CARE: An Automobile Crash Data Analysis Tool

The University of Alabama, with National Highway Transportation Safety Administration funding for traffic accident problem identification and evaluation, has developed the Critical Analysis Reporting Environment software system to analyze automobile crash data. CARE’s developers designed it for use by transportation safety engineers and policymakers, rather than trained statisticians.

To date, eight different states have applied CARE to crash data. The tool provides decision makers with the software environment they need to make effective decisions involving the development and enforcement of strategies to reduce crashes, without having to rely on statisticians and analysts to interpret that information.

CARE provides an intuitive interface that transportation safety engineers can use to obtain information from the responses to a fixed set of multiple-choice questions on automobile crash forms that law enforcement officials fill out. As such, CARE’s analysis domain is restricted to categorical data, represented by nominal or ordinal values such as type of crash, driver gender, and the county where a given crash occurred.

Even for noncategorical, quantitative information such as driver age or number of vehicles involved, CARE groups the values into ranges labeled with nominal values such as ages 25-34, 35-45, and so on. Periodic batch updates at regular, prespecified intervals insert crash data into a data set suitable for CARE processing. Thus, no transactional updates are applied to the data stored in CARE.

The restriction of the data to categorical values has implications for both the types of analysis conducted and the storage implementation of CARE-compatible data sets. Analysis features a heavy emphasis on counting.

Our user community frequently poses questions like the following:

- How many crashes occur in a given county on Friday, and how does this compare to the other days of the week?
- What time of day on Monday has the highest number of crashes among 16- to 24-year-old drivers?
- On what days and at what times do the highest number of pedestrian-related crashes involving alcohol occur?

The number of such questions can be made arbitrarily long. Queries regarding frequency of occurrence can be made about any of the variables collected in a traffic accident report. In most cases, a simple frequency distribution over one variable can answer questions such as “What is the worst time of day for accidents involving drivers 65 and older?” A cross-tabulation over two variables can also be used, such as “How many crashes occur during each hour of the day for every day of the week?”

The underlying data are strictly categorical, which makes implementing the query set both simple and efficient. CARE’s storage footprint is also quite small, making it possible to freely distribute...
data sets to a variety of platforms and creating a fully portable, self-contained system.

Because each variable is categorical, and a typical query has only a few variables, CARE can process such queries more efficiently than a typical database system or statistical analysis package. For example, a simple frequency distribution query over 1,000,000 records can return the answer in a few seconds when executed on an average Windows-based computer. Moreover, the system stores the records compactly and efficiently, requiring a small amount of disk space. Such efficiency is important because a particular investigation can lead to a chain of queries, determined in an ad hoc fashion.

APPLICATION STRUCTURE

Although CARE has been developed principally for the Windows platform, a version also exists that runs over the Web. A CARE data set consists of a single worksheet, consisting of rows or records and columns of attributes, which we also call variables. In this regard, CARE resembles Excel or common statistical packages such as SAS and SPSS. It does not support SQL or general relational database operations such as joins.

Users access restricted subsets of the data in a worksheet through a filter mechanism. This predefined subset of data is based on a Boolean expression over a subset of the variables in the worksheet. For example, a filter can be defined to consist of all records in which Driver Gender = Male and Driver Age is in the range 16...22, indicating all 16- to 22-year-old male drivers.

Filters are specified using a graphical user interface that constructs standard Boolean expressions. CARE then applies these expressions to analyze the data set. In particular, at any given time, exactly one filter is active. The default filter is the all filter, which includes every row in the original data set. The user can construct and store new filters, as well as change the active filter as many times as desired. The active filter determines the specific rows the system uses when performing an analysis.

TYPICAL USAGE SCENARIO

Either the frequency or cross-tabulation functions can count raw numbers of occurrences to analyze traffic crash data. The frequency tool provides a simple count of how often each value occurs in the domain of a categorical variable. CARE provides both tabular and graphical representations of the results.

Figure 1 shows a simple example of the GUI frequency output for a traffic accident database. In this example, the alcohol filter is active, and the GUI examines the day of week variable. In this case, all records have valid data in this variable. As with any typical relational DBMS system, we treat missing values as null values, which we can either include or exclude from the query. If the user chooses to include null values, and such values are present, an additional line labeled NULL appears in the frequency distribution and bar chart, along with a count of the null values.

The cross-tabulation tool provides a frequency analysis with respect to the cross product of two variables. Continuing with our alcohol filter, Figure 2 shows a cross-tabulation result that contains such an analysis with respect to the combination of the

Figure 1. A crash frequency result displayed in CARE’s graphical user interface.
Causal Driver Algorithm: A Value-Added Feature

Most states require an algorithm that defines the causal driver—the vehicle driver who most likely made the greatest contribution to a crash’s occurrence. The few crashes in which the causal driver cannot be defined receive a null assignment. Some states require the reporting officers to specify their opinion of the causal driver—which can be extremely valuable information.

CARE’s causal driver algorithm uses a simple and intuitive approach. First, analysts identify variables in the accident record that could indicate driver or vehicle causation. A team of experts then assigns a value between zero and 10 to each code within each of these variables: Zero indicates no causation and 10 indicates the highest level of causation. For example, variables that indicate the driver violated the law by, for example, failing to yield or driving under the influence of alcohol or drugs, usually receive high values. Variables that indicate lesser violations that may have contributed to the crash but did not cause it to receive lower values. The algorithm merely adds these metrics for each unit and designates the unit with the highest total value as the causal driver.

The causal driver algorithm provides an example of value added by the CARE system. Raw data is rarely provided in a form that can be used to produce immediately useful information. In most cases, the raw data must be edited, placed into categories, or otherwise preprocessed to render it useful. For example, to determine the importance value of Ambulance Delay Time, the Ambulance Arrival Time is subtracted from the lance Delay Time, the Ambulance Arrival Time is subtracted from the lance Delay Time.

While this transformation tends to mask the raw data’s details, we do not intend for CARE to meet all information needs completely. Other, more generalized packages exist for this purpose. Instead, CARE has repeatedly met its goal of helping practitioners satisfy at least 95 percent of their information needs.
the proportions, and we want to focus on those that are greatest. These assumptions might not hold for other applications, and some adjustment in the significance indicator would be needed.

Given these constraints, we recognize that there are two components to an overrepresentation:

- the degree or magnitude of the overrepresentation, and
- whether or not the overrepresentation is statistically significant.

Attribute values with overrepresentations that are both high in magnitude and statistically significant are substantially overrepresented. Although the threshold for statistical significance is always at the 99 percent level, the user can set the threshold for the magnitude of the overrepresentation.

The default overrepresentation threshold is 1.5. Conversely, an attribute value may be underrepresented. Just as Saturday was overrepresented for alcohol crashes by a factor of 2 in the preceding example, Saturday was underrepresented for non-alcohol crashes using alcohol crashes as a control subset. The degree of overrepresentation is 20 percent over 40 percent = 0.5, the inverse of the degree of overrepresentation for alcohol crashes.

An attribute that reflects a high degree of underrepresentation that is also statistically significant is said to be substantially underrepresented. Underrepresented data values typically represent values for which there is no problem—thus a county in which alcohol crashes are underrepresented would be a less likely candidate for selective alcohol enforcement.

This simple concept can be used to augment the frequency distribution shown in our typical usage scenario. In particular, by clicking the Significance box in Figure 3, the bars in the chart appear in red if a particular attribute is overrepresented, using the complement as a control subset. Red bars represent substantial overrepresentations, green bars represent substantial underrepresentations, and blue bars are neither substantially over- nor underrepresented. The threshold button lets users define thresholds for the magnitude of the overrepresentation to be deemed substantial. The default value is 1.5 for overrepresentations, while the default threshold for underrepresentations is 0.6667. The statistical significance test must also hold for an over- or underrepresentation to be considered substantial.

Figure 3 shows a frequency distribution for alcohol crashes by county, and for which the significance button has been checked. It shows that the tall bars representing counties with many alcohol crashes are not necessarily where the alcohol problems occur. In particular, Jefferson, the most populated county in Alabama, is actually substantially underrepresented with respect to alcohol crashes, as the green bar indicates. In contrast, some smaller counties, indicated with red bars, are substantially overrepresented with respect to alcohol crashes.

This simple tool is extraordinarily useful to decision makers with little or no statistical training.
Our user community intuitively wants to see the frequency distributions, and this audience finds using color to denote overrepresentation significance extremely intuitive. CARE hides the statistical details of the over- or underrepresentation calculations from the user, who sees only colors denoting problem and nonproblem values.

**IMPACT INFERENCE TOOL**

For users who need to understand additional statistical details regarding over- or underrepresentations, our Information Mining Performance Attainment Control Technique (IMPACT) provides a visual comparison between the experimental and control subsets.

Figure 4 shows an IMPACT run that compares experimental subset alcohol to the nonalcohol control subset, using the day of the week as the comparison variable. Saturday and Sunday are overrepresented, while the remaining days are underrepresented. In the bar graph, the light blue bars represent alcohol, while the burgundy bars represent nonalcohol. In the table, the red lines represent substantial over- and underrepresentations.

The table contains the raw frequencies in both subsets as well as the percentages of either subset for each value. The table contains two additional numeric values: Over Rep and Max Gain. An asterisk beside the Over Rep value denotes statistically significant over- and underrepresentations and appears regardless of whether the over- or underrepresentation is substantial. Max Gain shows the number of cases that would be reduced if the Subset Frequency were reduced to its expected value.

Overrepresentations often indicate problems that must be addressed through countermeasures, such as sobriety checks, public relations campaigns, new laws and restrictions, or safety devices. In these cases, a safety professional might ask, “What is the potential benefit from a proposed countermeasure?” The answer: A countermeasure is unlikely to reduce crashes to a level less than what is analogously found in the control group.

For example, in Figure 4, Saturday is overrepresented for alcohol crashes. The maximum benefit from implementing a countermeasure to reduce Saturday alcohol crashes would probably be no more than reducing Saturday alcohol crashes to the proportion found in Saturday nonalcohol crashes. Specifically, Saturday alcohol crashes could probably not be reduced to any less than 12.68 percent of the total alcohol crashes. Assuming the successful implementation of a countermeasure to effect such a reduction, approximately 1,031 alcohol crashes would be eliminated—the Max Gain for Saturday.

Max Gain provides a powerful metric that can aid the design of countermeasures. If a choice must be made between implementing two different countermeasures, Max Gain helps conduct a cost-benefit analysis. The countermeasure with the higher Max Gain value has the higher potential benefit.

Figure 5 shows a pragmatic application of Max Gain: an IMPACT run comparing alcohol versus nonalcohol crashes, executed on the County attribute. Because the County values sort in decreasing order of Max Gain, the counties that appear at the top of the list have the most potential to benefit from implementing alcohol programs. Assuming a politically neutral resource allocation, those counties should receive the most resources for implementing alcohol enforcement programs.
IMPACT also can define any arbitrary set as the control subset, rather than restricting the analysis to the original subset’s complement. Although the complement generally provides a good comparison set, defining another control subset often can be more useful. For example, in looking at the crashes in a particular city or county, choosing another specific city or county for comparison—rather than the complement of all remaining cities and counties—can be useful. IMPACT provides the comparative frequencies for the experimental and control subsets, side by side.

IMPACT AS A DATA MINING TOOL

IMPACT can also function as a data mining tool, using techniques designed to surface information from a database when users have little knowledge of the database and no a priori expectation of what types of conclusions can ultimately be drawn. Overrepresentations are surfaced for the data set at large, without users having to select a particular variable or set of variables.

To function as a data mining tool, IMPACT must run in profiling mode. In this mode, it surfaces overrepresentations from the data set, then orders these overrepresentations based on Max Gain. In profiling mode, the overrepresentations are no longer organized by variable, but aggregated among all variables. Figure 6 shows an example of using IMPACT for data mining to compare alcohol versus nonalcohol crashes.

Profiling surfaces the characteristics of a typical alcohol-related crash by ranking the overrepresented values in terms of Max Gain, displaying the highest Max Gain values first. The typical alcohol crash’s characteristics are at the top of the list: dark roadway, rural, male, open country, total damage to vehicle, injury-related crash, two-lane roadway, Saturday, no passing zone, and so on.

In this scenario, the user selected no variables, providing only information for defining the comparison subsets. Effectively, this represents a typical data mining paradigm, in which the user specifies little or no input information and the database automatically generates valuable information.

IMPLEMENTATION ISSUES

An inverted file structure provides the basis for a CARE data set’s physical view. Our notion of an inverted file differs somewhat from most existing concepts, which provide indexed access to information.

Ramez A. Elmasri and Shamkant B. Navathe defined fully inverted files to be files in which every variable has a separate index. In the information retrieval domain, analysts use inverted files as indexes to promote fast searches for keywords that appear in one or more documents. In particular, given a set of documents, an inverted file is a separate file that contains all keywords for all documents, with links from a particular keyword to all the documents in which that keyword appears.

While our notion of an inverted file shares the same goal of fast access to individual variables, our
Computer implementation inverts the database by storing the actual data in column order with the contents of each variable in a separate file, instead of the typical row order. That is, the file storage unit for a particular variable consists of a file containing a sequence of values such that

- the number of values equals the number of records in the data set's conceptual view, and
- the $i^{th}$ value in the FSU for variable $x$ is the value of $x$ for the $i^{th}$ record in the data set's conceptual view.

The physical view of a CARE data set therefore consists of a set of $n$ FSUs such that each FSU

- has the same number of values;
- these values equal the number of rows in the data set's conceptual view; and
- the aggregate of the $i^{th}$ values of each FSU, for all $i$, defines the $i^{th}$ row in the data set's conceptual view.

Table 1 shows this idea in a simplified context. CARE encodes physical values as bit strings in each FSU. An individual variable's domain size determines the number of bytes required for each FSU value. That is, if the number of unique values is less than or equal to 256—the number of unique bit strings representable in a single byte—then each FSU value is a one-byte quantity. Similarly, if the number of possible values lies between 257 and 32,765—the number of bit strings representable in two bytes—then each FSU value is a two-byte quantity.

A separate code table defines the domain size for each variable and provides a translation from database codes to a human-interpretable description of each category.

CARE's filters are defined by bit strings of length $n$, where the $i^{th}$ bit specifies in the obvious way whether the $i^{th}$ row is part of the analysis, with 0 indicating no and 1 indicating yes. Filters are created through the user interface, and users can specify any desired Boolean expression over any desired subset of data set variables. Users apply filters during the analysis as a mask that controls which values the system reads. In particular, the system only reads values from the FSUs that correspond to a position where the bitmap contains a 1. This can be done efficiently using built-in library functions that allow using array notation to access file positions. For example, the $i^{th}$ value can be retrieved from the FSU called gender just by referring to gender [i]. The system automatically buffers values from proximal locations whenever the program retrieves a particular element this way.

**PERFORMANCE**

Inverted files offer the primary advantage of improved performance. Except for data mining queries, most queries involve only a few variables. With inverted files, only those variables directly involved in the query must be read.
Consider the case of a simple frequency distribution without filters, such as the day-of-week distribution for all crashes. With a single frequency distribution, the system must open and read only the one FSU containing that frequency distribution variable—it can ignore the remaining data. If a query requires multiple frequency distributions, the system only needs to read the FSUs containing the variables for those frequency distributions.

Similarly, for a crosstab query, the system reads only the two FSUs that contain the two variables in the crosstab. In general, this storage scheme can completely ignore all variables that it does not need for a particular computation.

Filters require an additional initial computation to generate the filter bitmap. This must be read from \( k \) FSUs, where \( k \) equals the number of variables in the Boolean expression that defines the filter. In most cases, \( k \) is quite small compared to the total number of variables, thus substantially limiting the amount of additional data the system must read.

In general, the system reads a small number of FSUs for a particular query. This means that CARE—unlike other statistical analysis applications and relational database management systems that store rows of data contiguously in a single file—can more efficiently process queries and produce results. In queries that involve a large number of variables, such as some IMPACT runs that mine a large portion of the variables in the data set, the performance does worsen. This occurs because the amount of data the system must read in these queries approaches that which row-contiguous system queries must read.

However, CARE amortizes the performance costs: Users only pay for the data they use, as a detailed comparison between CARE’s storage format and the typical row-major format commonly associated with other applications shows.6

We designed CARE’s data sets for distribution on stand-alone Windows platforms, making them application-independent of network or server support. The data’s locality further enhances performance. Using our data encoding, data sets tend to be quite small: The Alabama crash data set for the year 2000 contains nearly 150,000 records and 300 variables, yet occupies only a bit more than 40 Mbytes in the CARE environment.

In addition to the stand-alone platform, we provide an auto-updating system for computers connected to networked servers. Changes in the data, software version, or filter library automatically propagate to the desktop.

RELATED WORK

CARE relates to several efforts in the general area of data warehousing.7,8 Most existing data warehouse systems function as online analytical processing systems. OLAP systems fall into several categories, with relational OLAP and multidimensional OLAP chief among them.

ROLAP systems store data for analysis in a relational database, frequently in a star schema. MOLAP systems use a special multidimensional database for this storage. MOLAP systems compute and store intermediate results in a disk-based structure modeled after the idea of a multidimensional array. Typically, business intelligence tools are then developed to provide high-level, GUI-based analyses against the ROLAP and MOLAP data structures. Developing and maintaining such a data warehouse involves significant overhead, which is generally only considered to be worthwhile in situations that involve a large amount of data.

Developers typically view OLAP systems as a relatively heavyweight investment. The underlying databases require major server support. Business intelligence tools that run against such databases must be implemented in a networked, client-server environment that requires substantial network and communications overhead. Significant effort goes into the query optimization of such systems9-11 to improve their performance. Additionally, the tools used in this environment are typically quite complex, providing extremely powerful functionality but also requiring extensive user training.

Compared to such systems, CARE is quite lightweight, which makes it most accurately classified as a data mart.7,8 CARE stores its data locally, thus eliminating network and communications overhead. Given its minimal overhead, implementing a CARE-based data analysis system costs much less than a typical large-scale data warehousing system. Although this provides correspondingly less functionality, CARE is sufficiently functional to meet its user population’s needs. While some highway
crash data analysts might require different statistical operations, such operations would still be based fundamentally on analyzing frequency distributions involving the types of nominal variables we’ve described.

CARE has proven to be a successful application in the traffic safety community for two reasons: its simplicity and efficiency. It is currently being used in several states, including Alabama, Delaware, Florida, Iowa, Michigan, North Carolina, Rhode Island, and Tennessee. CARE received the 1995 Administrator’s Award from the National Highway Transportation and Safety Administration. The Federal Aviation Administration and NASA also have used it to investigate aviation incidents and accidents, for which the data have the same conceptual and semantic structure as highway crashes.

References

Allen S. Parrish is an associate professor and director of the CARE Research & Development Laboratory in the Department of Computer Science at the University of Alabama. His research interests include software engineering, highway safety information systems, and data warehousing. Parrish received a PhD in computer science from Ohio State University. Contact him at parrish@cs.ua.edu.

Brandon Dixon is an associate professor in the Department of Computer Science at the University of Alabama. His research interests include software engineering, theoretical computer science, and highway safety information systems. Dixon received a PhD in computer science from Princeton University. Contact him at dixon@cs.ua.edu.

David Cordes is an associate professor and department head in the Department of Computer Science at the University of Alabama. His research interests include software engineering, networking, and software systems. Cordes received a PhD in computer science from Louisiana State University. Contact him at cordes@cs.ua.edu.

Susan Vrbsky is an associate professor in the Department of Computer Science at the University of Alabama. Her research interests include database systems and real-time systems. Vrbsky received a PhD in computer science from the University of Illinois. Contact her at vrbsky@cs.ua.edu.

David Brown is a professor in the Department of Computer Science at the University of Alabama. His research interests include highway safety information systems and software engineering. Brown received a PhD in industrial engineering from Texas Tech University. Contact him at brown@cs.ua.edu.