Nugget Discovery in Visual Exploration Environments by Query Consolidation [*]

Di Yang
Worcester Polytechnic Institute
Worcester, MA, USA
diyang@wpi.edu

Elke A. Rundensteiner
Worcester Polytechnic Institute
Worcester, MA, USA
rundens@wpi.edu

Matthew O. Ward
Worcester Polytechnic Institute
Worcester, MA, USA
matt@cs.wpi.edu

ABSTRACT

Queries issued by casual users or specialists exploring a data set often point us to important subsets of the data, be it clusters, outliers or other features of particular importance. Capturing and caching such queries (henceforth called nuggets) has many potential benefits, including the optimization of both the performance of the underlying system as well as the search experience of users. Unfortunately, current visual exploration systems, while facilitating data exploration by providing graphical depictions of the data, have not yet tapped into this potential resource of identifying and sharing important queries. In this paper, we introduce a query consolidation strategy aimed at solving the general problem of isolating important queries from the potentially huge amount of queries submitted. Particularly, our solution clusters redundant queries caused by exploration-style query specification, which is prevalent in data exploration systems. Then, it generates a representative for each group of clustered queries. To measure the similarity between queries, we design a highly effective distance metric that integrates both the query specification and the actual query result. To overcome its high time complexity when comparing queries with large result sets, we also design an approximation method, which offers high time efficiency while still providing excellent accuracy. A user study conducted on real multivariate data sets comparing our proposed technique to others in the literature confirms that the proposed distance metric indeed matches well with users' intuition. As proof of feasibility, we develop a prototype Nugget Management System (NMS) based on our proposed query consolidation solution. A second user study comparing a freeware visual exploration system XMDVTool, both with and without being supplemented by NMS, indicates that both the efficiency and accuracy of users' visual exploration are indeed enhanced when supported by our technology.

[*]This work is supported under NSF grant IIS-0119276 and a grant from the NSA.

Keywords

Query redundancy, query consolidation, distance metrics

1. INTRODUCTION

Learning the important queries, in which users are interested, is a critical problem for information and knowledge management community. Such queries may be good indicators of users' common interest in certain portions of the data, or reflect "insights" from experts about the noteworthy features in the data. Generally, figuring out these important queries could benefit both the underlying systems and their users. On one hand, knowing the potential queries that users may raise in the future enables caching and optimized execution of incoming queries [13, 20]. By reusing the results of important queries, query caching techniques can save both the query execution time and data transmission cost. On the other hand, being informed about the important queries raised by other users may be helpful for the users. I.e., a basketball fan who is searching for sports equipment from an Internet search engine could benefit from knowing that the current most popular query for sports equipment is about: "T-MAC 6", the latest version of a series of basketball shoes. Efforts to provide users such help are called collaborative querying [10, 18]. Collaborative querying aims to help users to formulate queries by harnessing the collective knowledge of other searchers.

Visualization systems [23, 19, 9] traditionally facilitate knowledge discovery by conveying the information to users via graphical depictions. They have paid, however, little, if any, attention to the potential gains for visual exploration by identifying and exploiting important queries. To symbolize the value of such important queries, we call them "nuggets" [1]. Clearly, technology must be developed to extract and utilize these important queries submitted to visualization systems. Here, we focus on solving a critical problem for isolating "nuggets" from the potentially huge volume of queries submitted during users' exploration.

Query redundancy arises as users with common interest may submit queries with similar but not identical specifications. For example, users interested in the "T-MAC 6" basketball shoes, may submit their queries to a search engine with different specifications, such as "T-MAC 6 basketball shoes", "T-MAC 6 shoes" or "T-MAC 6 shoes for basketball". Query redundancy in visual exploration systems tends to be caused by their exploration-style query specifi-

[1]We use the term "nugget" and "important queries" interchangeably in the remainder of this paper.

cation mechanism, rather than by ambiguous language and semantics (demonstrated by the previous example in this paragraph). This is because their users utilize visual interaction techniques, such as sliders or selecting windows, to specify the queries that roughly express their interest, rather than explicitly specifying exact queries as typically in SQL-style query systems. Users are more interested in the query results fed back by the systems, rather than the exact specification of particular queries. For example, a students at WPI may search local restaurants through a life-guide system by moving sliders that specify the distance and price. Then, very similar queries, such as "in 5 miles, 20 - 30$ per person" , "in 5.3 miles, 23 - 32$ per person" or "in 5.5 miles, 18 - 31$ per person", may be generated during her exploration until she finds the satisfactory results.

Storing either specifications or results of such redundant queries may not only cause unnecessary memory or disk consumption, but may also make the search for queries of potential interest to become increasingly time-consuming. Previous work [25, 24] addressed the query redundancy caused by ambiguous language expressions, mainly in the web and search engine context. However, little effort has been put into how to solve the redundancy caused by the exploration-style query specification mechanism, which is general problem for information exploration system and particularly common in visual exploration systems. This now is the topic of our work.

To solve this problem, we propose to consolidate redundant queries by conducting query combination based on their similarity. The design of distance metrics that effectively measure the similarity between queries in visual environments is challenging. In this paper, our query consolidation solution targets range queries over multi-dimensional data, the major query type in current multivariate visualization systems [23, 14]. So, the technical challenge we address is measuring the similarities between multi-dimensional range queries in multivariate environment .

In our solution, we compare two queries by first comparing their query specifications. The queries having similar selective ranges will be given a smaller distance between each other in terms of query specification. Then considering that even the queries with similar query specification may lead to very different query results, we further compare the results of queries. We propose an algorithm, which we call Exact Transformation Measure (ETM), to measure the similarity between the results of two queries. The main idea of ETM is to measure the distance between two datasets by the cost of transforming one of them to be the same as the other. Two queries that are measured to be similar in both query specification and query results will be assigned a small distance in our solution, while the distance between other pairs will be set to be large. To overcome its high time complexity of ETM when comparing queries with large result sets, we also design an approximation method for our metric, which we show to offer high time efficiency while still providing excellent accuracy.

To evaluate the effectiveness of our metric, we have performed a user study to compare our proposed measure with other distance metrics. The results of our user study show that our metric outperforms other distance metrics in terms of matching with users' intuition. In most cases, the distances computed by our proposed metrics are identical or very close to those provided by the users – while other techniques from the literature tend to fair worse.

As further proof of feasibility, we have incorporated our query consolidation solution into a prototype framework, which we call Nugget Management System (NMS). NMS extracts, consolidates, and maintains important queries submitted to visualization systems. It is currently implemented on XmdvTool [23], a freeware multivariate visualization system. 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 user studies confirmed the effectiveness of NMS and thus also our query consolidation solution.

The main contributions of this work are: 1. We design a query consolidation solution that effectively reduces the potential redundancy among important queries collected. 2. Since query consolidation is based on similarity of nuggets, we develop an distance metric (QD + ETM) to capture the distance between two nuggets. ETM can be used as a standard similarity metrics to compare the subsets of multi-dimensional datasets. 3. We perform user study to compare different nugget distance metrics. The results of the user study indicate that our metric (QD + ETM) works best in terms of matching with users intuition. 4. We introduce a framework that manages important queries in visualization systems, called NMS. NMS utilizes the resulting nugget pool to guide users exploration process. 5. We implement NMS into XmdvTool, a freeware multivariate data visualization tool. 6. 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 visual exploration.

The remainder of this paper is organized as follows: Section 2 gives the preliminaries. Section 3 introduces the techniques used in query consolidation. Our proposed distance metric between two queries is discussed in Section 4. In section 5, we describe a user study comparing different distance metrics. Finally, we discuss the application of query consolidation for exploration support in Section 6.

2. PRELIMINARIES

2.1 Queries in Visualization Systems

Query specification, one of the most important interaction techniques for information management systems, is also being widely used in visualization systems to let users locate and isolate the information of interest. Typical querying techniques used in visualization systems include brushing, filtering and dynamic querying [1]. In this work, we concentrate on multivariate data and thus on queries that impose multi-dimensional constraints on such data (e,g., multi-dimensional range queries). This query type is based on the brushing technique [14], the primary interaction technique for multivariate visualization systems. A specific query Q in our case can be defined as:

$Q =$ Select $D.A_r, D.A_s, ..., D.A_t$ From D Where $D.A_r = [A_r.b_l : A_r.b_h], D.A_s = [A_s.b_l : A_s.b_h], ..., D.A_t = [A_t.b_l : A_t.b_h]$; $\{D.A_r, D.A_s, ..., D.A_t\} \subseteq$ attributes of D, $A_x.b_l$ and $A_x.b_h$ are the lower 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 $A_x.b_h$";

This query type is independent from the display methods in visualization systems, such as Parallel Coordinates, Scatterplots and Glyphs [23]. Without loss of generality, we use Parallel Coordinates [12], which is a widely used display method for multivariate data, to demonstrate our queries in this paper. In Parallel Coordinates, each dimension corresponds to an axis, and the N axes are organized as uniformly spaced vertical lines. A data element in the N-dimensional space manifests itself as a connected set of points, one on each axis. Points lying on a common line or plane create readily perceived structures in the image. Figure 1 shows an example of a four dimensional dataset displayed with Parallel Coordinates. Figure 2 demonstrates a specific range query over this data. Visually a range query appears as a blue band across the axes, which represents the query's selective ranges on each dimension, and the red (highlighted) lines that indicate the selected records (result) of the query. Users specify different queries by adjusting the lower and upper bounds of the blue band (selection ranges).

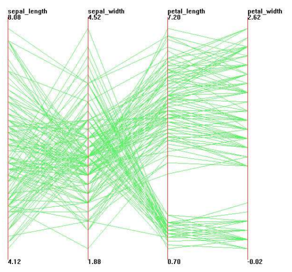


Figure 1: "Iris" dataset displayed with Parallel Coordinates

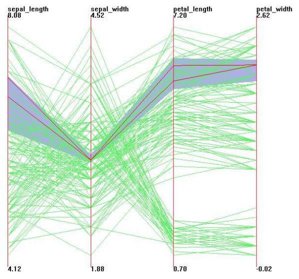


Figure 2: An example query over "Iris" dataset in Parallel Coordinates

2.2 Nugget Extraction

Generally, the term nugget refers to some valuable information in the dataset, which is isolated by an important query. For the purpose of this paper, a nugget is defined by a range query Q (discussed in Section 2.1) over a dataset D as well as the result of this query, dataset Q(D). A more extensive range of nugget types will be considered in our future work.

Similar to other visualization systems [8, 17], users of NMS can explicitly indicate if a particular piece of information is of interest to them. This is done by explicitly saving the given query and labeling it by a persistent nugget name. However, since visual exploration is generally an intensive process that may require may continuous concentration and response by the users. a non-intrusive nugget extraction method would clearly be preferred. In NMS, nuggets can also be automatically extracted by observing a user's exploration. "Visiting time" is one factor [5] 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. For example, if a subpart shown in Figure 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 potential nugget.

However, nugget extraction, as the initial effort to identify the important queries, could cause the problem of "nugget redundancy". Its solution, as the topic of this work, will be discussed in Sections 3, 4 and 5.

3. NUGGET CONSOLIDATION

3.1 The Nugget Redundancy Problem

Now we explain why relying on nugget extraction alone may lead to a major problem of "nugget redundancy". The main reason is that in visual exploration, users utilize visual interaction techniques, such as sliders or selecting windows, to specify the queries while observing the impact on the linked data display, rather than by explicitly typing exact queries as typically done in SQL query systems. By using these exploration-style query specification mechanism, many similar but not identical queries may be generated even when users are querying over the same features in the dataset, be it a cluster or an outlier. Another reason for nugget redundancy is the automatic nugget extraction. During the visual exploration, a user may continuously yet only slightly adjust the query specification. Extracting all of these similar queries as different nuggets will lead to a nugget pool with many redundant nuggets. Figure 3 shows an example of three similar nuggets, which actually capture the same pattern in the dataset.

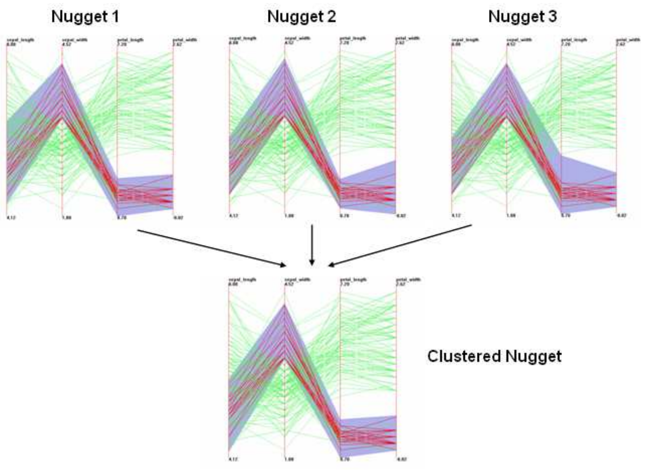


Figure 3: Clustering three similar nuggets

3.2 Nugget Clustering

Now we introduce our solution of nugget consolidation, which keeps the nugget pool in modest size yet with high representativeness. Several different techniques, such as sampling , filtering and clustering achieve this goal. In our solution, we choose clustering, a widely used techniques to combine similar queries [25, 24], which groups similar nuggets together and generates a representative for each group. 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 group member as their representative. Since in many cases, we may hardly be able to 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 nugget in a group makes more sense than just picking one. And since we can use importance (indicated by "visiting time") as the weight in clustering process, the representative generated will primarily reflect the feature of the dominant (super important) nugget, if there is any. One example of forming a representative (clustered nugget) for similar nuggets is shown in Figure 3.

Clustering aims to group objects based on their similarities. We thus need to devise a distance measure that best expresses the domain specific similarity between objects. In particular, we have to develop a suitable distance metric for our multi-dimensional nuggets. This metric will be discussed in Sections 4 and 5.

With a proper distance metric, any generic clustering algorithm [27, 11] can be applied to conduct nugget clustering. To provide real time clustering service for our nugget pool, incremental clustering algorithms [4] are more suitable for our system. Further discussion on the specific clustering algorithm used in our system can be found in [26].

4. DISTANCE METRICS BETWEEN NUGGETS

4.1 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 [24, 25] has studied this problem. The major principle utilized for measuring the query similarity (QS) can be summarized as Formula 1,

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

where QS(A,B) represents the query similarity between Nuggets A and B, and QA and QB are the qualifier 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 as discussed below.

Discrete Domains: A discrete domain composed of nominal values is easy to handle. Because of the discrete property, a direct use of Formula 1 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.

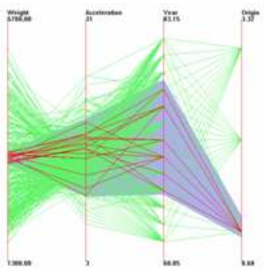


Figure 4: Query 1

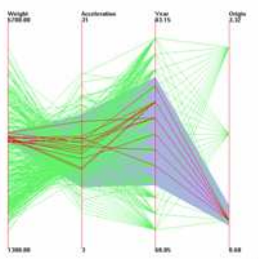


Figure 5: Query 2

As shown in Figures 4 and 5, queries 1 and 2 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.

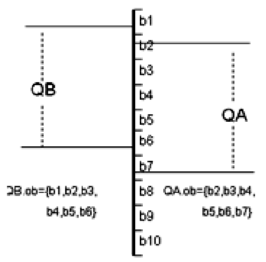


Figure 6: Overlap case

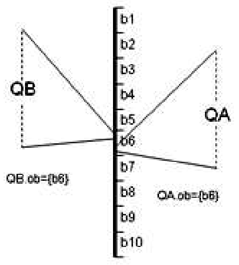


Figure 7: non-overlap case

As demonstrated in Figures 6 and 7, now both overlap and non-overlap cases are handled by our new algorithm. In Figure 6, $QA \cap QB = |QA.ob \cap QB.ob| = |\{b2, b3, b4, b5, b6\}| QA \cup QB = |QA.ob \cup QB.ob| = |\{b1, b2, b3, b4, b5, b6, b7\}| = 7$,and QS=5/7=7.1. In Figure 7, $QA \cap QB = |QA.ob \cap QB.ob| = |\{b6\}| = 1, QA \cup QB = |QA.ob \cup QB.ob| = |\{b6\}| = 1$, QS=1/1=1. In practice, we could set QS less then 1 for non-overlap cases, because after all they are not perfect matches.

The "occupied bin count strategy" can be used as a uniform query similarity metric for range queries on a single dimension. The discrete domain uses each discrete value as its bin, while the continuous domain divides the continuous domain into bins first.

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. The reason why We choose "Minimum" rather than Manhattan Distance or Euclidean Distance is because "Minimum" best capture the "visual similarity" of two nuggets. More detail discussion on this choice can be found in [26].

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.

$$QD(A, B) = 1 - QS(A, B) \tag{2}$$

4.2 Data Distance

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

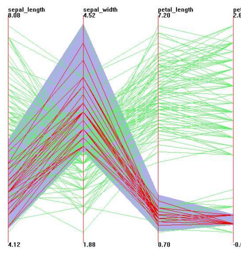


Figure 8: A nugget capturing a cluster in the "Iris" dataset

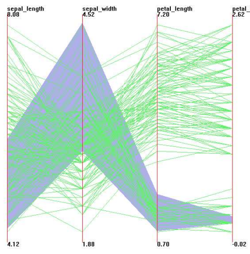


Figure 9: A nugget with no data record included

shown in Figures 8 and 9, 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 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 [2, 7, 16, 6] can be generally classified into two main categories, statistical and transform-cost approaches. Below, we explain why we choose the latter, and then introduce a proposed algorithm extending a basic transform cost algorithm.

4.2.1 *Statistical Approaches*

Since traditional statistic methods, such as average and deviation, cannot fully capture the characteristics of two subsets, a more sophisticated method based on histograms, Histogram Difference Measures (HDM), has been developed. HDM is used in data abstraction quality measure [7], approximate query processing of databases [2]. It compares the histograms of two sets of data, meaning the distributions of data points.

However multi-dimensional histograms surfers 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. As a much cheaper alternative, the integration of single-dimensional histograms first compares histograms on each dimension separately and then integrates the results into a normalized result. 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.

4.2.2 *Transform Cost Approaches*

As a general notion, Transform Cost methods have been shown to be effective in a wide range of different areas, such as "Edit Distance" in string matching [16], and "DIFF" in change detection to HTML and XML files [6]. In the Transform Cost Approaches, 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 the Nearest Neighbor Measure (NNM). 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 statistics of datasets. But NNM appears to work better for measuring the quality of representativeness due its n to 1 mapping strategy, meaning a data point in one dataset may be mapped as nearest neighbor for more than one data point in another. For example, 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 .

4.3 The proposed ETM measure

To solve this problem, we propose a new algorithm called Exact Transformation Measure (ETM). ETM 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.

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 the following 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 10 shows an example of two 2-dimensional datasets where moving and adding are not sufficient to make a cost effective transformation plan.

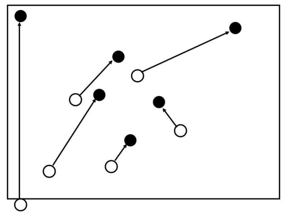


Figure 10: Trasforming A to B with moving and adding operation only

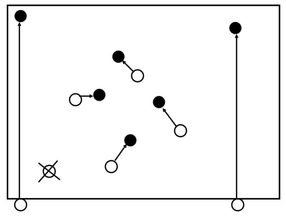


Figure 11: Transforming A to B with, moving, adding and deleting

As shown in Figure 10, given dataset A (shown as white points) and B (shown as black points), 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 would help us to achieve a more cost-effective transformation, as shown in Figure 11.

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 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”. We choose to “Delete”, if moving a data point brings too much global cost.

4.3.1 Cost Models

To make an optimal transformation plan, which has the minimum cost, 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 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 of two steps: generation (generating a point at a random position) and moving (moving the point to a certain position). Thus, COA can be expressed in the following way:

$$COA = GC + MC \tag{3}$$

a) Generating Cost (GC): It is not hard to see that generating a new element for A would cause a greater “mutation” to it, when A is small. For instance, when $|A| = 0$, 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:

$$GM = \frac{MPD}{a+1} \tag{4}$$

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 centroid) 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, we set the upper limit to 1.

- *Cost of Delete(x)– Cost($D[x]$)*

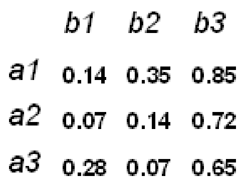
Similar to GC, the change cost of deleting is associated with the cardinality of A and unrelated to the position where the deleted data point lies. The difference here is that we do not need to handle the cases where a=0, because we can delete a data point only if it exists. So, we use Cost Of Delete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

$$COD = \frac{MPD}{a} \tag{5}$$

4.3.2 Making Transformation Plan

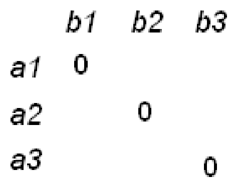
Having defined the cost models, we now establish our solution for finding an optimal (most cost-effective) transformation plan. We observe that the Hungarian Assignment Method[22, 15] which was designed for finding minimum cost bipartite matches, provides a good approach to solve this NP- hard-like problem 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 12. 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 13). Further details of Hungarian Assignment Method can be found in [22, 15].



	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 12: Input matrix



	b1	b2	b3
a1	0		
a2		0	
a3			0

Figure 13: Output matrix

To be able to use Hungarian Assignment Method in all the cases, we still have to tackle two issues: **1)** two subsets to be compared do not necessarily have the same population. **2)** We need to incorporate the adding and deleting actions into the transformation plan. For this, two modifications to the input matrix are needed.

1) 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.

2) Incorporating Adding and Deleting: When moving a_i to b_i is more expensive then deleting it and adding a new data point to where b_i lies, we choose the latter "deleting + adding" strategy instead of moving. 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.

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 X and Y, where A and B are the datasets contained by them.

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.

$$ND[X, Y] = \alpha \cdot QD[X, Y] + \beta \cdot DD[X, Y] \qquad (\alpha + \beta = 1) \qquad (7)$$

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

5. USER STUDY ON DISTANCE METRICS

Now, we discuss several experimental studies conducted to show the effectiveness of the proposed distance metric.

5.1 Experimental Setup

Users: 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. 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 nuggets 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 designed nuggets are good examples of real nuggets which users could make in their navigations, because they covered all the pattern types we discussed in this work and have varying 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: The user study consisted of 20 questions. Each of them requires users to indicate a distance between a pair of nuggets. Particularly, all the distances were scaled by the integers from 0 to 10 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", initially selected. The sequence of the questions was randomly arranged, but once it was arranged , it was kept identical for all users.

Experimental Methodology: A brief instruction for the user study was given before the specific questions were presented to the user. 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.

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 to compute the distances between the same 20 pairs of nuggets [1]. Finally, we compare the distances given by the users with those computed by each of the metrics. For this, we introduce a function "Dif" 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=$|D_U(N_x, N_y) - D_M(N_x, N_y)|$. For each pair of nuggets, we assign different amount of credit, called accurate credit, to each metric based on the difference between the distances given by this metric and by the user. Concretely, we give 3 credits to a metric if Dif=0, 2 credits, 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. We also use pie graphs, which we call "Dif Distribution Graph" to show us the exact number and percentage of each "match category" (Dif=0, Dif=1, and so on).

5.2 Experimental Results

[1]In this section, we use "QD + ETM (HA)" to present our original QD + ETM solution, which employs the Hungarian Assignment Method. We use "QD + ETM (CC)" to present another version of our QD + ETM solution (we'll discuss later in this section), which is implemented by an approximation algorithm to Hungarian Assignment Method

Figure 14 shows the accumulative credits earned by each metric. Our accurate credit strategy counts all the "matches"

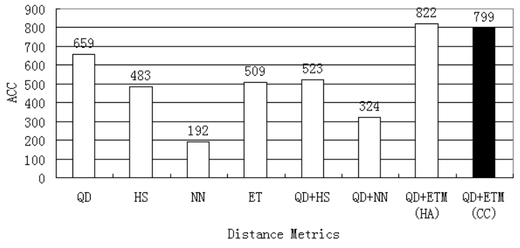


Figure 14: Accumulative credits earned by each distance metrics for all 400 cases

and sums up the Accurate Credit earned by each distance metric for all the 400 cases. Generally, a metric that matches well with more users in more questions will earn higher accumulative credit. As shown in Figure 14 [2], QD+ETM (HA) earns much higher accumulative credit than any other metric. This indicates that it matches the users intuition best among all distance metrics.

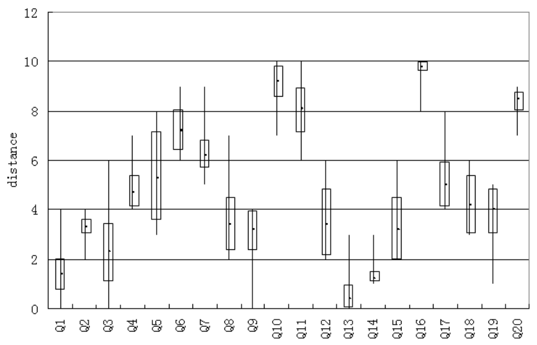


Figure 15: Distances given by 20 users for 20 questions

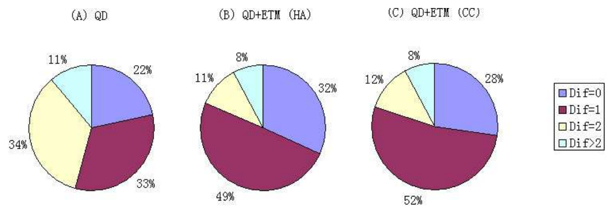


Figure 16: Dif distribution of QD, QD+ETM (HA) and QD+ETM (CC)

Figures 16 shows the distribution of "Dif"s for several distance metrics. We observe that QD + ETM (HA) 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 metric 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 (HA) captures the distances between nuggets best among all the metrics we discussed in this work.

5.3 Approximation to Hungarian Assignment Algorithm

The best quality usually comes at the price of the highest cost. Since the Hungarian Assignment (HA) Method 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 17 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 (HA) has highest cost in terms of maximum, minimum and average CPU time used, which means the metric is best at capturing users' intuition but worst in terms of time efficiency.

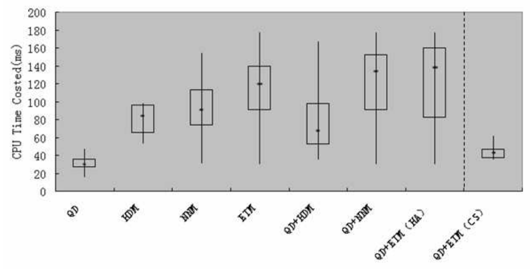


Figure 17: CPU time by different distance metrics

To address this, we now propose to employ a much cheaper approximation algorithm of HA instead of the full-fledged HA. 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 construct a global optimal transformation plan by conducting complicated matrix operations as we did with HA. Instead, we "move" each non-overlapping 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" moved 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, etc. This continues until a data point finds an unoccpupied neighbor, or the data point has to be "deleted" in the transformation plan.

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.

As shown in Figure 17, from the maximum, minimum, average and also standard deviation of CPU time cost, we learn that QD+ETM(CC) is the second fastest distance metric among all those we have discussed and only slightly slower than the cheapest metric (QD). Here, we also need to point out that, although QD+ETM (CC) saves much CPU time

[2]please temporarily ignore QD + ETM (CC) metric which will be discussed later

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. 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, one could 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 the exactly same answer with QD+ETM (HA) in 18 pairs of them. For the remained 2 pairs of nuggets, QD+ETM (CC) versus QD+ETM (HA) has a difference equal to 1 and another has a difference equal to 2.

As shown in Figure 14, the accumulative credits earned by QD+ETM (CC) are very close to those of QD+ETM (HA) and much higher than any other metrics. From figure 16, we 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" ($Dif > 2$).

5.4 Conclusions on Distance Metrics

Based on the above experimental studies we conclude:

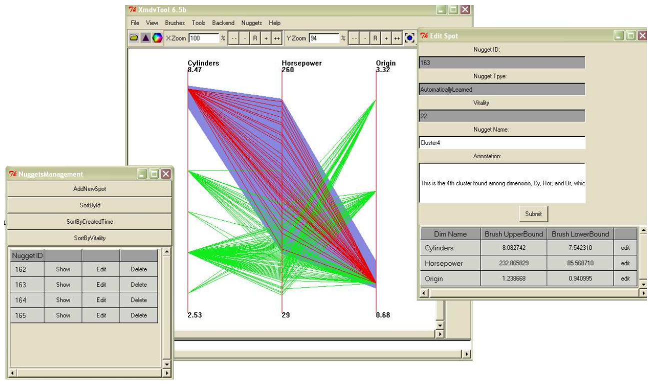
1) QD+ETM (HA) agrees well with users' intuition on distances between nuggets. It is thus the best distance metrics in terms of capturing nugget distance.

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 accuracy with HA, and it costs significantly less on average (80%) CPU time than HA for the nuggets on three real datasets we employed.

3) Query Distance (QD) only, as a cheap metrics, works well in many cases. 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. If two nuggets have a small query distance and we aim to form as precise clusters as possible, we have to consider data distance. 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).

6. APPLICATION OF NUGGET CONSOLIDATION FOR EXPLORATION SUPPORT

To verify the usefulness of the proposed solution, we have integrated Nugget Consolidation into our NMS framework and then evaluated its effectiveness in aiding users during data exploration. More precisely, we have extended XMDV-tool, a freeware multivariate visualization system [23], with the services of nugget extraction, consolidation, and maintenance to provide nugget support during visual data exploration. In our system, nugget consolidation first removes the redundant nuggets generated during nugget extraction. Then, the "nugget representatives" produced by nugget consolidation can be retrieved by the users and feed into the nugget maintenance stage. Nugget maintenance expels the out-of-date nuggets or those extracted by misinterpreting users interest from our system. While we provide automated algorithm for these services, we also offer interfaces that enable users to manipulate each individual nugget. For example, a user can modify the query specification of a nugget or attach her own understanding as annotation to a nugget. By doing so, NMS may form a nugget pool that well presents users' discoveries, interest and expertise. This well-organized nugget pool will be used in nugget-guided exploration, which can guide users exploration in both user- and system-initiated modes. Figure 18 shows a screen shot of our prototype system. More details of our prototype NMS system can be found in [26].



Nugget ID			
162	Show	Edit	Delete
163	Show	Edit	Delete
164	Show	Edit	Delete
165	Show	Edit	Delete

Nugget ID: 163

Nugget Tpye: AutomaticallyLearned

Vitality: 22

Nugget Name: Cluster4

Annotation: This is the 4th cluster found among dimension, Cy, Hor, and Or, whic

Dim Name	Brush UpperBound	Brush LowerBound	
Cylinders	8.082742	7.542310	edit
Horsepower	232.865829	85.568710	edit
Origin	1.238668	0.940995	edit

Figure 18: A screen shot from our NMS prototype when looking for clusters hidden in the dataset

To compare users' efficiency and accuracy when solving tasks with and without the help of NMS. Specifically, we randomly divided 12 users, all WPI students, into 4 groups, 3 users per group (all different from the subjects in user study for distance metrics). All the 4 groups were asked to finish the same 5 knowledge discovery tasks, which were based 3 real datasets. In this 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. All the users were encouraged to finish the tasks as quickly and correctly as possible. Figure 19 shows time used by each individual user and also each group.

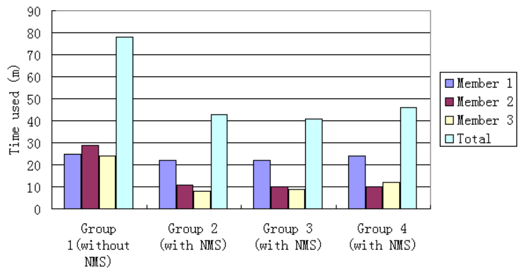


Figure 19: Comparison of users' efficiency in different groups

As shown in Figure 19, 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. Thus, we showed that NMS may greatly improve users time efficiency when solving knowledge discovery tasks. Our preliminary evaluation also shows that NMS enhances users' accuracy of finishing these tasks. More details of these user studies to NMS can be found in [26].

7. RELATED WORK

[3] introduces a technique for mining a collection of user transactions with an Internet search engine to discover clusters of similar queries and similar URLs. [10] describes a collaborative query system that helps users with query formulation by finding previously submitted similar queries through mining web logs. Similar works include [24, 25]. These works which aim to mine important queries from query logs have similar goal to us. However, they focus on keyword-based queries in web searches, which is different from our work in terms of both the query type and the visual specification context.

Although these studies on keyword-based queries in web searches cannot be directly applied to our context, their efforts to measure the similarity between queries parallel ours. As pointed out by [10], when measuring the similarity between two queries, we should not only compare their "terms", which means the query specifications, but also compare the results of them. We use the same principle to measure the similarity between queries in our work, although the specific methods we designed to conduct comparisons are different. Previous studies on similar queries [24] also provides us the basic idea of comparing the query specification, which is that the specifications with a large "overlap" should be more similar to each other. We extended this basic idea to handle different types of domains and to guarantee the "visual simialirty" between queries in our context.

Our work is also closely related to visualization and interaction techniques [23, 19, 9], because our NMS framework is designed to facilitate visual exploration. The query type we target is based on a major querying mechanism for multivariate visualization systems, called brush [14]. Finally, the framework of NMS has good potential to support visual analytics as also targeted in [21, 17].

8. CONCLUSIONS

As the main contribution of this work, we present a query consolidation solution, which consolidates redundant queries caused by exploration-style query specification, which is commonly found in visual exploration systems. Our solution clusters queries based on their similarity. To solve the challenge of measuring the similarity between queries, we have developed a collaborative distance metric, which compares both specifications and results of two queries. Our user study conducted on visual queries over real datasets shows that our distance metric matches well with users' intuition. Further, we outline our effort in realizing some nugget support by incorporating query consolidation into a freeware visual exploration system. Our preliminary evaluation indicates that it indeed greatly enhance both the efficiency and accuracy of users' visual exploration.

9. REFERENCES

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

[2] B. Babcock, S. Chaudhuri, and G. Das. Dynamic sample selection for approximate query processing. In *ACM SIGMOD*, pages 539–550, 2003.

[3] D. Beeferman and A. L. Berger. Agglomerative clustering of a search engine query log. In *KDD*, pages 407–416, 2000.

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

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

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

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

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

[9] 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.

[10] L. Fu, D. H.-L. Goh, S. S.-B. Foo, and J.-C. Na. Collaborative querying through a hybrid query clustering approach. In *ICADL*, pages 111–122, 2003.

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

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

[13] D. Lee and W. W. Chu. Semantic caching via query matching for web sources. In *CIKM*, pages 77–85, 1999.

[14] A. Martin and M. Ward. High dimensional brushing for interactive exploration of multivariate data. *Proc. of Visualization*, pages 271–278, 1995.

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

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

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

[18] Z. W. Ras. The role of support and confidence in

collaborative query answering. In *Intelligent Information Systems*, pages 221–226, 2001.

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

[20] G. Soundararajan and C. Amza. Using semantic information to improve transparent query caching for dynamic content web sites. In *DEEC*, pages 132–138, 2005.

[21] 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.

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

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

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

[25] J.-R. Wen and H. Zhang. Query clustering in the web context. In *Clustering and Information Retrieval*, pages 195–226. 2003.

[26] D. Yang. Analysis-guided exploration of multivariate data. Master's thesis, Worcester Polytehnic Institute, 2007.

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