International Journal of Modern Engineering Research (IJMER)
www.ijmer.com Vol.2, Issue.6, Nov-Dec. 2012 pp-4497-4501 ISSN: 2249-6645

A Heuristic Algorithm to Reduce Information Overhead Based On Hierarchical Category Structure

Kishore Kumar Reddy[1], M M M K Varma[2]
[1](M.Tech, Department of CSE, Sri Sivani College of Engineering, Andhra Pradesh, India
[2] (Assistant Professor, Department of CSE, Sri Sivani College of Engineering, Andhra Pradesh, India

Abstract : *Database systems are being increasingly used for interactive and exploratory data retrieval. In such retrieval queries often cause in too many answers, this phenomenon is referred as "information overload". In this paper, we proposed a solution which can automatically categorize the results of SQL queries to address this problem. The proposed solution dynamically generate a labeled hierarchical category structure where users can determine whether a category is relevant or not by checking its label and can explore the relevant categories and ignore other categories to reduce information overload. In the proposed solution, an analytical model is designed to estimate information overload faced by a user for a given exploration. Based on the model, we formulate the categorization problem as a cost optimization problem and develop heuristic algorithms to calculate the min-cost categorization.*

Keywords**:** *Data Mining, Information Overload, Categorization, Ranking, Discretization*

I. Introduction

Database systems are being increasingly used for interactive and exploratory data retrieval [1, 2, 8, 14]. In such retrieval, queries often result in too many answers. Not all the retrieved items are relevant to the user, only a few result set is relevant to the user. Unfortunately, user needs to check all the retrieved items to find relevant information need to the user query. This phenomenon is commonly referred to as "information overload". Consider a scenario, real-estate database that maintains information like the location, price, number of bedrooms etc. of each house available for sale. Suppose that a potential buyer is looking for homes in the Seattle/Bellevue Area of Washington, USA in the $200,000 to $300,000 price range. The above query, henceforth referred to as the "Homes" query, returns 6,045 homes when executed on the MSN House&Home home listing database. Information overload makes it difficult for the user to differentiate the interesting and uninteresting items, which leads to a huge wastage of user's time and effort. Information overload can happen when the user is not certain about the query. In such a situation, user can pose a broad query in the beginning to avoid exclusion of potentially interesting results. For example, a user shopping for a home is often not sure of the exact neighbourhood she wants or the exact price range or the exact square footage at the beginning of the query. Such broad queries may also occur when the user is naïve and refrains from using advanced search features [8]. Finally, information overload is inherent when users are interested in browsing through a set of items instead of searching among them.

In the context internet text search, there has been two canonical ways to avoid information overload. First, they group the search results into separate categories. Each category is assigned a descriptive label examining which the user can determine whether the category is relevant or not. Then user can visit the relevant categories and ignore the remaining ones. Second, they present the answers to the queries in a ranked order. Thus, categorization and ranking present two complementary techniques to manage information overload. After browsing the categorization hierarchy and/or examining the ranked results, users often reformulate the query into a more focused narrower query.

Therefore, categorization and ranking are indirectly useful even for subsequent reformulation of the queries.

In contrast to the internet text search, categorization and ranking of query results have received much less attention in the database field. Recently ranking of query results has gained some attention. But all the existing methods have not critically examined the use of categorization of query results in a relational database. Hence in this paper, categorization of database query results presents some unique challenges that are not addressed in various search engines/web directories. In all the existing search engines, the category structures are created a priori and the items are tagged in advance as well. At search time, the search results are integrated with the pre-defined category structure by simply placing each search result under the category it was assigned during the tagging process. Since such categorization is independent of the query, the distribution of items in the categories is susceptible to skew: some groups can have a very large number of items and some very few.

For example, a search on 'databases' on Amazon.com yields around 34,000 matches out of which 32,580 are under the "books" category. These 32,580 items are not categorized any further (can be sorted based on price or publication date or customer rating) and the user is forced to go through the long list to find the relevant items. This defeats the purpose of categorization as it retains the problem of information overload. In this paper, we propose techniques to automatically categorize the results of SQL queries on a relational database in order to reduce information overload. Unlike the "a priori" categorization techniques described above, we generate a labelled hierarchical category structure automatically based on the contents of the tuples in the answer set. Since our category structure is generated at query time and hence tailored to the result set of the query at hand, it does not suffer from the problems of a priori categorization discussed above.

This paper discusses how such categorization structures can be generated on the fly to best reduce the information overload. We begin by identifying the space of categorizations and develop an understanding of the exploration models that the user may follow in navigating the hierarchies. Such understanding helps us compare and contrast the relative goodness of the alternatives for

categorization. This leads to an analytical cost model that captures the goodness of a categorization. Such a cost model is driven by the aggregate knowledge of user behaviours that can be gleaned from the workload experienced on the system. Finally, we show that we can efficiently search the space of categorizations to find a good categorization using the analytical cost models. Our solution is general and presents a domain-independent approach to addressing the information overload problem. We perform extensive experiments to evaluate our cost models as well as our categorization algorithm.

The rest of the paper is organized as section 2: discuss about the related work, section 3: discuss about basics of categorization section 4: presents the Proposed Solution, section 5: discuss about cost model, section 6: categorization algorithm, section 7: discuss about Experimental setup, section 8: concludes the paper.

II. Related Work

OLAP and Data Visualization: Our work on categorization is related to OLAP as both involve presenting a hierarchical, aggregated view of data to the user and allow to drilldown/ roll-up the categories [13]. However, in OLAP, the user needs to manually specify the grouping attributes and grouping functions [13] in our work, those are determined automatically. Information visualization deals with visual ways to present information [6]. It can be thought as a step after categorization to the further reduce information overload: given the category structure proposed in this paper, we can use visualization techniques visually display the tree [6].

Data Mining: The space in which the clusters are discovered is usually provided there whereas, in categorization, we need to find that space. Second, existing clustering algorithms deal with either exclusively categorical [11] or exclusively numeric spaces [17] in categorization, the space usually involves both categorical and numeric attributes. Third, the optimization criteria are different while it is minimizing inter-cluster distance in clustering to decrease information overload. Our work differs from classification where the categories are already given their along with a training database of labelled tuples and we need predict the label of future, unlabeled tuples [12].

Discretization/Histograms: The discretization assumes that there is a class assigned to each numeric value and uses the entropy minimization heuristic [10]. On the other hand, the histogram bucket selection is based on minimization of errors in result size estimation [5, 15].

Ranking: Ranked retrieval has traditionally been used in Information Retrieval in the context of keyword searches over text/unstructured data [3] but has been proposed in the context of relational databases recently [2,4,14]. Ranking is a powerful technique for reducing information overload and can be used effectively in complement with categorization. Although categorization has been studied extensively in the text domain [9, 16] to the best of our knowledge, this is the first proposal for automatic categorization in the context of relational databases.

III. Basics of Categorization

Space of Categorizations

Let R be a set of tuples. R can either be a base relation or a materialized view or it can be the result of a query Q. We assume that R does not contain any aggregated or derived attributes, i.e., Q does not contain any GROUP BYs or attribute derivations (Q is a SPJ query). A hierarchical categorization of R is a recursive partitioning of the tuples in R based on the data attributes and their values. We define a valid hierarchical categorization T of R inductively as follows.

***Base Case*:** Given the root or "ALL" node (level 0) which contains all the tuples in R, we partition the tuples in R into an ordered list of mutually disjoint categories (level 1 nodes2) using a single attribute.

***Inductive Step*:** Given a node C at level (l-1), we partition the set of tuples tset(C) contained in C into an ordered list of mutually disjoint subcategories (level l nodes) using a single attribute which is same for all nodes at level (l-1). We partition a node C only if C contains more than a certain number of tuples. The attribute used is referred to as the categorizing attribute of the level l nodes and the subcategorizing attribute of the level (l-1) nodes. Furthermore, once an attribute is used as a categorizing attribute at any level, it is not repeated at a later level, i.e., there is a 1:1 association between each level of T and the corresponding categorizing attribute. We impose the above constraints to ensure that the categorization is simple, intuitive and easily understandable to the user.

Associated with each node C is a category label and a tuple-set as defined below:

Category Label: The predicate label(C) describing node C. For example, the first child of root (rendered at the top) has label 'Neighbourhood ∈ {Redmond, Bellevue}' while the first child of the above category has label 'Price: 200K– 225K'.

Tuple-Set: The set of tuples tset(C) (called the tuple-set of C) contained in C either appearing directly under C (if C is a leaf node) or under its subcategories (if C is a non-leaf node). Formally, tset(C) is the set of tuples, among the ones contained in the parent of C, which satisfy the predicate label(C). In other words, tset(C) is the subset of tuples in R that satisfies the conjunction of category labels of all nodes on the path from the root to C

The label of a category unambiguously describes to the user which tuples, among those in the tuple set of the parent of C, appear under C. Hence, user can determine whether C contains any item that is relevant or not by looking just at the label and hence decide whether to explore or ignore C.

IV. Proposed Model

We present two models capturing two cases in data exploration. One scenario is that the user explores the result set R using the category tree T until finds relevant tuple t ∈R. For example, the user may want to find every home relevant to her in the "Homes" query. In order to ensure that user finds every relevant tuple and needs to examine every

tuple and every category label except the ones that appear under categories she deliberately decides to ignore. Another scenario is that the user is interested in just one (or two or a few) tuple(s) in R so she explores R using T till she finds that one (or few) tuple(s). For example, a user may be satisfied if she finds just one or two homes that are relevant to her. For the purpose of modeling, we assume that, in this case, the user is interested in just one tuple, i.e., the user explores the result set until finds the first relevant tuple. We consider these two cases because they both occur commonly and they differ in their analytical models.

Exploration Model for 'All' case

Figure 1 describes the exploration model for all case the user starts the exploration by exploring the root node. Given user has decided to explore the node C, if C is a non-leaf node, user non-deterministically chooses one of the two options

Option 'SHOWTUPLES': Browse through the tuples in tset(C). Note that the user needs to examine all tuples in tset(C) to make sure that she finds every tuple relevant to her.

Option 'SHOWCAT': Examine the labels of all the n subcategories of C, exploring the ones relevant to her and ignoring the rest. More specifically, she examines the label of each subcategory Ci starting from the first subcategory and non-deterministically chooses to either explore it or ignore it. If user chooses to ignore Ci, simply proceeds and examines the next label (of C_i+1). If user chooses to explore C_i, it does so recursively based on the same exploration model, i.e., by choosing either 'SHOWTUPLES' or 'SHOWCAT' if it is an internal node or by choosing 'SHOWTUPLES' if it is a leafnode. After she finishes the exploration of C_i, user goes ahead and examines the label of the next subcategory of C (of C_i+1). Note that we assume that the user examines the subcategories in the order it appears under C it can be from top to bottom or from left to right depending on how the tree is rendered by the user interface.

If C is a leaf node, 'SHOWTUPLES' is the only option (option 'SHOWCAT' is not possible since a leaf node has no subcategories).

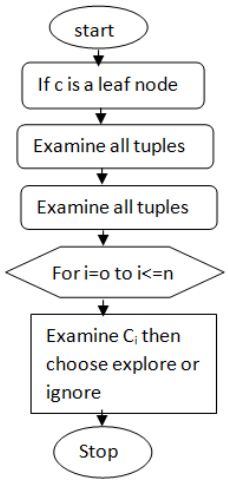

Figure 1: Flow chart model of exploration of node C in 'All' case

Exploration Model for 'One' Scenario

The user starts the exploration by exploring the root node. Given that the user has decided to explore a node C, user non-deterministically choose one of the two options

***Option 'SHOWTUPLES'*:** Check the tuples in tset(C) starting from the first tuple in tset(C) till user finds the first relevant tuple. In this paper, we do not assume any particular ordering/ranking when the tuples in tset(C) are presented to the user.

***Option 'SHOWCAT'*:** Examine the labels of the subcategories of C starting from the first subcategory till the first one she finds interesting. As in the 'ALL' case, user checks the label of each subcategory Ci starting from the first one and non-deterministically chooses to either explore it or ignore it. If user chooses to ignore Ci, simply proceeds and checks the next label. If user chooses to explore Ci, it does so recursively based on the same exploration model. We assume that when she drills down into Ci, user finds at least one relevant tuple in tset (Ci); so, unlike in the 'ALL' case, the user does not need to examine the labels of the remaining subcategories of C.

V. Cost Estimation

Since we want to generate the tree imposes the least possible information overload on the user, we need to estimate the information overload that a user will face during an exploration using a given category T.

Cost Model for 'ALL' case

Let us first consider the 'ALL' case. Given a user exploration X using category tree T, we define information overload cost, or simply cost (denoted by Cost All(X, T)), as the total number of items examined by the user during X. This definition is based on the assumption that the time spent in finding the relevant tuples is proportional to the number of items the user needs to examine more the number of items user needs to examine, more the time wasted in finding the relevant tuples, higher the information overload.

Cost Model for 'ONE' Scenario

The information overload cost CostOne (T) that a user will face, on average during an exploration using a given category tree T is the number of items that a user will need to examine till user finds the first relevant tuple. Let us consider the cost CostOne(C) of exploring the subtree rooted at C given that the user has chosen to explore C, CostOne (T) is simply CostOne (root). If the user goes for option 'SHOWTUPLES' for C and frac (C) denotes the fraction of tuples in tset(C) that she needs to examine, on average, before she finds the first relevant tuple, the cost, on average, is frac(C)*|tset(C)|. If user goes for option 'SHOWCAT', the total cost is (K*i + CostOne (C_i)) if C_i is the first subcategory of C explored by the user (since the user examines only i labels and explores only C_i).

$$\mathrm{Cost_{One}}(C) = P_w(C) * frac(C) * |tset(C)|$$
$$+ (1-P_w(C)) * \sum_{i=1}^{n} \left(\prod_{j=1}^{(i-1)} (1-P(C_j)) * P(C_i) * (K*i + \mathrm{Cost_{One}}(C_i)) \right)$$

VI. Categorization Algorithm

Since we know how to compute the information overload cost Cost All (T) of a given tree T, we can enumerate all the permissible category trees on R, compute their costs and pick the tree Topt with the minimum cost. This enumerative algorithm will produce the cost-optimal tree but could be prohibitively expensive as the number of permissible categorizations may be extremely large.

Hence we present our preliminary ideas to reduce the search space of enumeration. First we present our techniques in the context of 1-level categorization i.e., a root node pointing to a set of mutually disjoint categories which are not subcategorized any further.

Reducing the Choices of Categorizing Attribute

Since the presence of a selection condition on an attribute in a workload query reflects the user's interest in that attribute, attributes that occur infrequently in the workload can be discarded right away while searching for the min-cost tree. Let A be the categorizing attribute chosen for the 1-level categorization. If the occurrence count NAttr(A) of A in the workload is low, the SHOWTUPLES probability Pw(root) of the root node will be high. Since the SHOWTUPLES cost of a tree is typically much higher than its SHOWCAT cost and the choice of partitioning affects only the SHOWCAT cost, a high SHOWTUPLES probability implies that the cost of the resulting tree would have a large first component (Pw(root)*|*tset(root)*|) which would contribute to a higher total cost. Therefore, it is reasonable to consider eliminating such low occurring attributes without considering any of their partitioning.

Specifically, we eliminate the uninteresting attributes using the following simple heuristic: if an attribute A occurs in less than a fraction x of the queries in the workload, i.e., NAttr (A)/N < x, we eliminate A. The threshold x will need to be specified by the system designer/domain expert. For example, for the home searching application, if we use x=0.4, only 6 attributes, namely neighborhood, price, bedroomcount, bathcount, property-type and square footage, are retained from among 53 attributes in the MSN House2Home dataset.

VII. Experimental Evaluation

To evaluate the performance of the proposed model, we present the results of an extensive empirical study we have conducted to evaluate the accuracy of our cost models in modeling information overload and evaluate our cost-based categorization algorithm and compare it with categorization techniques that do not consider such cost models. Our experiments consist of a real-life user study as well as a novel, large-scale, simulated, cross-validated user-study. For both studies, our dataset comprises of a single table called ListProperty which contains information about real homes that are available for sale in the whole of United States. The ListProperty contains 1.7 million rows (each row is a home) and, in addition to other attributes, has the following non-null attributes: location (neighborhood, city, state, zip code), price, bedroomcount, bathcount, year-built, property-type (whether it is a single family home or condo etc.) and square-footage.

Accurate Cost Model: There is a strong positive correlation between the estimated average exploration cost and actual exploration cost for various users. This indicates that our workload-based cost models can accurately model information overload in real life.

Better Categorization Algorithm: Our cost-based categorization algorithm produces significantly better category trees compared to techniques that do not consider such analytical models.

Table 1: Pearson's Correlation between estimated cost and actual cost

Subset	Correlation
1	0.39
2	0.98
3	0.32
4	0.48
5	0.16
6	0.16
7	0.19
8	0.76
All	0.90

To further confirm the strength of the positive correlation between the estimated and actual costs, we compute the Pearson Correlation Coefficient for each subset separately as well as together in Table 1. The overall correlation (0.9) indicates almost perfect linear relationship while the subset correlations show either weak (between 0.2 and 0.6) or strong (between 0.6 and 1.0) positive correlation. This shows that our cost models accurately model information overload faced by users in real-life.

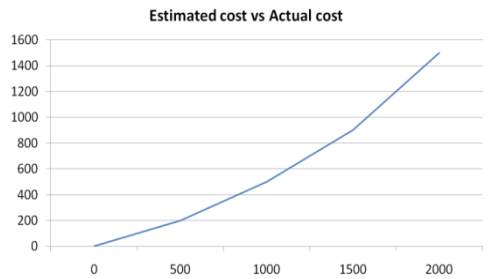

Figure 2: Correlation between actual cost and estimated cost

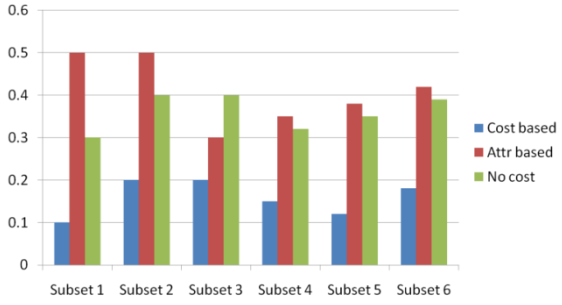

Figure 3: Cost of various techniques

Figure 2 plots the estimated cost against the actual cost for the 800 explorations. The plot along with the trend line (best linear fit with intercept 0 is y = 1.1002x) shows that the actual cost incurred by users has strong positive correlation with the estimated average cost.

Figure 3 compares the proposed technique with the Attr-cost and No cost techniques based on the fractional cost averaged over the queries in a subset.

VIII. Conclusion

Database systems are being increasingly used for interactive and exploratory data retrieval. In such retrieval queries often cause in too many answers, this phenomenon is referred as "information overload". In this paper, we proposed a solution which can automatically categorize the results of SQL queries to address this problem. The proposed solution dynamically generate a labeled hierarchical category structure where users can determine whether a category is relevant or not by checking its label and can explore the relevant categories and ignore other categories to reduce information overload. In the proposed solution, an analytical model is designed to estimate information overload faced by a user for a given exploration. Based on the model, we formulate the categorization problem as a cost optimization problem and develop heuristic algorithms to calculate the min-cost categorization. Our proposed cost-based categorization algorithm produces significant better category trees compared to techniques that do not consider such cost-models.

References

[1] S. Agrawal, S. Chaudhuri and G. Das. DBExplorer: A System for Keyword Search over Relational Databases. In Proceedings of ICDE Conference, 2002.

[2] S. Agrawal, S. Chaudhuri, G. Das and A. Gionis. Automated Ranking of Database Query Results. In Proceedings of First Biennial Conference on Innovative Data Systems Research (CIDR), 2003.

[3] R. Baeza_yates and B. Ribiero-Neto, Modern Information Retrieval, ACM Press, 1999.

[4] N. Bruno, L. Gravano and S. Chaudhuri, Top-k Selection Queries over Relational Databases: Mapping Strategies and Performance Evaluation. In ACM TODS, VO, 27, No. 2, June 2002.

[5] N. Bruno, S. Chaudhuri and L. Gravano. STHoles: A Multidimensional Workload-Aware Histogram. Proc. of SIGMOD, 2001.

[6] S. Card, J. MacKinlay and B. Shneiderman. Readings in Information Visualization: Using Vision to Think, Morgan Kaufmann; 1st edition (1999).

[7] J. Chu-Carroll, J. Prager, Y. Ravin and C. Cesar, A Hybrid Approach to Natural Language Web Search, In Proc. of Conference on Empirical Methods in Natural Language Processing, July 20

[8] S. Dar, G. Entin, S. Geva and E. Palmon, DTL's DataSpot: Database Exploration Using Plain Language, In Proceedings of VLDB Conference, 1998.

[9] S. Dumais, J. Platt, D. Heckerman and M. Sahami, Inductive learning algorithms and representations for text categorization, In Proc. Of CIKM Conference, 1998.

[10] U. Fayyad and K. Irani. Multi-Interval Discretization of Continuous-Valued Attributes for Classification Learning. Proc. of IJCAI, 1993.

[11] V. Ganti, J. Gehrke and R. Ramakrishnan. CACTUS - Clustering Categorical Data Using Summaries. KDD, 1999.

[12] J. Gehrke, V. Ganti, R. Ramakrishnan, W. Loh. BOATOptimistic Decision Tree Construction. Proc. of SIGMOD, 1999.

[13] J. Gray, S. Chaudhuri, A. Bosworth, A. Layman, D. Reichart, M. Venkatrao, F. Pellow and H. Pirahesh. Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Totals. Journal of Data Mining and Knowledge Discovery", Vol 1, No. 1, 1997.

[14] V. Hristidis and Y. Papakonstantinou, DISCOVER: Keyword Search in Relational Databases, In Proc. of VLDB Conference, 2002

[15] V. Poosala, Y. Ioannidis, P. Haas, E. Shekita. Improved Histograms for Selectivity Estimation of Range Predicates. Proc. of SIGMOD, 1996.

[16] F. Sebastiani, Machine learning in automated text categorization, ACM Computing Surveys, Vol. 34, No. 1, 2002.

[17] T. Zhang, R. Ramakrishnan and M. Livny. BIRCH: an efficient data clustering method for very large databases. Proc. of ACM SIGMOD Conference, 1996.