STAR: A System for Tuple and Attribute Ranking of Query Answers

Nishant Kapoor
nkapoor@cise.ufl.edu

Gautam Das
gdas@cse.uta.edu

Vagelis Hristidis *
vagelis@cis.fiu.edu

S. Sudarshan
sudarsha@cse.iitb.ac.in

Gerhard Weikum
weikum@mpi-sb.mpg.de

Abstract

In recent years there has been a great deal of interest in developing effective techniques for ad-hoc search and retrieval in structured repositories such as relational databases - e.g., searching online databases of homes, used cars, and electronic goods. In many of these applications, the user often experiences "information overload", which occurs when the system responds to an under-specified user query by returning an overwhelming number of tuples, each displayed with a huge number of features (or attributes). We have developed a search and retrieval system that tackles this information overload problem from two angles. First, we show how to automatically rank and display the top-n most relevant tuples. Second, our system offers techniques for ordering the attributes of the returned tuples in decreasing order of "usefulness" and selects only a few of the most useful attributes to display.

1 Introduction

An increasing number of internet sites allow users to search a large inventory of objects (e.g., cars, homes, cameras) through a query interface. The user query is typically translated to a SQL query, which by nature is boolean; it either selects or rejects a database tuple. The SQL retrieval model is often inadequate for such exploratory search applications, and the user often experiences "information overload", which occurs when the system responds to an under-specified user query by returning an overwhelming number of tuples, each displayed with a huge number of features (or attributes). To illustrate these scenarios better, we use the following running example throughout the paper:

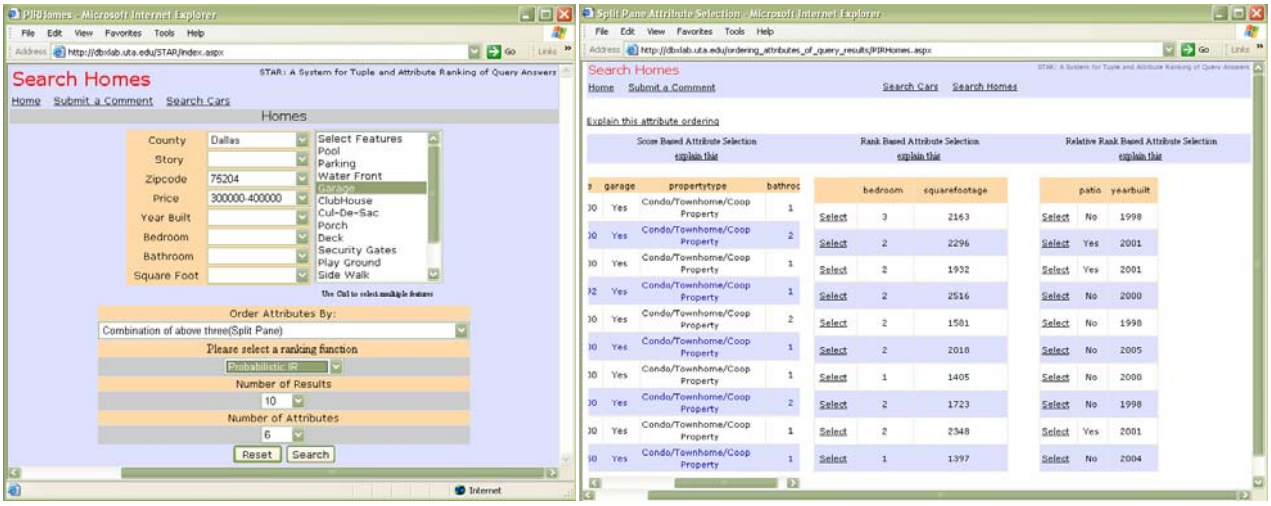
Example 1: *Consider an inventory database of an auto dealer, which contains a single table T with N rows and M attributes where each tuple represents a car for sale. The table has numerous attributes that describe details of the car, such as Price, Make, Model, Age, Zipcode, Mileage, EngineSize, NumCylinders, AccidentHistory, SecuritySystem, AirConditioning, and so on.*

*Partly supported by NSF grant IIS-0534530.

Current database query languages such as SQL follow the Boolean retrieval model, i.e., tuples that exactly satisfy the selection conditions laid out in the query are returned - no more and no less. While extremely useful for the expert user, this retrieval model is inadequate for ad-hoc retrieval by exploratory users who cannot articulate the perfect query for their needs - either their queries are very specific, resulting in no (or too few) answers, or are very broad, resulting in too many answers. In the example above, a simple conjunctive query such as "Select * from T where Model=sedan and Price $\leq$ 16000 and Mileage $\leq$ 20000" may overwhelm the user with too many result tuples.

In addition to too many result tuples, another source of information overload is the number of features (or attributes) of the result tuples that are returned. The number of attributes in typical automotive and home databases ranges from 25 to a hundred or more, as can be easily verified from cars or homes sales web sites. With such a large number of attributes, it is usually not possible to display all attributes of the answers to a query. Tabular displays of answers on web sites therefore typically display only a few attributes that are considered to be most "useful" to the user. However, the decision on what attributes to display is usually made manually, and fixed for all users, regardless of what attributes are likely to be useful to a particular user. Consequently, the answer tuples to a query may get displayed with a fixed set of attributes that do not reveal important relevant details, leading to a less-than-satisfying experience for the user.

We have developed STAR, a Web-based search and retrieval system that offers solutions to both the above problems. To address the limitations of the Boolean retrieval model for such queries, our system ranks the database query results as well as orders the attributes of the results. In particular, it computes a *score* of a tuple which represents the degree to which the tuple is "relevant" for the query, and return a few tuples with the best scores (e.g., top-n tuples where n is a small number such as 10 or 100) to the user. In addition, the system assigns a score to each attribute and displays only a few of the attributes with the best scores (e.g., top-m attributes, where m is a small number such



(a) Query page (b) Query Output

Figure 1. Sample Query in STAR

as $m \leq 15$). The ability to return only the most relevant tuples (and their most relevant attributes) enables the user to more effectively interact with the system. Furthermore, from a user interface design point of view, instead of having to display a long and wide table of results, we only need to display a much more compact result table.

Our system offers a variety of options for tuple ranking and attribute ordering[1] in order to better suit the profile of the user. We employ two tuple ranking methods to compute the top-n result tuples: (a) A simple *Similarity-based* ranking function which computes a (weighted) additive score between the query and each tuple, and (b) A *Probabilistic Information Retrieval-based* ranking (PIR-based) ranking function [1]. PIR-based ranking "extends" the original query by drawing on available knowledge of previous user preferences for the unspecified attribute values We also adopt the attribute ordering techniques developed in Das et al. [2], where given a query and a tuple ranking function for query results, the task is to return the top-m most useful attributes of the answer tuples. In our system an attribute is considered useful if it plays an influential role in the computation of the top-n ranked answer tuples of a query. This notion of attribute usefulness is quite broad, and several interesting variants are developed, each variant conveying different types of information. The developed variants (discussed in more detail in [2]) are: *Score-Based*, *Rank-Based*, *Relative Rank-Based*, *Split-Pane* and *Per-Tuple*.

Demo Overview Our Web-based system STAR, shown in Figure 1 and available at `http://dbxlab.uta.edu/STAR`, enables users to query a homes and a cars database, both downloaded from the Internet. The user can specify conditions on attributes such as square footage, number of bedrooms, etc., for the homes database, or mileage, year etc., for the cars database. The user can also select one of the available tuple ranking methods as well as one of the attribute ordering methods. At the results page the user can select any result to view its complete details.

[1]Note that we use the term "rank" for tuples and "order" for attributes to avoid confusion.

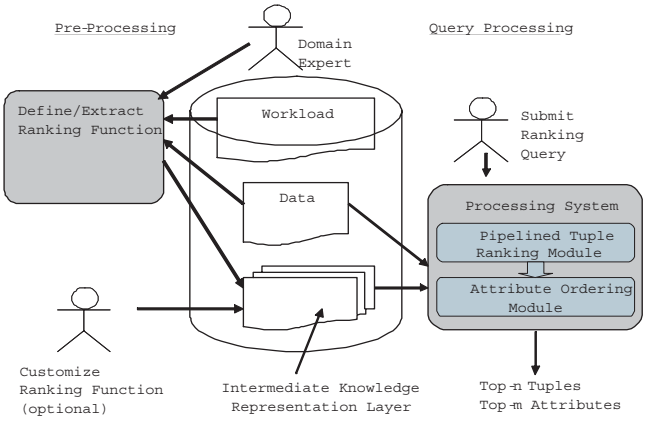


Figure 2. Architecture of the STAR System

System Architecture The backend of the STAR system has been developed in C# and the front-end in ASP, while the data is stored in Microsoft SQL Server 2000. The whole system including the database system and the Web Server run on a Pentium 4 2.8GHz system with 1GB of RAM. Figure 2 shows the architecture of the system.

References

[1] S. Chaudhuri, G. Das, V. Hristidis, and G. Weikum. Probabilistic ranking of database query results. In *VLDB*, 2004.

[2] G. Das, V. Hristidis, N. Kapoor, and S. Sudarshan. Ordering the attributes of query results. In *SIGMOD*, 2006.