Seminar Report : Automatic Categorization of SQL-Query-Results

Abhijith Kashyap
rk39@cse.buffalo.edu

March 24, 2008

Abstract

Search queries on database-systems typically return *too many results* - many of them irrelevant to the user. This phenomenon is commonly referred to as information-overload, as the user expends a huge amount of effort sifting through the result-set looking for interesting results. This article reviews two approaches to tackling this problem. Both approaches are based on *categorization*; the query results are grouped into categories. These categories are then organized into a hierarchy forming a *navigation-tree*. The user traverses this tree, top-down, and chooses to view the results upon reaching the desired category.

1 INTRODUCTION

In recent years, there has been a tremendous increase in the amount of information stored by database-applications. Also, search-engine style exploratory queries are becoming a common phenomenon on these systems. These queries typically return a huge result-set. Only a small portion of the result is of interest to the user, who expends considerable effort searching for the relevant results.

In the internet text-search scenario, there has been two ways to tackle this problem - *ranking* and *categorization*. There have been attempts to adapt these solutions in the database-scenario. Ranking of database query results has been proposed in [3,4,5]. Work on SQL-Query-Result Categorization is rather recent and is the focus of this article.

A common approach for categorization, (followed by search engines, web-directories) involves around creating a *fixed* category structure. All data items are assigned category labels as well. At search time, items in the search-results are simply grouped by their category labels. Since the category structures are *independent* of the query, the distribution of query results on the category hierarchy tends to get skewed. For the same reason, fixed category structures tend to have longer navigation paths.

In this article, I survey the approaches

proposed to tackle the aforementioned problems in categorization. The first solution was proposed by [1]. A purported improvement to the approach in [1] is proposed in [2].
The rest of the article is organized as follows: In section 2, presents an overview of the two approaches. Section 3 compares the proposed solutions and examines their strengths and weaknesses, and conclude in section 4.

2 DISCUSSION

2.1 Approach:

Both [1] and [2], propose to create a *navigation tree* for a query q, dynamically at query time, based on query-result. The navigation tree recursive partitions the query results at each level, starting from the root. At each level, the partitioning is done based on a single attribute in the result relation. An attribute can be used for partition the result-set at most once. The partitions are assigned descriptive labels and form a categorization of the result-set based on that attribute.
The criteria for categorization is inferred, in both approaches, by analyzing the user behavior on the system - using the database query-log.
The motivation, for both approaches, is to reduce the effort on the part by the user in navigating query results. To capture this effort, they model the navigational *cost*, on average, faced by the user traversing the presented navigation tree. Both assume that users traverse the navigation tree, top-down, starting form the root. The cost consists of two components - the cost of examining category labels and the cost of examining query results.
Although the basic framework is the same, the two works differ in the following aspects - user navigation model, the cost model and cost estimation and the space for categorization; resulting in different navigation trees for same queries. This is discussed next.

2.2 Navigation Model:

In [1], the authors consider two distinct navigation scenarios - (1) ONE, the user is searching for a specific item and stops once she finds it and (2) ALL, the user browses through all the results by navigating to each node in the navigation tree. All other scenarios, user interested in "some" results is assumed to fall between these two scenarios. A given user, after examining the node's label, has three choices at any node:

1. SHOWRESULT: The user can choose to see all the tuples falling under the given node.

2. EXPLORE: User can drill down further into the hierarchy. This option is available only for non-leaf nodes.

3. IGNORE: User can ignore the node.

In [2], the authors assume that the user is interested in only a small sub-set of query result present the navigation tree as a set of hierarchical cluster over the result set.

2.3 Cost Model and Estimation:

The two different navigation models for the user in [1] have different cost models. To estimate the cost, the authors associate probabilities to each of the actions specified in the subsection 2.2 above and then build the navigation tree that minimizes the cost of reaching the first (ONE scenario) or all(ALL scenario) results. These probabilities are estimated by analyzing the query log. Details can be found in section 4.2 of [1].
In [2], the authors reduce the problem of building the optimal navigation tree to that of building an optimal decision tree [6]. Intuitively, the decision tree fits the description of the navigation tree provided in section 2.1. The *Information Gain* is modeled as the reduction in navigation cost caused by splitting the results by a given attribute.

3 CRITICAL REVIEW

In this section, the perceived advantages and disadvantages of the system are described:

3.1 Advantages

1. Both approaches are inherently better than the naive way of categorization - that of having a fixed category structure. The cost based approach reduces the information-overload faced by a user.

2. They provide a strong framework for future work in this area.

3.2 Disadvantages

1. Considerable time and effort is needed to generate and maintain the category structure especially in [2].

2. The navigation tree generated may confuse the user, especially in "complex" domains for. e.g. Bioinformatics.

3. The heuristics applied in [1] are unintuitive and may skew the navigation tree to generate trees with higher cost.

4. The heuristics in [1] do not consider the ONE scenario.

5. The over-simplified heuristics are also applied in [2], in assumption of *perfect trees*.

4 CONCLUSION

Both approaches can be considered *much* better than the original approach; that of a navigation hierarchy based on a fixed category structure. However, a considerable amount of effort is expended in creating and maintaining these category structures especially in case of [2]. The navigation trees generated may, at times, seem un-intuitive to the user.
Also, how well these systems to various domains remains to be seen.

REFERENCES:

[1] K. Chakrabarti, S. Chaudhuri, and S. won Hwang. Automatic categorization of query results. In SIGMOD, pages 755766,

2004.
[2] Z. Chen and T. Li. Addressing Diverse User Preferences in SQL-Query-Result Categorization. In SIGMOD, pages 641652, 2004.
[3] K. Chakrabarti, V. Ganti, J. Han, and D. Xin. Ranking objects based on relationships. In SIGMOD Conference, pages 371382, 2006.
[4] S. Chaudhuri, G. Das, V. Hristidis, and G. Weikum.Probabilistic ranking of database query results. In VLDB,pages 888899, 2004.
[5] G. Das, V. Hristidis, N. Kapoor, and S. Sudarshan. Ordering the attributes of query results. In SIGMOD,2006.
[6] J. R. Quinlan. Induction of decision trees. Machine Learning,1(1):81106, 1986.