sequence among the nuggets that best reveal some knowledge.)

Bibliography

[1] H. Abdi. Distance. In *N.J. Salkind (Ed.): Encyclopedia of Measurement and Statistics*, 2007.

[2] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu. A framework for clustering evolving data streams. In *VLDB*, pages 81–92, 2003.

[3] C. Ahlberg and B. Shneiderman. Visual information seeking: tight coupling of dynamic query filters with starfield displays. In *CHI Conference Companion*, page 222, 1994.

[4] A.Inselberg. Multidimensional detective. *Proc. of IEEE InfoVis*, pages 100–107, 1997.

[5] P. Andreae, B. Dawkins, and P. O'Connor. Dysect: An incremental clustering algorithm. *Document included with public-domain version of the software, retrieved from Statlib at CMU*, 1990.

[6] M. Ankerst, S. Berchtold, and D. A. Keim. Similarity clustering of dimensions for an enhanced visualization of multidimensional data. *Proc. of IEEE Symposium on Information Visualization, InfoVis'98, p. 52-60*, 1998.

[7] A. Arya. Content description for face animation. In *Encyclopedia of Information Science and Technology (I)*, pages 546–549. 2005.

[8] B. Babcock, S. Chaudhuri, and G. Das. Dynamic sample selection for approximate query processing. In *ACM SIGMOD International Conference on Management of Data*, pages 539–550, 2003.

[9] I. Becerra-Fernandez and D. Aha. Case-based problem solving for knowledge management systems, 1999.

[10] M. R. Bolin and G. W. Meyer. A perceptually based adaptive sampling algorithm. In *SIGGRAPH*, pages 299–309, 1998.

[11] J. Borges and M. Levene. Data mining of user navigation patterns. In *WEBKDD*, pages 92–111, 1999.

[12] F. Can. Incremental clustering for dynamic information processing. *ACM Trans. Inf. Syst.*, 11(2):143–164, 1993.

[13] S. Card, T. Moran, and A. Newell. *The Psychology of Human-Computer Interaction*. Lawrence Erlbaum Associates, 1983.

[14] S. K. Card, J. Mackinlay, and B. Shneiderman. *Readings in Information Visualization: Using Vision o Think*. Morgan Kaufman, 1999.

[15] Z. Chedrawy and S. S. R. Abidi. An item-based collaborative filtering framework featuring case based reasoning. In *IC-AI*, pages 286–292, 2005.

[16] CIA. *P1000 Strategic Plan for Information Visualization*. CIA/AIPASG, 1995.

[17] M. Claypool, P. Le, M. Wased, and D. Brown. Implicit interest indicators. In *Intelligent User Interfaces*, pages 33–40, 2001.

[18] W. Cleveland and M.McGill. Graphical perception: Theory, experimentation and application to the development of graphical methods. *Journal of the American Statistical Association*, 79:531–554, 1984.

[19] G. Cobena, S. Abiteboul, and A. Marian. Detecting changes in xml documents. In *ICDE*, pages 41–52, 2002.

[20] Q. Cui, M. O. Ward, E. A. Rundensteiner, and J. Yang. Measuring data abstraction quality in multiresolution visualization. *IEEE InfoVis*, pages 183–190, 2006.

[21] G. D, Z. M.X, and A. V. Interactive visual synthesis of analytic knowledge. *IEEE VAST*, pages 51–58, 2006.

[22] M. C. F. de Oliveira and H. Levkowitz. From visual data exploration to visual data mining: A survey. *IEEE Trans. on Vis. and Computer Graphics*, 9(3):378–394, 2003.

[23] Deborah F. Swayne, Dianne Cook and A. Buja. XGobi: Interactive dynamic data visualization in the X Window System. *Journal of Computational and Graphical Statistics*, 7(1):113–130, 1998.

[24] F. H. del Olmo, E. Gaudioso, and J. Boticario. A reinforcement learning approach to achieve unobtrusive and interactive recommendation systems for web-based communities. In *AH*, pages 409–412, 2004.

[25] R. Dubes and A. K. Jain. Validity studies in clustering methodologies. *In Pattern Recognition*, page 11, 1979.

[26] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *KDD*, pages 226–231, 1996.

[27] R. Gentleman. Using go for statistical analysis. *In proceedings of the 16th COMPSTAT Conference*, pages 171–180, 2004.

[28] M. Gorawski and R. Malczok. Aec algorithm: A heuristic approach to calculating density-based clustering *ps* parameter. In *ADVIS*, pages 90–99, 2006.

[29] S. Guha, R. Rastogi, and K. Shim. CURE: an efficient clustering algorithm for large databases. In *ACM SIGMOD*, pages 73–84.

[30] S. Guha, R. Rastogi, and K. Shim. Cure: an efficient clustering algorithm for large databases. *SIGMOD Record, vol.27(2), p. 73-84*, June 1998.

[31] J. Han, X. Hu, and N. Crecone. A visualization model of interactive knowledge discoverys systems and its implementations. *Information Visualization.*, 2(2):105–125, 2003.

[32] A. Inselberg and B. Dimsdale. Parallel coordinates: A tool for visualizing multidimensional geometry. *Proc. of Visualization '90, p. 361-78*, 1990.

[33] D. Jonker, W. Wright, D. Schroh, P. Proulx, and B. Cort. Information triage with trist. *2005 International Conference on Intelligence Analysis*, 2005.

[34] D. A. Keim. Information visualization and visual data mining. *IEEE Trans. on Vis. and Computer Graphics*, 7(1):100–107, 2002.

[35] D. A. Keim and H.-P. Kriegel. Visualization techniques for mining large databases: A comparison. *Transactions on Knowledge and Data Engineering, Special Issue on Data Mining*, 8(6):923–938, 1996.

[36] E. L. Kelly. In search of a new generation of knowledge management applications.

[37] T.-H. Kim and S.-B. Yang. An effective recommendation algorithm for clustering-based recommender systems. In *Australian Conference on Artificial Intelligence*, pages 1150–1153, 2005.

[38] E. M. Knorr and R. T. Ng. Algorithms for mining distance-based outliers in large datasets. In *Proc. VLDB*, pages 392–403, 1998.

[39] E. M. Knorr, R. T. Ng, and V. Tucakov. Distance-based outliers: Algorithms and applications. *VLDB J.*, 8(3-4):237–253, 2000.

[40] J. Lederberg. Preface:twelve-step process for scientific experiments: Epicycles of scientific discovery. In *Exitement and Fascination of Science*. 1989.

[41] M. Liao, A. Abecker, A. Bernardi, K. Hinkelmann, and M. Sintek. Ontologies for knowledge retrieval in organizational memories, 1999.

[42] J. Lin and B. Katz. Question answering from the web using knowledge annotation and knowledge mining techniques clustering,. In *Proc. CIKM*, pages 116–123, 2003.

[43] N. Magnenat-Thalmann and D. Thalmann. Computer animation. In *The Computer Science and Engineering Handbook*, pages 1300–1318. 1997.

[44] J. Melton. The sql language: A case study. In *The Computer Science and Engineering Handbook*, pages 1171–1189. 1997.

[45] J.-E. Michels, K. G. Kulkarni, C. M. Farrar, A. Eisenberg, N. M. Mattos, and H. Darwen. The sql standard. *it - Information Technology*, 45(1):30–38, 2003.

[46] J. Munkres. Algorithms for the assignment and transportation problems. *Journal of the Society of Industrial and Applied Mathematics*, 37(1).

[47] W. Niblack, R. Barber, W. Equitz, M. Flickner, E. Glasman, D. Petkovic, P. Yanker, C. Faloutsos, and G. Taubin. The qbic project: querying images by content using color, texture and shape. In *Proc. of SPIE Storage and Retrieval for Image and Video Databases*, pages 173–187, 1993.

[48] C.-C. Pan, K.-H. Yang, and T.-L. Lee. Approximate string matching in ldap based on edit distance. In *IPDPS*, 2002.

[49] D. Papadias and Y. Theodoridis. Spatial relations, minimum bounding rectangles, and spatial data structures. *International Journal of Geographical Information Science*, 11(2):111–138, 1997.

[50] D. Papadias, Y. Theodoridis, T. K. Sellis, and M. J. Egenhofer. Topological relations in the world of minimum bounding rectangles: A study with r-trees. In *SIGMOD Conference*, pages 92–103, 1995.

[51] K. P.E. Collaborative visual analytics: Inferring from the spatial organization and collaborative use of information. *IEEE VAST*, pages 137–144, 2006.

[52] D. Ren, K. Scott, B. Wang, and W. Perrizo. A distance-based outlier detection method using p-tree. In *CAINE*, pages 160–164, 2004.

[53] K. Rhem. *Division Uses 'Command Post of Future'*. American Forces Press Service, 2004.

[54] E. A. Riskin. Optimal bit allocation via the generalized bfos algorithm. *IEEE Transactions on Information Theory*, 37(2):400–, 1991.

[55] S.Harve, E.Hetzler, K.Perrine, E.Jurrus, and N.Miller. Interactive visualization of multiple query results. *Proc. of IEEE InfoVis*, 2001.

[56] B. Shneiderman. Tree visualization with tree-maps: A 2d space-filling approach. *ACM Transactions on Graphics, Vol. 11(1), p. 92-99*, Jan. 1992.

[57] S. B. Shum and N. Hammond. Transferring hci modelling and design techniques to practitioners: A framework and empirical work. In *BCS HCI*, pages 21–36, 1994.

[58] G. Somlo and A. E. Howe. Incremental clustering for profile maintenance in information gathering web agents. In *Agents*, pages 262–269, 2001.

[59] M. Swain and D. Ballard. Color indexing. *International Journal of Computer Vision*, page 11C32, 1991.

[60] D. R. Tauritz and I. G. Sprinkhuizen-Kuyper. Adaptive information filtering algorithms. In *IDA*, pages 513–524, 1999.

[61] J. J. Thomas and K. A. Cook. *Illuminating the Path: The Research and Development Agenda for Visual Analytics*. IEEE Computer Society, Los Alamitos CA, 2005.

[62] K. Tranbarger and F. P. Schoenberg. The hungarian method for the assignment problem. *Naval Research Logistic Quarterly*, (2).

[63] K. Tranbarger and F. P. Schoenberg. On the computation and application of prototype point patterns. *Department of Statistics, UCLA. Department of Statistics Papers*, 2004.

[64] M. Ward. Xmdvtool: Integrating multiple methods for visualizing multivariate data. *Proc. of Visualization '94, p. 326-33*, 1994.

[65] J. Wen, J. Nie, and H. Zhang. Clustering user queries of a search engine. In *World Wide Web*, pages 162–168, 2001.

[66] W. Wright, D. Schroh, P. Proulx, B. Cort, and D. Jonker. *Advances in nSpace-The Sandbox for Analysis*. Poster at Conference on Intelligence Analysis, 2005.

[67] G. Xue, H. Zeng, Z. Chen, W. Ma, and Y. Yu. Clustering user queries of a search engine. In *Proc of the European Conference on Information Retrieval*, pages 330–344, 2005.

[68] T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: an efficient data clustering method for very large databases. In *ACM SIGMOD*, pages 103–114.

[69] T. Zhang, R. Ramakrishnan, and M. Livny. Birch: an efficient data clustering method for very large databases. *SIGMOD Record, vol.25(2), p. 103-14*, June 1996.