Utility-based Resolution of Data Inconsistencies

Amihai Motro, Philipp Anokhin, Aybar C. Acar
(ami, panokhin, aacar)@gmu.edu
Information and Software Engineering Department
George Mason University

ABSTRACT

A virtual database system is software that provides unified access to multiple information sources. If the sources are *overlapping* in their contents and *independently* maintained, then the likelihood of *inconsistent* answers is high. Solutions are often based on ranking (which sorts the different answers according to recurrence) and on fusion (which synthesizes a new value from the different alternatives according to a specific formula). In this paper we argue that both methods are flawed, and we offer alternative solutions that are based on knowledge about the *performance* of the source data; including features such as recentness, availability, accuracy and cost. These features are combined in a flexible *utility* function that expresses the overall value of a data item to the user. Utility allows us to (1) define meaningful ranking on the inconsistent set of answers, and offer the top-ranked answer as a preferred answer; (2) determine whether a fusion value is indeed better than the initial values, by calculating its utility and comparing it to the utilities of the initial values; and (3) discover the best fusion: the fusion formula that optimizes the utility. The advantages of such performance-based and utility-driven ranking and fusion are considerable.

1. INTRODUCTION

Occasionally, two or more independent sources of information (e.g., databases) may contain values that purport to represent the same real world value. If these values are different, the representations are said to be *inconsistent*. In systems that integrate information from multiple, independent information sources, often known as *virtual* database systems, some queries may thus result in answer that are inconclusive. Hence, the issue of data inconsistency poses a significant challenge all such systems.

Note that we assume that the inconsistencies are not due to differences in *semantics*; e.g., the use of different units of measurement or different notations for telephone numbers. In other words, we assume that all semantic inconsistencies have already been reconciled. Note also that we assume that the answer should be a single value; that is, it is not possible for several different values to be correct.

iqis-2004 Paris, France

1.1 A Classification of Solutions

Confronted with an inconsistent set of answers, one can offer at least five solutions: (1) multi-answer, (2) ranked answer, (3) preferred answer, (4) random answer, and (5) fused answer.

A *multi-answer* is simply the complete and inconsistent set of alternative answers. In the area of logic databases, it is also referred to as a *disjunctive* answer. Multi-answers may be regarded as providing users with "raw" information, leaving them with the task of interpreting the results "outside" the system.

A *ranked answer* still offers the complete set of alternatives, but the alternatives are now ranked according to their likelihood of being the correct answer. Rankings are usually based on *recurrence*. In the inconsistent set of answers obtained from multiple sources some answers could be identical. In general, identical answers tend to reinforce the likelihood of their value being the correct value. Consequently, rankings are derived from a process similar to *voting*; that is, an answer provided by n sources is ranked above an answer provided by $n-1$ sources. Clearly, multi-answers can be regarded as a special case of ranked answers (when the different alternatives are ranked identically).

Often a unique answer is required, and the set of alternatives must be reduced to a single answer. We observe three possible solutions:

If ranking is possible, then the top-ranked answer would be offered as the *preferred answer*. Since rankings are based on votes, the preferred answer is simply the most popular answer.

A *random answer* is a single answer from the set, selected at random. It is useful when the differences among the alternative values is judged to be inconsequential, or when there is insufficient information for ranking (but a single answer is still required).

An underlying assumption of the four types of answers discussed so far is that the *true* value is one of the values in the set. A *fused answer* is an answer obtained by combining the individual answers into a single value; a value that may be different from each of the alternatives. Usually, fusion would be used for numerical data, such as averaging multiple measurements, but could possibly be used for non-numerical data as well, such as combining two incomplete addresses into a single address. Usually, fusion is performed according to an "expert formula": a formula specified by an expert

familiar with the sources.

1.2 Current Weaknesses

Because multi-answers require no decisions, and random answers apply arbitrary decisions, both may be considered naive solutions that deserve no further investigation. On the other hand, the other three types of answers have serious limitations.

Because ranked and preferred answers are based on recurrence, they are useful only when the cardinality of the set of answers is high and there is a high degree of recurrence. When the set of alternative answers is small or when the recurrence rate is low, ranking is entirely unreliable. For example, when there are two or three values for the age of a person, or when there are many values for the salary of a person, all different, ranking is not meaningful.

As mentioned above, fusion is usually performed according to an expert formula. Beyond this general promise, however, there is no proof that the fused value is *optimal* in any sense; indeed, there is no proof even that it is *better* than any of the alternative values.

1.3 Features and Utility

Information sources vary greatly with respect to many parameters, such as their availability, reliability, and other performance measures. Yet, the methods discussed so far do not take any such considerations into account. In this paper we advocate the use of such performance measures in the resolution of answer inconsistencies. Specifically, we assume

- A set of performance parameters; we refer to these as data *features*.
- Each alternative answer is associated with a value for each of these features.
- A utility function which is a linear combination of the features. This function expresses the overall value of an individual answer to the user.

These assumptions permit us to pursue three important goals.

First, we may define ranking that is based on utility rather than recurrence. For each individual answer we may calculate the utility and then sort the answers according to their utilities. Such ranked answers are useful even when the set of alternatives is small (such as two or three values).

Second, when creating a fusion, we may calculate the utility of the new value, and determine whether it is indeed better than the available alternatives.

Third, we can address the issue of *optimal* fusion. A fusion formula is a set of coefficients; varying these coefficients generates different fusion values, each with its own utility. The utility of the fusion is therefore a function of the fusion coefficients. In optimizing this function we search for a combination of coefficients (the fusion formula) that maximizes the utility.

Altogether, we claim that the advantages of performance-based and utility-driven ranking and fusion are considerable.

Section 2 discusses six specific data features that might impact the construction of answers in the presence of inconsistencies. Two approaches that rely on such metadata, ranking and fusion, are defined in Section 3. Section 4 discusses the construction of ranked, preferred and fused answers given the assumptions stated earlier. In particular, it shows how to derive each of the data features of fusion values, and hence how to calculate the utility of the fusion. Section 5 discusses the problem of optimal fusion and demonstrates it with an example. Section 6 reviews related research, and Section 7 concludes with a summary and discussion of future work.

2. DATA FEATURES: KNOWLEDGE ABOUT DATA

Knowledge about data is usually referred to as *metadata*. There are many kinds of metadata, and we shall consider only those that might impact the construction of answers in the presence of inconsistencies.

In particular, we discuss six kinds of metadata, which we shall refer to as *features* of the data. They are: (1) recentness, (2) cost, (3) accuracy, (4) availability, (5) priority, and (6) quality. A brief discussion of each of these features follows.

Recentness. Recentness refers to the *time* in which the information was published. When resolving conflicts, newer information is usually preferred. Arguably, recentness is the most common method for resolving information conflicts. We shall use t to denote the feature of recentness.

Cost. When alternative answers are available, an important decision factor could be the cost of materializing each answer; for example, when two inconsistent answers are offered, one may want to choose the cheapest answer (especially when the difference in their costs is significant). When the providing sources charge for their services, cost may refer to the actual fees required. In a networked environment, cost may refer to the time necessary to retrieve (download) each answer. Or a complex cost formula may be devised that will combine different cost components. Cost will be denoted with c.

Accuracy. Often, the data in databases is only an approximation of the true data. When information about the goodness of the approximation is recorded, the results obtained from a database can be interpreted more reliably. Usually, when considering a set of alternative values (all of which are approximations of the same true value), the value with the highest level of accuracy would be preferred. We shall denote accuracy with s.

There are many methods for recording the accuracy of an approximation. As a simple example, numerical values could be associated with a range, such as $\pm 5\%$. A description of our preferred model for representing accuracy follows.[1]

Assume that each database value is associated with a *degree of accuracy*. This accuracy is denoted with a probability density function, whose mean is the value itself. Formally, each database "value" is indeed a random variable; the mean of this variable becomes the *stored* value, and is interpreted as an approximation of the *true* value; the standard deviation of this variable is a measure of the level of accuracy of the stored value. We shall assume that all probability density functions have *normal* distributions.

Low standard deviation (a sharply peaked curve) indicates that the probability that the true value is close to the mean (the stored value) is high; this probability declines rapidly as the true value shifts from the mean. Hence, lower standard deviation corresponds to higher accuracy. Conversely, high

[1]Indeed, inaccuracy is just one of many known forms of uncertainty and imprecision.

standard deviation (a flat curve) indicates that the probability that the true value is close to the mean (the stored value) is low; this probability declines slowly as the true value shifts from the mean. Hence, higher standard deviation corresponds to lower accuracy. Indeed, as the standard deviation increases, the distribution approaches a *uniform* distribution, in which all values assume identical probabilities. Such database values would have the lowest level of accuracy.

Availability. Availability is the probability that an answer promised by a particular information source would be retrievable when needed. In a network environment it measures the robustness of the route to the source as well as the robustness of the server on which the information resides. When considering alternative answers from multiple sources, their respective availabilities is an important factor. Availability will be denoted with v.

Priority. Priority simply indicates a preference for particular sources. This preference may be derived from their past performance. Alternatively, sources may be associated with a level of *credibility* or *authority* or *trust* that was granted by a certifying agency. We shall denote priority with p.

Quality. Quality refers to a level of *performance* of the data that is *assured* by the source; that is, any specification which the data is warranted to meet or exceed. Whereas other features, such as accuracy or cost, are associated with distinct measurements, this feature can be used when the source can only provide a *lower bound* which is guaranteed to hold. Possibly, it could overlap with other features. For example, in place of or in addition to the feature of accuracy, a source may provide a minimal level of accuracy that each data value is guaranteed to meet.

Similarly, one could define a dual feature that is measured by *upper bounds*; that is, a specification that is guaranteed *not to exceed* the specified value.

All these features assume numerical values, and, for simplicity, we shall assume that the range of each feature is between 0 and 1. Moreover, each of these numerical values indicates a "level of performance" with respect to the feature, and we shall assume that in each case 1 is the highest level of performance (the best), and 0 is the lowest level of performance (the worst). The advantage of features that are "normalized" to have the range $[0, 1]$ is that it is easier to define "combination" features. Assume, for example that we wish to define a combination feature which is half cost and half accuracy. If cost and accuracy are expressed in their original (unnormalized) ranges, finding the correct weights that express "half and half" is considerably more difficult.

Features may be assigned to data items of different *granularity*. In general, each individual database value could have its own feature values. At times, all the values in a column (attribute) or in a row (tuple) could have the same feature value. Furthermore, the entire database could have the same feature value.

In specific situations, some of these features may not apply, or there may be other useful features that have not be mentioned here. It should be emphasized, though, that in this paper we do not argue for the appropriateness of individual features; rather, we argue for an approach to the resolution of data inconsistencies that is based on the relative merits of the alternative values.

3. RANKING AND FUSION

To construct ranked, preferred and fused answers we must provide formal definitions for ranking and fusion. In both cases we shall employ linear combinations.

3.1 Ranking

We shall assume ranking policies that are linear combinations of features. Let $\{f_1, f_2, \ldots, f_m\}$ be the set of features. A *ranking policy* is an expression $w_1 \cdot f_1 + w_2 \cdot f_2 + \cdots + w_m \cdot f_m$, where $0 \leq w_i \leq 1$ and $\sum_{i=1}^{m} w_i = 1$. The ranking of a specific data value is calculated by substituting its specific feature values for $f_1, f_2, \ldots, f_m$.

Note that the effect of selecting $w_i = 0$ is to ignore the feature f_i in the ranking policy. In addition to the obvious purpose of allowing users to indicate their complete lack of interest in the specific feature, we shall assume that ranking policies are adjusted automatically to remove any feature when it is not possessed by *all* the alternative values. We shall refer to the set of values x_i for which $w_i \neq 0$ as the features *participating* in the ranking.

Linear ranking expressions, while powerful, are not the most general. For example, they cannot express the ranking policy "if availability is below 0.5, then ignore this feature altogether in the ranking."

As ranking policies express the overall value to users, they can be interpreted as definitions of *utility*. Therefore, we shall also refer to a ranking policy as a *utility function* and to the calculated ranking of a data value as the *utility* of that value.

3.2 Fusion

Given a set of alternative values (approximations) $\{x_1, x_2, \ldots, x_n\}$, fusion is an attempt to combine them into a value that is closer to the correct value than any of the given values. In this sense, fusion creates an $(n+1)$'th approximation by means of synthesis.

In this paper we limit our discussion to fusions of *numerical* data values.

Let $\{x_1, x_2, \ldots, x_n\}$ be a set of independent values all representing the same true value. The *fusion* of $x_1, x_2, \ldots, x_n$ is a linear combination $a_1 \cdot x_1 + a_2 \cdot x_2 + \cdots + a_n \cdot x_n$ where $0 \leq a_i \leq 1$ and $\sum_{i=1}^{n} a_i = 1$.

The set of all the fused valued obtained by trying all the possible coefficients that satisfy $0 \leq a_i \leq 1$ and $\sum_{i=1}^{n} a_i = 1$ is the interval $[\min\{a_1, \ldots, a_n\}, \max\{a_1, \ldots, a_n\}]$. This corresponds to our intuition that the true value lies somewhere in this range.

Note that the effect of selecting $a_i = 0$ is to remove x_i from the fusion process. This may be advantageous in numerous cases, for example when the availability of x_i is low, or when its cost is high. We shall refer to the set of values x_i for which $a_i \neq 0$ as the values *participating* in the fusion.

Again, linear fusion expressions are powerful, but not the most general. For example, they cannot express the fusion policy "if x_1 is a statistical outlier,[2] then ignore it altogether in the fusion process".

4. CONSTRUCTING ANSWERS

4.1 Ranked and Preferred Answers

[2]Its value deviates significantly from the mean of the other values.

The process of constructing ranked and preferred answers is straightforward: we calculate the utility of each alternative value x_i according to the prevailing utility function. The values are then sorted according to their utility (to create a ranked answer) or the value with the highest utility is chosen (to create a preferred answer).

Note that preferred answers that are derived from ranking-by-recurrence (voting) require that all the alternative values be retrieved apriori. In distinction, preferred answers that are based on utility are derived from metadata, and require the retrieval of the winning value only.

4.2 Fused Answers

Constructing a fused answer is also straightforward. A more complex issue is to derive the features of this new value. Once these features have been derived, the overall utility of the fused value can be calculated. This utility should then be compared to the utilities of the initial values, to determine whether any improvement has been attained.

Consider the fusion $x = a_1 \cdot x_1 + a_2 \cdot x_2 + \cdots + a_n \cdot x_n$. Assume that k of the n coefficients are positive. Without loss of generality we may assume that the positive coefficients are $a_1, a_2, \ldots, a_k$; i.e., the participating values are $x_1, x_2, \ldots, x_k$. The derivation of the various feature values of x is discussed below.

In deriving each feature value we must address the issue of appropriate ranges. As already mentioned, we shall assume that all feature values are in the range $[0, 1]$, with 1 corresponding to the best value and 0 corresponding to the worst. For each of the six features we must show how actual values are mapped into this range, and we must ensure that the value assigned to their fusion is in this range as well. However, we shall only be concerned with the *specific* fusion at hand. For example, assume an answer set with three alternative values x_1, x_2, and x_3, assume that their individual values for some feature are c_1, c_2, and c_3, and assume that their fusion should be assigned the feature value $c = c_1 + c_2 + c_3$; then we shall only be concerned that the four values c_1, c_2, c_3 and c are mapped appropriately.

4.2.1 Recentness

A prevailing method for indicating recentness is to use *timestamps*, which are usually clock values, and we shall assume that this method is used. A range of "active" timestamps is created from the timestamps of the multi-answer and the *present* timestamp (the current clock value). To map this range to the interval $[0, 1]$, the oldest timestamp in the set is mapped to 0, and the present timestamp is mapped to 1. Intermediate timestamps are interpolated linearly into this range.

When a set of values is combined to a new value, the timestamp of the new value is always the current clock value. Hence, the new value receives the recentness value 1:

$$t(x) = 1$$

Obviously, the new value is in the range $[0, 1]$.

The new recentness value reflects the time in which the new value was created, rather than the recentness values of the alternatives. As an example, assume two values x_1 and x_2 with recentness t_1 and t_2, respectively, and assume the fusion $1 \cdot +x_1 + 0 \cdot x_2$. The new value is equal to x_1, yet its recentness is not t_1 but 1. This is because the new value reflects a decision that is reached at the *present* time. A current decision, which is based, presumably, on current knowledge, justifies an *update* of the timestamp. Note that the feature of the fused value is *independent* of the features of the available values.

4.2.2 Cost

To create the fusion value x, the participating values must all be retrieved. Hence the cost of fusion is the *sum* of the costs of the participating values.

To guarantee that all costs and the sum of the costs are in the range $[0, 1]$, we normalize each cost by dividing it by the sum of the costs of all the given values; so that higher costs are associated with lower values we subtract this result from 1. Let $c_r(x_i)$ denote the "raw" cost of retrieving x_i (for example, in dollars or in seconds of download time); the normalized cost is

$$c(x_i) = 1 - \frac{c_r(x_i)}{\sum_{i=1}^{n} c_r(x_i)}$$

As indicated above, the raw cost of the fusion is simply the sum of the raw costs of the participating values:

$$c_r(x) = \sum_{i=1}^{k} c_r(x_i)$$

Hence, the normalized cost of the fusion is

$$c(x) = 1 - \frac{\sum_{i=1}^{k} c_r(x_i)}{\sum_{i=1}^{n} c_r(x_i)}$$

Which can be expressed, using normalized costs only:

$$c(x) = 1 - \sum_{i=1}^{k} (1 - c(x_i))$$

It can be shown that if $k = n$ (all values participate) then $c(x) = 0$ (the cost is the highest possible). The latter formula can also be written as

$$c(x) = 1 - \sum_{i=1}^{n} \lceil a_i \rceil \cdot (1 - c(x_i))$$

where the cost of the fusion is a function of all n costs and coefficients.[3]

4.2.3 Accuracy

If each of the participating database values has a normal distribution with mean x_i and standard deviation σ_i, then the fusion $x = a_1 \cdot x_1 + a_2 \cdot x_2 + \cdots a_k \cdot x_k$ would have a normal distribution with parameters

1. Mean: $\sum_{i=1}^{k} a_i \cdot x_i$
2. Standard deviation: $\sqrt{\sum_{i=1}^{k} a_i^2 \cdot \sigma_i^2}$

The mean of the fusion corresponds to the fused value itself and the standard deviation of the fusion is its accuracy. It can be shown that the standard deviation of the fusion lies between the lowest and the highest standard deviations.

To obtain accuracy values in the range $[0, 1]$, we divide each standard deviation by the highest standard deviation; so that higher standard deviations (lower accuracies) are associated with lower values, we subtract this result from 1.

[3] $\lceil a_i \rceil$ may be read "if $a_i = 0$ then 0 else 1".

Let σ_i denote the standard deviation of x_i; the accuracy of x_i is

$$s(x_i) = 1 - \frac{\sigma_i}{\max_{i=1}^{n} \sigma_i}$$

As indicated above, the standard deviation of the fusion is

$$\sqrt{\sum_{i=1}^{k} a_i^2 \cdot \sigma_i^2}$$

Hence, the accuracy of the fusion is

$$s(x) = 1 - \frac{\sqrt{\sum_{i=1}^{k} a_i^2 \cdot \sigma_i^2}}{\max_{i=1}^{n} \sigma_i}$$

Which can be expressed, using accuracies only:

$$s(x) = 1 - \sqrt{\sum_{i=1}^{k} a_i^2 \cdot (1 - s(x_i))^2}$$

The latter formula can also be written as

$$s(x) = 1 - \sqrt{\sum_{i=1}^{n} a_i^2 \cdot (1 - s(x_i))^2}$$

where the accuracy of the fusion is a function of all n accuracies and coefficients.

4.2.4 Availability

Availability is the probability that at a random moment the particular information source is available. This feature needs no mapping as it is already in the range $[0, 1]$.

To create the fusion value x, all the sources for the participating values must be available. Assuming that the availabilities of these sources are *mutually independent*, the availability of x is the product of the availabilities of the sources:

$$v(x) = \prod_{i=1}^{k} v(x_i)$$

It is easy to verify that $v(x)$ is in the range $[0, 1]$. The latter formula can also be written as

$$v(x) = \prod_{i=1}^{n} \max\{v_i(x), \lfloor (1 - a_i) \rfloor\}$$

where the availability of the fusion is expressed as a function of all n availabilities and coefficients.[4]

4.2.5 Priority

Without loss of generality, we assume that priority is stated in six levels, which are integers between 0 and 5, with 5 corresponding to the highest priority and 0 corresponding to the lowest. These are mapped trivially to the range $[0, 1]$, with 1 corresponding to highest priority and 0 corresponding to lowest priority.

When a new value is synthesized from a set of database values using a linear formula, and each database value is associated with its own priority, it is intuitive to use the same formula for synthesizing the priorities. In other words, the priorities are synthesized using the same proportions that were used in the fusion:

$$p(x) = \sum_{i=1}^{k} a_i \cdot p(x_i)$$

It is easy to verify that $p(x)$ is in the range $[0, 1]$. The latter formula can also be written as

$$p(x) = \sum_{i=1}^{n} a_i \cdot p(x_i)$$

where the priority of the fusion is expressed as a function of all n priorities and coefficients.

[4] $\max\{v_i(x), \lfloor (1 - a_i) \rfloor\}$ may be read "if $a_i = 0$ then 1 else $v_i(x)$".

feature type	feature value
recentness	$t(x) = 1$
cost	$c(x) = 1 - \sum_{i=1}^{n} \lceil a_i \rceil \cdot (1 - c(x_i))$
accuracy	$s(x) = 1 - \sqrt{\sum_{i=1}^{n} a_i^2 \cdot (1 - s(x_i))^2}$
availability	$v(x) = \prod_{i=1}^{n} \max\{v(x_i), \lfloor (1 - a_i) \rfloor\}$
priority	$p(x) = \sum_{i=1}^{n} a_i \cdot p(x_i)$
quality	$q(x) = \min_{i=1}^{n} \{\max\{q(x_i), \lfloor (1 - a_i) \rfloor\}\}$

Table 1: Fusion features.

4.2.6 Quality

Without loss of generality, we assume that quality is stated in eleven levels, which are integers between 0 and 10, with 10 corresponding to the highest priority and 0 corresponding to the lowest. These are mapped trivially to the range $[0, 1]$, with 1 corresponding to highest priority and 0 corresponding to lowest priority.

The nature of threshold specifications suggests that the quality of the fusion is the *lowest* of the quality thresholds of the participating values:

$$q(x) = \min_{i=1}^{k} \{q(x_i)\}$$

It is easy to verify that $q(x)$ is in the range $[0, 1]$. The latter formula can also be written as

$$q(x) = \min_{i=1}^{n} \{\max\{q_i(x), \lfloor (1 - a_i) \rfloor\}\}$$

where the quality of the fusion is expressed as a function of all n qualities and coefficients.[5]

If a dual feature is used that guarantees that the data value would not exceed an upper bound, then the feature of the fusion would be the *highest* of the features of the participating values.

4.3 The Utility of the Fusion

Consider a fusion $x = a_1 \cdot x_1 + a_2 \cdot x_2 + \cdots + a_n \cdot x_n$. Table 1 summarizes the features of this fusion. Let $u = w_1 \cdot t + w_2 \cdot c + w_3 \cdot s + w_4 \cdot p + w_5 \cdot a + w_6 \cdot s$ be the utility function.

The utility of the fusion is

$$u(x) = w_1 \cdot t(x) + w_2 \cdot c(x) + w_3 \cdot s(x) + w_4 \cdot p(x) + w_5 \cdot v(x) + w_6 \cdot q(x) \quad (1)$$

where $t(x), c(x), s(x), p(x), v(x), q(x)$ are calculated as in Table 1.

[5] $\max\{q_i(x), \lfloor (1 - a_i) \rfloor\}$ may be read "if $a_i = 0$ then 1 else $q_i(x)$".

An obvious conclusion apparent in Table 1 is that the features of a linear fusion are derived from the features of the participants in a variety of ways, depending on the particular feature. Indeed, our discussion avoided additional features, if their fusion was too similar to the fusion of features already considered.

5. THE BEST FUSION

Our approach to fusion was to assume that an "expert" would provide the specific values for the coefficients $a_1, \ldots, a_n$ that are used to combine the given approximations $x_1, \ldots, x_n$. The fusion process would be considered successful if and only if the utility of the fused value exceeds the utility of each of the individual approximations: $u(x) > \max_{i=1}^{n} u(x_i)$.

Still, the coefficients provided by an expert may prove to be successful, but may not deliver the *highest* utility possible by fusion. In other words, we would like to select the fusion (i.e., the fusion coefficients a_i) that would *maximize* the utility.[6]

As an intuitive illustration, assume that one receives two predictions on the time that an approaching hurricane will make landfall: 36 hours and 24 hours. Assume that the first estimate was issued at 6 am by a source of high authority, whereas the second estimate was issued at noon by a source of lesser authority. Intuitively, one would conclude that landfall is likely to occur between 24 and 36 hours. The actual estimate to be adopted will be impacted by the relative importance assigned to the recentness of an estimate vs. the authority of the source that issued it (the utility). Thus, if one values authority over recentness, one will choose a value closer to 36 (and vice versa). Our approach is to find the value between 24 and 36 that optimizes the utility.

The utility of the fusion combines six individual features, but the effect of the fusion coefficients on each feature is different, and we observe three cases. One feature, recentness, is not affected by the selection of the coefficients. In the case of cost, availability or quality it only matters whether a coefficient is zero or positive (i.e., the actual value of a positive coefficient is irrelevant). Finally, in the case of accuracy or priority the actual value of a positive coefficient is important.

In Equation 1 the utility of the fusion was expressed as a function of the values x_i (the coefficients a_i and the weights w_i were constants). We now assume that the values x_i and the weights w_i are constants and treat the utility of the fusion as a function of the coefficients a_i. In this function, the features of each x_i are constants as well, and, for clarity, are denoted $t_i, c_i, s_i, v_i, p_i, q_i$, rather than $t(x_i), c(x_i), s(x_i), v(x_i), p(x_i), q(x_i)$.

[6]Of course, such optimization is meaningful only when the utility of the fusion is dependent on the coefficients a_i. Thus, if the only feature under consideration is recentness, a feature not dependent on the coefficients, then all linear fusions have the same utility.

$$
\begin{aligned}
u(a_1, a_2, \ldots, a_n) &= w_1 \cdot 1 \\
&+ w_2 \cdot (1 - \sum_{i=1}^{n} \lceil a_i \rceil \cdot (1 - c_i)) \\
&+ w_3 \cdot (1 - \sqrt{\sum_{i=1}^{n} a_i^2 \cdot (1 - s_i)^2}) \\
&+ w_4 \cdot \prod_{i=1}^{n} \max\{v_i, \lfloor (1 - a_i) \rfloor\} \\
&+ w_5 \cdot \sum_{i=1}^{n} a_i \cdot p_i \\
&+ w_6 \cdot \min_{i=1}^{n} \{\max\{q_i, \lfloor (1 - a_i) \rfloor\}\}
\end{aligned}
\tag{2}
$$

There are numerous techniques and software packages that can be used to maximize $u(a_1, a_2, \ldots, a_n)$ (for example, Matlab, Mathematica or Microsoft Excel), and we do not address here specific techniques of optimization.

5.1 Example

Assume a conflicting set of 5 values. The raw features associated with these values are given in Table 2. Prior to optimization these values are normalized, as shown in Table 3 (the current timestamp is assumed to be 200).

First, assume a utility function with equal weights for all its 6 features. The highest utility (0.501) would be achieved by the fusion

$$x = 0.482 \cdot x_2 + 0.303 \cdot x_3 + 0.215 \cdot x_5$$

Note that the optimal solution suggests ignoring the values x_1 and x_4 altogether.

Next, assume a utility function with equal weights for the four features concerned with "content" (i.e., all but availability and cost). The highest utility (0.666) would be achieved by the fusion

$$x = 0.148 \cdot x_1 + 0.293 \cdot x_2 + 0.337 \cdot x_3 + 0.222 \cdot x_4$$

Finally, assume a utility function whose only nonzero weight is cost=1.0. The highest utility (0.968) would be achieved by the "fusion" $x = x_4$. Note that x_4 is indeed the value with the lowest cost. Tables 4 and 5 summarize these and two other utility functions.

6. RELATED WORK

Previous work related to the subject of this paper can be reviewed in three categories: (1) methods for combining different ranked lists of data objects in a single ranked list (the term *collection fusion* is commonly used here); (2) research that considers features of information sources to determine the value of their data to users (the term *data quality* is often used in these works); and (3) methods for combining different "readings" of the same "phenomenon" in a single value (the true definition of *data fusion*). The following subsections provide brief summaries of the scientific literature in these three categories.

6.1 Collection Fusion

The advent of Internet *meta-search engines*, systems that submit search requests to multiple search engines and then

Feature (raw)	x_1	x_2	x_3	x_4	x_5
Recentness (timestamp)	10	20	30	30	60
Cost (cents)	80	50	30	10	140
Accuracy (standard deviation)	2.5	0.5	2	1	1.5
Availability (probability)	0.6	0.4	0.7	0.9	0.3
Priority (on a scale of 0–5)	4	2	5	1	3
Quality (on a scale of 0–10)	7	6	3	4	5

Table 2: Example: raw features.

Feature (normalized)	x_1	x_2	x_3	x_4	x_5
Recentness	0	0.053	0.105	0.105	0.263
Cost	0.258	0.161	0.097	0.032	0.452
Accuracy	0	0.800	0.200	0.600	0.400
Availability	0.6	0.4	0.7	0.9	0.3
Priority	0.8	0.4	1.0	0.2	0.6
Quality	0.7	0.6	0.3	0.4	0.5

Table 3: Example: normalized features.

feature	u_1	u_2	u_3	u_4	u_5
Recentness	0.167	0.250	0	0	0
Cost	0.167	0	0.333	0.500	1
Accuracy	0.167	0.250	0.333	0	0
Availability	0.167	0	0.333	0	0
Priority	0.167	0.250	0	0.500	0
Quality	0.167	0.250	0	0	0

Table 4: Example: five different utility formulas.

No.	Optimal fusion
u_1	$0.482 \cdot x_2 + 0.303 \cdot x_3 + 0.215 \cdot x_5$
u_2	$0.148 \cdot x_1 + 0.293 \cdot x_2 + 0.337 \cdot x_3 + 0.222 \cdot x_4$
u_3	$0.735 \cdot x_2 + 0.184 \cdot x_4 + 0.082 \cdot x_5$
u_4	x_3
u_5	x_4

Table 5: Example: five corresponding optimal fusions.

combine their results, has sparked interest in the problem of fusing several independent ranked lists of data objects (e.g., documents) in a single ranked list. The relationship to our subject here is rather minor, and we mention it primarily because the term *data fusion* is often used as well. The body of work on this subject is rather large. Surveys and reviews of major collection fusion strategies may be found in [6, 10, 19].

6.2 Data Quality

Works in this category are relevant because they too attempt to quantify the value or fitness of information on the basis of its source and context. Most of these works are presented as *data quality management*, and have often adopted general quality management paradigms [14, 16]. Such quantifications have been especially important in appraising data obtained from the Web, where signal-to-noise ratio in information is often quite low [13], or, with more traditional databases, in applications where the integrity and fitness of the data are considered critical [15]. In assessing the value and fitness of data, many different *dimensions of quality* have been proposed, and although there is no clear agreement on the features that are most important, accuracy, cost, age, availability and reliability have been mentioned most often. A particularly relevant study, which encompasses both data quality and collection fusion, suggests fusing several ranked lists of documents on the basis of various dimensions of quality that are associated with the individual sources [12].

6.3 Data Fusion

Data fusion is an area which has long been almost synonymous with sensor fusion. Sensor fusion is concerned with the interpretation and reconciliation of real-time data received from multiple sensors. The primary applications of such fusion are in military command, control and communications systems [5, 4]. Other applications include navigation and traffic control systems [9], weather and earth sciences [8] and process control in industry [17]. Although such applications are abundant in the literature, to the best of our knowledge, there is no generalized work involving the fusion of individual data items to create new fused values in the area of *information systems*.

Possibly, the closest work can be found in a sequence of studies in probability-based fusion of heterogeneous database attributes [3, 18, 2]. All these works describe systems in which alternative values retrieved from different sources are combined in a *probability function*, which is then offered to the user as "partial values" (i.e., the alternative values are assigned individual probabilities that total 1). The intricacies of assigning these probabilities are different in each study. Whereas [3] assigns all partial values equal probabilities, [18] assumes the probabilities are provided with the data, and [2] tries to approximate these probabilities using Jaccard's similarity coefficients. In yet another study, a winning alternative is chosen using the Dempster-Shafer theory of evidence [7]. Note that none of these approaches create a new value from the values retrieved, and all assume that values are non-numeric.

Using source features for the specific purpose of resolving database inconsistencies in multidatabase environments is the subject of [1]. The utility-based approach described in this paper was sketched briefly in [11].

7. CONCLUSION

7.1 Summary

In this paper we addressed the important issue of *data inconsistencies* in answers constructed from multiple information sources. This issue comes up in all virtual database systems, if they assume (as reality dictates) that their information sources are mutually independent.

To address this issue, we proposed a model that associates metadata (which we called *features*) with each of the information sources. Specifically, we investigated in some depth six features of data sources that may affect the way inconsistencies are resolved: recentness, cost, accuracy, availability, priority, and quality. Of course, there may be other valid features. Yet, it is our claim that it would be possible to incorporate most of these additional features into the framework that was developed here.

These features were combined linearly into a *utility* function, with user-provided weights. This function assigns a utility to each of the values in the inconsistent answer. These values may then be ranked, yielding a *ranked answer*, or the top-ranked answer may be offered as a *preferred answer*.

An underlying assumption of these two types of answers is that the "true" value is one of the values in the set. We then turned our attention to the possibility that the true value is a new value which is a (linear) combination of the given values, called the *fusion*. For each of the six features we showed how to calculate the feature value of the fusion, resulting in an overall utility for the fusion. The fusion should be preferred if its utility exceeds the utility of each of the original elements.

Different fusion formulas would deliver fusions with different utilities. Finally, we considered the problem of the *best fusion*, which is a problem of optimization: find the fusion coefficients that yield the fusion with the best utility.

The number of features is expected to be small (under 10), and when the cardinality of the answer is moderate (say, under 20), finding the best fusion using existing software should not seriously impact the performance of a virtual database system.

7.2 Future Directions

In various places in this paper we assumed that the inconsistent data is numeric. A potentially important extension to this work would be to handle inconsistencies among non-numeric forms of data as well. In this paper we defined the concept of utility and used it for three purposes: (1) to rank of the available alternatives (or conclude a preferred alternative), (2) to determine whether a proposed linear fusion is better than the available alternatives, and (3) to find the best linear fusion coefficients.

Calculating the utilities of alternative values from different sources did not assume that the values are necessarily numeric. Hence, the first result is applicable to the non-numeric case.

The other two results, however, required that the alternative values be fused in a new value, and we assumed that values are numeric and fusions are linear combinations. One possibility for fusing non-numeric values is to divide them into subcomponents, and use recurrence to determine the preferred value of each subcomponent. For example, inconsistent addresses may be fused together by selecting the most "popular" value in each address component (city, zip-code, state, etc.).[7] It should be possible to define the utility of such fusions and compare it to the original utilities, though we expect it to be considerably harder to define fusions that would optimize this utility.

8. REFERENCES

[1] P. Anokhin. *A Comprehensive Approach to the Resolution of Inconsistencies in Multidatabase Environments*. PhD thesis, School of Information Technology and Engineering, George Mason University, Fairfax, VA, 2001.

[2] A.L.P. Chen and C.S. Chang. Determining probabilities for probabilistic partial values. In *Proceedings of the International Conference on Data and Knowledge Systems for Manufacturing and Engineering*, 1994.

[3] L.G. DeMichiel. Resolving database incompatibility: An approach to performing relational operations over mismatched domains. *IEEE Transactions on Knowledge and Data Engineering*, 1(4):485–493, 1989.

[4] T. Kurien, A. Chao, and S.W. Gully. Information fusion for onboard and offboard avionics. In *Proceedings of the SPIE International Conference on Sensor Fusion, Vol. 3376*, 1998.

[5] M.J. Larkin. Fusion of multiple active sonar waveforms. In *Proceedings of the 3rd SPIE International Conference on Sensor Fusion*, 1999.

[6] H.-J. Lenz. Multi-data sources and data fusion. In *Seminar on New Techniques and Technologies for Statistics*. EuroStat, 1998.

[7] E.-P. Lim, J. Srivastava, and S. Shekhar. Resolving attribute incompatibility in database integration: An evidential reasoning approach. In *Proceedings of ICDE-94, 10th International Conference on Data Engineering*, pages 154–163, 1994.

[8] E.W. Measure. A neural network data fusion for retrieval of atmospheric temperature profiles from satellite and surface based radiometry. In *Proceedings of Fusion-98, First International Conference on Multisource-Multisensor Information Fusion*, 1998.

[9] A. Mirabad, N. Mort, and F. Schmid. A fault tolerant train navigation system using multisensor, multifilter integration techniques. In *Proceedings of Fusion-98, First International Conference on Multisource-Multisensor Information Fusion*, 1998.

[10] M. H. Montague. *Metasearch: Data Fusion for Document Retrieval*. PhD thesis, Dartmouth College, Computer Science, Hanover, NH, May 2002. TR2002-424.

[11] A. Motro, P. Anokhin, and J. Berlin. Intelligent methods in virtual databases. In *Proceedings of FQAS-00, Fourth International Conference on Flexible Query Answering Systems*, pages 580–591. Advances in Soft Computing, Physica-Verlag, Heildelberg, Germany, 2000.

[12] F. Naumann. Data fusion and data quality. In *Seminar on New Techniques and Technologies for Statistics*. EuroStat, 1998.

[13] B. Pernici and M. Scannapieco. Data quality in web information systems. In *Proceedings of ER-2002, 21st*

[7] Recall that a disadvantage of ranking-by-recurrence is that all the alternative values must be retrieved apriori.

International Conference on Conceptual Modeling, Tampere, Finland, 2002. Springer-Verlag.

[14] T.C. Redman. *Data Quality for the Information Age*. Artech House, 1997.

[15] A.S. Rosenthal, D.M. Wood, and E.R. Hughes. Methodology for intelligence database data quality. *MITRE Corp.*, 2002.

[16] D. M. Strong, Y. W. Lee, and R. Y. Wang. Data quality in context. *Communications of the ACM*, 40(5):103–110, 1997.

[17] A. Sun. An expert network for factory automation using multisensor information fusion. In *Proceedings of Fusion-98, First International Conference on Multisource-Multisensor Information Fusion*, 1998.

[18] F.S.-C. Tseng, A.L.P. Chen, and W.-P. Yang. Answering heterogeneous database queries with degrees of uncertainty. *Distributed and Parallel Databases*, 1(3):281–302, 1993.

[19] T. Tsikrika and M. Lalmas. Merging techniques for performing data fusion on the web. In *Proceedings of the 10th International Conference on Information and Knowledge Management*, pages 127–134. ACM Press, 2001.