April 1982

Report No. STAN-B-82-947

Modelling Degrees of Item Interest for a General Database Query System

by

Neil C. Rowe

Department of Computer Science

Stanford University
Stanford, CA 94305

Modelling Degrees of Item Interest for a General Database Query System

Neil C. Rowe
Department of Computer Science
Stanford University
Stanford, CA USA 94305

Table of Contents

Abstract

Many databases support decision-making. Often this means choices between alternatives according to partly subjective or conflicting criteria. Database query languages are generally designed for precise, logical specification of the data of interest, and tend to be awkward in the aforementioned circumstances. Information retrieval research suggests several solutions, but there are obstacles to generalizing these ideas to most databases.

To address this problem we propose a methodology for automatically deriving and monitoring "degrees of interest" among alternatives for a user of a database system. This includes (a) a decision theory model of the value of information to the user, and (b) inference mechanisms, based in part on ideas from artificial intelligence, that can tune the model to observed user behavior. This theory has important applications to improving efficiency and cooperativeness of the interface between a decision-maker and a database system.

This work is part of the Knowledge Base Management Systems Project, under contract # N00039-82-G-0250 from the Defense Advanced Research Projects Agency of the United States Department of Defense. The views and conclusions contained in this document are those of the author and should not be interpreted as representative of the official policies of DARPA or the US Government.

Short title: "Degrees of Database Item Interest"

1. Motivation

1.1 Choice-of-alternatives domains

Many databases are used to choose among alternatives. Many of these involve simple entity alternatives with complex tradeoffs [Miller, 1969].

An example we have examined is cargo transport planning for a merchant shipping domain. Here the alternatives are different ways of getting a cargo from one place to another. Different ships may be used, or different routes, or different apportionments of the cargo, to achieve the same end result. Alternatives will differ widely in financial cost, time to execute, riskiness, and degree of resource imbalance. And within each factor several sub-factors will trade off. For instance, ship cargo capacity trades off with cost to operate, and closeness of a ship to a port may trade off with its appropriateness for a given cargo at that port.

Merchant shipping is just one example of distribution network management problems involving choices among item alternatives in a database. Another example is marketing strategy planning of where to most productively commit efforts. Databases that monitor ongoing processes support similar decision-making. For instance, a manager rates production efficiency by how well a schedule is maintained. Here the alternatives are different statistics, and the ratings the degree of deviation from expected values.

Electronic "bulletin boards" exemplify another class of applications. Users will often have widely different interests, so the choice of alternatives is among stored messages. Other examples are a consumer product ratings database and a university classes scheduling database.

Such applications are not "information retrieval" [Lancaster, 1979] in that they use numeric as well as symbolic parameters.

1.2 Why model degrees of interest?

Conventional database query languages (e.g. SEQUEL, QUEL, Query-by-Example [Ullman, 1980]) are not well suited for the above uses. Queries in them must represent a delineation of logical, absolute conditions or operations.

An analogy is putting a large irregularly shaped object into a rectangular box. Even if one has

rectangular boxes of many sizes and shapes, the best-fitting box will leave much wasted space around the edges. In merchant shipping for instance, small size, fast speed, and nearness to the loading port are all desirable in choosing a ship to carry a cargo, but one must ask something like "What ships are between 10,000 and 20,000 tons, over 10 knots in cruising speed, and within 200 nautical miles of Naples?" And that is for only three factors. If there are more, as in most real domains, one must decide which factors to ignore temporarily or queries will be very large. And still one cannot represent the tradeoffs explicitly.

Eventually the user can zero in on a choice under such circumstances. But the process is slower than if one had a smarter interface to the database that would recognize the tradeoffs and act appropriately. How much slower varies with the situation, but time may be expensive for management users of a query system. Much database research has studied improvements in query processing time. The issue here is the less studied but equally important one of session completion time.

1.3 Outline of paper

We first discuss some previous work. Then in part 3 we present our degrees-of-interest modelling method. To clarify how this model works, we give an important application in part 4. Part 5 introduces additional methods for automatically adjusting the model to user behavior. Finally, part 6 gives additional applications of this work.

2. Some previous work

2.1 Extensions to query languages

We can extend a logically descriptive query language by various means; [Yang and Salton, 1978] provides a good summary. We can put "importance" weights on terms, as in:

- Give me 100 ships in the Mediterranean where US registry weights 100, French registry 40, Italian 30, any other 10; and tanker over 70,000 deadweight weights 100, under weights 70, and any other ship type 30.

or:

- Give me 100 freighters over 10,000 tonnage that are near Naples, where 100 more nautical miles from Naples equals 1000 less tonnage and equals 3 knots less speed.

Both are awkward and somewhat ambiguous. They're trying to make mathematical statements that aren't easily expressible in English. Note that *some* mathematics is necessary here, for:

- Give me 100 ships in the Mediterranean, where US registry is preferred to French, both to Italian, and all three to any other; and tankers over 70,00 deadweight preferred to other tankers, then to any other ship type.

doesn't make it clear how much more one thing matters than another. Fuzzy logic [Zadeh, 1979] is one solution, but its applicability is controversial.

This awkwardness has consequences. Note how much easier it is to remove the weighting and ask, "Give me the American tankers in the Mediterranean*', or "Give me ships over 10,000 tons less than 200 n.m. from Naples", even if these queries include too few items or too many unhelpful ones. For the main burden on the user of a query system is formulating queries, not reading answers. So term weighting in query languages is deservedly unpopular when offerred as an option.

Note also some dangers of term weighting. To do it well, one must understand the domain of the database well. Weights can interact unexpectedly, and changing a weight changes its relative importance to all other weights, not just to one. When users make errors in weights it's often hard to locate what is wrong. And users can be wrong about the *form* of a weighting function, confusing additive and multiplicative factors, and weight value adjustments won't work.

2.2 Automatic weighting

Sometimes weights can be assigned in advance for a large class of queries. The classic examples are document retrieval systems that find a set of documents that are the "best match" to user-supplied keywords [Lancaster, 1979]. The usual weighting is the number of common keywords with the document's keywords. Another example is the real-time monitoring of a large virtual database such as a production line or a utility plant, where the relative deviation of readings from normal values is the weighting.

Predefined weightings are by definition inflexible. While they work for a few applications that are sufficiently stable and well-understood, they won't for most database applications. Even in the above examples users need exceptions. Certain document keywords may be more important than others, yet it may be wrong to totally ignore the others; or in process monitoring, a person may want to pay particular attention to a particular measurement today because of recent problems, or know he can ignore a measurement because it is not meaningful today.

2.3 Attempts to compromise

Compromises between user weighting and automatic weighting do not work very well. If you decide to let the user adjust any weights of a system-defined metric, you might as well let him change any weight, for if there were information in the database that would never need to be focused on by weighting, it wouldn't deserve to be in the database in the first place. Thus all the problems cited in 2.1 apply. The size of the changes doesn't really matter; users will still have difficulties seeing how weights interact even if they only have to change one. It's like a naive user allowed to modify a complex computer program.

2.4 Different user views

If a database is characterized by distinct types of usage, then weighting is easier to implement, since each usage can have its own. Historically, user views or subschemas [Wiederhold, 1977] have contained only yes-or-no information, but associating weights with them is straightforward.

But for many databases, particularly large general-purpose types, usages and users cannot be so easily characterized. Needs will differ considerably from session to session with the same user, or even within the same session. And users oftentimes will not know what category of usage they are following. Hence some kind of inference from user behavior is what is really needed, not subschemas alone.

3. Our modelling approach

Our solution to the abovementioned problems has two parts, a flexible representation for degrees of user interest, and an inference mechanism. We now discuss the first; part 5 discusses the latter.

We use a relational database model [Ullman, 1980], and use "item" to mean a tuple or relation row, "field" to mean an attribute or relation column. For simplicity we consider the decision as a choice among items of a single relation.

3.1 A decision theory model

We use decision theory to model the value of real-world options corresponding to database items. Figure 3-l summarizes the approach.

3.1 .1 Utilities and suitabilities

Following classical decision theory [Raiffa, 1970], we interpret choosing between alternatives as meaning choosing the alternative with largest total utility to the user. Utilities can be the sum of several sub-utilities, and sub-utilities are weighted by probabilities of occurrence. Effective utility is the product of a utility and its corresponding probability.

Alternatives in a database can be considered a special case of the general decision theory model, for the probabilities usually can be restricted to prerequisite satisfaction. That is, the probability that a given feature is "suitable" for a user choice; we shall call such probabilities "suitabilities".

As an example, consider a database used for making merchant shipping cargo assignment decisions. Utilities are the financial cost of loading a ship; the fuel, crew wages, and miscellaneous transit costs for a voyage; and the time delay incurred in getting a cargo to its destination. Suitabilities are the ability of a ship to carry a particular kind of cargo; and the ability of a ship (due to its dimensions) to be serviced at a particular port.

Or consider a university classes database which students use to decide what courses to take next term. Utilities are the work load for a course and the number of trips to campus required. Suitabilities are relevance of a course to a student's interests, the degree to which he satisfies the stated prerequisites for it, and tolerability of the instructor.

Field values retrieved from a database
(e.g. latitude, longitude, length, draft, fuel capacity)

Estimated field values
(e.g. weight capacity, speed)

Slot values
(e.g. source port, cargo size)

Utility and suitability terms
(e.g. voyage costs, port cost:, time cost, port suitability, cargo suitability)

Utility weights
(e.g. importance of voyage cost vs. time cost)

Total utility and total suitability

Item choice probabilities (p_i)
(e.g. how likely a user is to choose this ship)

Set preferences
(e.g. whether the user prefers this set of ships to that)

Figure 3-1: The cost-function degrees-of-interest model (with merchant shipping examples)

3.1.2 On choosing utilities and suitabilities for a domain

Generally speaking, utilities and suitabilities should be independent "conceptual chunks". To do this, each utility or suitability should aggregate logically related subfactors. For instance, financial cost for sending a ship somewhere and time "cost" of the time it takes the ship to get there are quite different things. While we want both of them small, it's not obvious how much to weight each. On the other hand, crew wages, fuel cost, and loading costs in port can all be measured in a single unit, money. So financial cost and time cost should be considered two separate utilities. This kind of distinction will be important in part 5.

In many cases a real-world phenomenon can be modelled by either a utility or a suitability, and it is very important to decide which is more appropriate. For instance, suppose a shipper is trying to choose a ship to carry 10,000 tons of cargo. Ships with cargo capacities smaller than 10,000 are undesirable since more than one would be needed, but ships with large cargo capacities are also undesirable since they cost more to operate than smaller ships. We could model this by a cost curve like a parabola with a cost minimum somewhere around 11,000 tons. However, the small values are undesirable in the sense of adequacy; the large values are undesirable for high cost. Thus the former should be a suitability, the latter a utility. Simple term weighting in query languages cannot recognize this distinction, and thus is fundamentally inadequate.

3.1.3 Exploiting correlations

Utilities and suitabilities can be computed in various ways from a database. Fuel cost in a shipping database can either come from a dedicated field or estimated from the tonnage, length, beam, or draft of the ship, to which it is usually closely correlated. One can analyze the database in advance, looking for strong such correlations between numeric attribute values, and compute regressions, linear or nonlinear as appropriate. The error in a regression can be characterized by bounds and dispersion, and information from several regression estimates can be combined to get a tighter estimate, using classical statistical methods.

Such regressions are very helpful for degrees-of-interest analysis of a database because they can reduce the amount of information about an item needed to evaluate it. In the context of a database query system one can exploit the information that had to be retrieved anyway to answer the user's query.

3.1.4 Combining utilities and suitabilities among themselves

We follow [Raiffa, 1970] in arguing that multi-attribute decision situations can frequently be reduced to a single utility function This done, sub-utilities may be combined by weighted summation. The weights can be user-specific, and can have uncertainty to them just like term estimates from regressions. We can take convolutions or assume normal distributions when combining several uncertain quantities.

Suitabilities can be combined multiplicatively when independent. (Hence it's important to aggregate nonindependent factors into a single value, following 3.1.2.) Weights are not necessary -- they are implicitly represented in the suitabilities themselves. Suitabilities are more convenient to refer to by their logarithms, so they can then be combined additively like utilities.

3.1 .5 Combining utilities with suitabilities

So for each alternative item in the database (usually meaning each entity as per the entity-relationship model [Ullman, 1980]), we can get a cumulative utility and suitability with respect to some user/usage. Now we must combine the utility and suitability. It is not possible to simply multiply them for each entity, for the appropriateness of one item depends on the relative inappropriateness of all other items. Various approaches have been proposed for this kind of problem (for example [Luce, 1959]). Our approach is to compute mutually exclusive selection probabilities for each item by:

$$p_i = s_i \cap \bigcap_{\substack{j=1 \\ j \neq i}}^{n} (1 - s_j R(j,i)) \qquad (3.1)$$

for each item i. Here s_i is the total suitability of item i, and R is the user indifference function on utilities that gives the probability the user prefers item j to item i. We explain this formula as "Item i will be chosen if (a) it is suitable, and (b) for every other item j, it beats that item j (i.e., either j is unsuitable or the user prefers i to j)". This is a model of a careful choice between alternatives, assuming plenty of time for the decision, and is not necessarily how people make choices.

For this work we are using the specific user indifference function:

$$R(j,i) = \Phi((u_j - u_i)/\sigma_{indiff}) \qquad (3.2)$$

where u_i is the total utility of the ith item, Φ the integral of the unit normal curve, and σ_{indiff} user tolerance to differences in utility; and we assume the larger a utility, the more desirable. This formula follows from our statistical assumptions and the additional reasonable assumption that the user always likes gaining the same amount of utility equally no matter what the circumstances. (Transformations on the utility values can be appropriate, like logarithms if the user attends to the ratios of costs instead of their differences.)

If total utility and total suitability are statistically independent of one another, we can use a computationally simpler variant:

$$p_i = s_i \prod_{\substack{j=1 \\ j \neq i}}^{n} (1 - s_j R(j,i)) \qquad (3.3)$$

or:

$$p_i = s_i / (1 - .5 s_i) \prod_{j=1}^{n} (1 - s_j R(j,i)) \qquad (3.4)$$

where Π indicates a product. Regarding statistical independence we argue that **(1)** complex domains with many competing factors will tend to show it, (2) nonindependence can often be removed by coalescing terms, and (3) even with nonindependence the above equation is a useful estimate, especially since the matching of every pair of items will tend to average out nonindependence effects.

These selection probabilities p_i, then, are quantitative "degrees of interest" in database items. They represent mutually exclusive probabilities the user will choose an item, if he had to choose only one.

3.1.6 Combining uncertain utilities with uncertain suitabilities

If total utility and total suitability values are characterized by a probability distribution, we must modify the above approach slightly. Since generally these two totals represent the cumulative effects of many random factors, their uncertainty will often be approximable by normal distributions. Thus we can speak of them as having means μ_u and μ_s, and standard deviations σ_u and σ_s.

The σ_{indiff} in our user indifference function (3.2) is a kind of perceptual vagueness of the user for utilities that are close together. A ship that costs $11,000 to send someplace is theoretically better than one that costs $11,500, but it might not be worth bothering about. A σ_u of the total utility is analogous to this user utility indifference. So we can revise our user indifference function used in **(3.1)** and (3.3) to:

$$R(j,i) = \Phi[(u_j - u_i)/(\sigma^2_{indiff} \quad \sigma^2_u)^{1/2}] \qquad (3.5)$$

The suitability standard deviation σ_s is less important. If we use formula (3.4, it controls the uncertainty in the selection probabilities p_i. But usually we're not interested in this, only the mean of the p_i distributions; and a good estimate, especially for small σ_s, can usually be found by knowing only the means of the s_i distributions.

3.1.7 Some caveats

Our approach is pragmatic, and does not claim to advance decision theory. Some design choices we have made are relatively arbitrary. Many refinements of this approach are possible based on the extensive literature of decision theory applications.

3.2 Slot filling

Our framework for modelling degrees of interest is quite general. In addition, a further degree of customization for users and usages is possible based on some specific inferences of what the user is trying to do.

For instance, properly speaking there is no one "cargo transport planning" task for a merchant shipping database. There are plans that involve cargos being sent from Naples to Marseilles, Barcelona to Genoa, and so on; and for each pair of ports, there are different kinds of cargo to send, and different tonnages, which will necessitate quite different kinds of arrangements. Somehow we must infer the particulars of what a user is doing today. We call this "slot filling".

3.2.1 Using frames

What we want to do is similar to the instantiation of "frames" or "procedural nets" in such work on planning as [Bobrow et al, 1977] and [Sproull, 1977] (the latter utilizing decision theory extensively as well). We can associate a set of variables with a task. For instance, cargo transport planning has a Source Port, a Destination Port, a Cargo Type, and a Cargo Amount. All these may not have determined values -- for instance, a user may not be sure which port is most convenient to get oil from -- but these are general "slots" we will try to fill when we can.

3.2.2 Kinds of slots

The major kinds of slots are:

- *Domain of discourse.* E.g., the user asks about available American tankers. This suggests interests in (a) tankers, (b) American ships, and (c) ships with "available" status. We can then set the suitabilities for those categories as high, and for all other sibling categories low.

- *Reference standard.* E. g., the user asks for ships within 100 nautical miles of Naples. This suggests that Naples is a "reference standard" for ship location, and hence that some action is being planned for Naples that will mean ships not there will have to travel there. Naples then is the zero point in calculating voyage distances.

- *Threshold* values. E. g., the user asks for ships with more than 10,000 tons capacity. This strongly suggests he is considering transporting about 10,000 tons of cargo, for at that value the restriction makes the most sense.

- *Answer* set size. E. g., the user asks for twenty ships satisfying the restrictions. This says something about the relative importance of "recall" (total value of information retrieved) vs. "precision" (density of information value retrieved), following [Lancaster, 1979].

3.2.3 Rules for slot filling and their order

We can write domain-specific rules to fill slots, using a pattern matcher which looks for certain categories of terms in the query-language expression of a query, and then sets certain global variables.

Rules can invoke other rules, as in other such "production systems" [Davis and King, 1976]. For instance, a restriction "tonnage greater than 10,000" suggests that the user has about 10,000 tons to transport (though see the discussion in next section), but also that he's not too fussy about the exact amount since he gave such a round number. So we could invoke a "fussiness" rule every time we fill a threshold-criterion slot, a rule that sets an associated standard deviation.

The order of rule application matters, as in most production systems. For instance, a port mentioned in a query can be either a Source Port or a Destination Port. Generally though the user querying mirrors in time the events that are being planned, and the Source Port usually occurs first. We can associate this tendency with a probability, and use a "certainty factor" scheme in the manner of systems such as PROSPECTOR [Duda et al, 1977].

3.2.4 Value uncertainty in rules

Besides uncertainty as to which rule to apply, there can be uncertainty about exact slot values. There are two cases, numeric and nonnumeric slots. An example of the former is Cargo Amount inference from the query restriction "tonnage must be more than 10,000 tons". We can't be sure the user has exactly 10,000 tons to transport -- he may be a "conservative" and have 9,000 tons and want to play it safe, or he may be a "liberal" and have 11,000 tons. Under similar circumstances, the same user often shows the same ratio of stated value to actual criterion. Hence we can associate with a user and slot a particular ratio mean and standard deviation, to characterize his behavior, and extend the ideas of section 3.1.6. (The mean will generally be very close to 1.)

For a nonnumeric slot example, suppose the user asks for tankers of a certain size. The mention of tankers suggests a high suitability for the tanker ship type, but not necessarily a zero suitability for

other types. We can characterize this tendency as a mean ratio of the suitability of such values to the suitability of explicitly mentioned ones, with a standard deviation.

3.3 Task apportionment

A general database system will have many different kinds of usage. Even with the flexibility we have introduced above with different utility weights, different slot values, etc. it is impossible to fit them all into the same cost-function (utility and suitability) structure. Cargo transport planning is really fundamentally different from planning an at-sea rescue operation, or deciding which ships to assign to which docks in a port, both of which could use the same merchant shipping database; the important utilities and suitabilities are quite different. We thus partition the database domain into "tasks", each with a separate cost function. We assume the user follows only one task at a time, a reasonable assumption for nearly all usage.

We can ask that the user state his task before starting to query the database, or define it beforehand in a subschema. Or we can use Bayesian inference to guess his task dynamically: by assigning probabilities to a fields's being mentioned in the course of a task, we can reason backward to the probability of a given task from the fields mentioned.

3.4 User hierarchies

We have mentioned a number of parameters that will vary from user to user -- tasks used, utility weights, suitabilities, slot-filling uncertainty. We can examine parameter values by user classes and over time. We can average for a class of users, say, and use this average as a default for new users that are known to belong to the class. User group or time period averages may be grouped hierarchically.

3.5 A few comments on implementation

Utilities and suitabilities can be thought of as derived data [Wiederhold, 1977] which will be recomputed by operations on actual field values as needed. (Though it may be efficient to precompute them for database entities that do not change much with time.) Specifications of the operations for deriving utilities and suitabilities belong in the conceptual schema with other routine-access information, and should be determined during database design. Calculation constants (e.g. fuel cost per gallon, the average time to dock in a port) can be stored separately to allow for easy examination and update. Additional storage requirements for degrees-of-interest analysis are not large.

Processing time using equation (3.3) should not be significant either, considering that the extra processing requires few additional page accesses, since it involves fetching information that we claim the user will need to retrieve explicitly anyway in order to make an intelligent choice. The Φ function can piecewise approximated, and all else that is required is multiplication. Thus even though the algorithm is $O(N^2)$ in the number of items for comparision N (though O(N) in the calculation of total utility and total suitability), this should be negligible compared to secondary storage access times which are unavoidable anyway.

4. An example application: very large query answers

We will discuss applications of this work in chapter 6. Now, however, we will try to clarify the ideas of the preceding section by a particular application, the handling of very large answers to queries on a database. This application was implemented for a merchant shipping database.

Some choice-of-alternatives situations have many factors to consider, implying a large number of superficially reasonable choices. The user may have to look through a lot of data in order to make a good decision. Database "end-user facilities" [Spath and Schneider, 1977] have not much addressed this circumstance because they have lacked tools. We suggest that a degrees-of-interest calculation can be used to sort the items and display them by decreasing estimated value, helping the user by increasing the probability that he will find what he needs early in the list. This can reduce the query "session time" discussed in 1.2. Again, in doing this we must make the classical decision-theory assumption that degrees of interest are unidimensional, although the last example here shows a way around this restriction.

4.1 Ordering rows and columns

Previous work has supplied many ideas for output presentation in decision-making contexts (cf. [Newsted and Wynne, 1976], [Press, 1971], [Rouse, 1975], [Spath and Schneider, 1977]). We decided to use a simple tabular format, consistent with our relational database model. So for most cases with our merchant shipping database, rows correspond to ships and columns correspond to attributes of ships. There are two issues: how to order the rows, and how to order the columns.

With rows, order the tuples by decreasing p_i selection probabilities (or the means of their distributions), and show them to the user in this order. If he's using a display device, fill the screen and ask if he'd like to see more, then get the next batch if he says yes, and so on. Give him commands to go back and look at any screenful again.

With columns, do something analogous: try to put the "most interesting" columns to the left. Estimate this from the sensitivity of the p_i selection probabilities on the average, to the column's information type. Make one important exception, however, for column ordering: the ruling parts or keys of the table being displayed should come before the things which they "rule", a standard tabular convention.

4.2 Output for the cargo transport task

We first show our program for an assumed cargo transport task, the task we have studied the most. The task is for moving a cargo by ship from one port to another. We do not claim a highly accurate cost model for this -- that would require long discussions with merchant shipping experts -- but we think we have included most of the appropriate factors appropriately.

Factors incorporated in utilities include loading and unloading costs, the value of the shipment, the insurance cost, crew wages, fuel costs, and delay in completing the shipment. Factors incorporated in suitabilities include whether the draft and length of the ship permit it to visit the harbors mentioned, how easily the ship can accommodate the specified amount of cargo, and whether the ship is in port (and hence more likely to be free). When these factors are not mentioned in the explicit query, or in a previous query involving these ships, approximations are inferred using correlations (after 3.1.3).

As for columns, -distance is found to be significantly more important to overall cost than length, draft, and other dimensions of ships. Length comes before draft because it came before it in the query, a useful default rule.

See Figure 4-1.

Note that ships are not listed in order of distance from Naples, length, or draft, but a complicated function combining the effects of all three (and weight capacity, a field not specifically requested but asked about in the query). The program infers Naples as the source part from this query; if another port were mentioned later, it would be identified as the destination port.

4.3 Output for the search and rescue task

Now we present an example of output appropriate to a search-and-rescue task. The time for a ship to reach the rescue site is critical, and thus is the major utility factor. Minor factors are the number of lifeboats on the rescue ship, the availability of a doctor, and the "undocking" time for ships in ports. Suitability depends only whether the ship is in port or not -- if it is, it may be less available due to repairs, red tape, etc.

Figure 4-2 shows output for a sample query. 40N11 E is inferred the trouble spot. Maximum speed is very important, so its column comes first. Distance is more immediately relevant to the problem than position and lifeboats, so it comes next. Latitude and longitude are given in a standard order since they go together conceptually. Last comes lifeboat capacity, following the order in the query.

Give length, maximum draft, and distance from Naples for all Italian ships greater than 25000 tons weight capacity and less than 800 nautical miles from Naples.

```
(THIS IS SCREEN # 1 OF 4)
     NAME        DIST LENGTH DRAFT    UTIL      SUIT
   ROSSINI   353   515     20       21714.55 .2767195
ACHILLE LAURO 424  "       "        23124.2     "
    APPIA     612    "       "       26856.81    "
  NAI MARCUS  92  5 2 3     30       17849.82 .2073752
LUIGI GRIMALDI 465 455     29       24367.21 .2314018
   POLINNIA   513   "       "       25268.84    "
   FEZZANO    645   "       "       27748.34    "
CIELO DI ROMA 513 523      30       25593.77 .2073752
  CORALLINA 577   536      36       29116.48 .2020731
   GALASSIA  743  418      32       28290.11 .1708212
(COMMAND (N= NEXT SCREEN, P= PREVIOUS, <NUMBER>=GO TO THAT SCREEN,
        Q=QUIT) :)N
(THIS IS SCREEN # 2 OF 4)
     NAME         DIST LENGTH DRAFT    UTIL       SUIT
  GATTOPARDO 457  536      36       26954.0   .1010365
  ORSA MINORE  513  "       "        27963.16    "
  CATERINA  M  46  559     38       20246.81 .09080319
 LUICI  GALVANI  46  "       "        "          "
  LAMINATORE  743  536     36       32107.91 .1010365
CORONA BOREALE 513 574     "        29072.2  .09813949
MARE TRANQUILLO 757 "      "        33484.64    "
    ACRUX    414    559     38       26796.52 .04540159
  AGIP  GENOVA  572  "       "        29608.62    "
CHEMICAL ORRIOS  0  587     46       23871.37 .003443577
(COMMAND (N= NEXT SCREEN,  P= PREVIOUS, <NUMBER>=GO TO THAT SCREEN,
        Q=QUIT) :)N
(THIS IS SCREEN # 3 OF 4)
      NAME          DIST  LENGTH  DRAFT    UTIL       SUIT
    SU   NURAXI    513    587     46       32891.05 .001721788
   BORDIGHERA   218   710     52       37142.43 .0006767076
    LIQUIMARE    513   "       "        42511.28     "
  AGIP VENEZIA     572   "       "        43585.05     "
MARINELLA D'AMICO  0  775     65       46984.98 .0004848006
  AGIP   SARDEGNA  46  "       "        47858.95     "
CIELO  DI  SALERNO 302 "       "        52722.75 .0002424003
. AGIP  ANCONA  5 1 3          "        56731.59     "
    AGIP   BAR1   .   " "      "          "          "
    PERTUSOLA       "    "      "          "          "
(COMMAND (N= NEXT SCREEN, P= PREVIOUS, <NUMBER>=GO TO THAT SCREEN,
        Q=QUIT) :)Q
```

Figure 4-1: Sample output for the cargo transport task

4.4 Output for the ship survey task

Finally, we show output for the ship survey task, wherein the user examines characteristics of the current merchant fleet for long-range planning purposes, not necessarily in order to send them anywhere. See Figure 4-3. Type is the most important information (since it so strongly affects how a ship can be used), followed by weight capacity. Fuel capacity and fuel type are ordered as per the query. Cost-function ordering here is equivalent to a sort first on type, then weight capacity, then fuel capacity, and then fuel type, so we haven't bothered to print utilitities and suitabilities.

This example illustrates a second, independent kind of "interestingness" we can simultaneously note for the user: we flag by asterisks particular values that are in some sense unusual. For numeric fields this means extremely small or large values (where the gap to non-extreme values is sufficiently large); for symbolic fields, values that occur very infrequently (where the gap to other frequencies is sufficient). This "local" kind of "interestingness" complements nicely the "global" kind presented in part 3, and is easy to compute.

Give, for all ships within 155 nautical miles of 40N11 E, their position, maximum speed, and lifeboat capacity.

```
(THIS IS SCREEN # 1 OF 1)
      NAME          MAXSPEED DIST LAT N/S LONG E/W LBOATCAP     UTIL    SUIT
  LENINGRADETS         15     39   3948  N   1148  E      52     3457.286  1.0
  GERASIMOS K          21     91   4127  "   1137  "      "      6069.355   "
   PATRICIA S          "      94   3927  "   1255  "      "      6268.81    "
BRITISH TENACITY       19     87   4048  "   924   "      25     6344.59    "
NIKOLAY NEKRASOV       21    126   3754  "   1112  "      52     8396.326   "
    LA BAHIA           12    150   3903  "   800   "      25     15779.76   "
    DEFIANCE           21    114   4049  "   1315  "      52     12238.51   .5
 EXPORT FREEDOM        "     114    "     "    "    "      "        "        "
 PRESIDENT POLK        "      1     14    "    "    "      "        "        "
     KRYMSK            "     115   4051  "    "    "      "      12304.99   "
(COMMAND (N= NEXT SCREEN, P= PREVIOUS, <NUMBER>=GO TO THAT SCREEN,
        Q=QUIT) :)N
(THIS IS SCREEN # 2 OF 2)
      NAME          MAXSPEED DIST LAT N/S LONG E/W LBOATCAP     UTIL    SUIT
      KUDU             21    115  4051  N   1315  E      52     12304.99  .5
CHIEF COLOCOTRONIS     19    115   "    "    "    "      30     13006.99   "
      GORI             "     115   "    "    "    "      "        "        "
    CATERINA M         17    115   "    "    "    "      61     13807.57   "
  LUIGI GALVANI        "     115   "    "    "    "      "        "        "
   WORLD COMET         "      1    15    "    "    "    "        "        "
  LISSY SCHULTE        14    115   "    "    "    "      60     15368.19   "
  AGIP SARDEGNA        12    115   "    "    "    "      25     16747.15   "
(COMMAND (N= NEXT SCREEN, P= PREVIOUS, <NUMBER>=GO TO THAT SCREEN,
        Q=QUIT) :) Q
```

Figure 4-2: . Sample output for the search-and-rescue task

Give ship weight capacity, type, fuel capacity, and fuel type for all ships within 150 nautical miles of Naples.

```
(THIS IS SCREEN # 1 OF 4)
        NAME          TYPE  WTCAP  FUELCAP  FUELTYPE
  AGIP SARDEGNA       TKR  288346    266      DIES
MARINELLA D'AMICO      "      "       "        "
  LISSY SCHULTE        "    182115    180    *H DIE*
 CHEMICAL ORRIOS       "    110549    113      COAL
CHIEF COLOCOTRONIS     "      "       "        "
       GORI            "      "       "        "
    CATERINA M         "    76370     73      B OIL
 CHEMICAL ENERGY       "      "       "        "
  LUIGI GALVANI        "      "       "        "
      MANDAN           "      "       "        "
(COMMAND (N= NEXT SCREEN, P= PREVIOUS, <NUMBER>=GO TO THAT SCREEN,
        Q=QUIT) :)N
 (THIS IS SCREEN # 2 OF 4)
        NAME          TYPE  WTCAP  FUELCAP  FUELTYPE
    WORLD COMET        TKR   76370    73      B OIL
  BRITISH POPLAR       "     51212    53       DIES
    NAI MARCUS         "       "      "        "
      VELENJE        *BLK* 54658     50      *JP-5*
     ALIKRATOR         CGO   19494    20       KERO
      ANEMO K          "       "      "        "
  ARKADIY GAYDAR       "       "      "        "
CORRIERE DELL'OVEST    "       "      "        "
     DEFIANCE          "       "      "        "
  EXPORT FREEDOM       "       "      "        "
(COMMAND (N= NEXT SCREEN, P= PREVIOUS, <NUMBER>=GO TO THAT SCREEN,
        Q=QUIT) :)N
 (THIS IS SCREEN # 3 OF 4)
     NAME          TYPE  WTCAP  FUELCAP  FUELTYPE
 GERASIMOS K       CGO  19494    20       KERO
 GUNHILD TORM      "      "       "        "
  JUMBOEMME        "      "       "        "
    KRYMSK         "      "       "        "
     KUDU          "      "       "        "
MAESHIMA MARU      "      "       "        "
  PATRICIA S       "      "       "        "
    PAYDE          "      "       "        "
PRESIDENT POLK     "      "       "        "
 ROBERTOEMME       "      "       "        "
(COMMAND (N= NEXT SCREEN, P= PREVIOUS, <NUMBER>=GO TO THAT SCREEN,
        Q=QUIT) :)N
 (THIS IS SCREEN # 4 OF 4)
      NAME          TYPE  WTCAP  FUELCAP  FUELTYPE
  SHIRLEY LYKES      CGO  19494    20       KERO
DNEPROVSKIY LIMAN  *REF* 31790     35       COAL
  LENINGRADETS     *TUG* *7915*   *10*     B OIL
(COMMAND (N= NEXT SCREEN, P= PREVIOUS, <NUMBER>=GO TO THAT SCREEN,
        Q=QUIT) :)Q
```

Figure 4-3: Sample output for the ship survey task

5. Inference of cost function parameters

Slot-filling and partitioning usage into tasks with distinct cost analyses are steps towards accommodating a broad class of users and usage, but for a large general-purpose database additional flexibility is needed. This can be provided by dynamic inference of analysis parameters from user behavior.

5.1 Why inference is needed

Widely varying situations are subsumed under the same task. For instance in regard to time criticality, cargo transport covers both emergency relief supplies and scheduled monthly shipments. In a database where both such extremes occur, what weight the user will be using today on time cost is unknown. Using the average weight for previous sessions of this user or his user class may be fruitless.

Note it is difficult for users to explicitly say what parameter values they are using today. Parameters may be far removed from the way they think of a problem. And as we remarked in 2.1, weights interact in complicated ways by themselves; it is still more complicated with suitabilities and slots to consider. Cost-function analysis is for computers, not people.

5.2 Some simplifying assumptions

Inference in general is computationally expensive. To use it for an often busy database we make some simplifications:

1. We use average values for users and user groups as our starting defaults for parameters; this can help convergence on an interpretation in analysis.

2. We assume the form of the cost function is fixed, i.e., what the utility and suitability terms are for a task, and how they go together. This is usually safe.

3. We reserve the right to do this analysis post hoc if it gets too complicated. We can allocate it a fixed amount of resources while the user is logged in, dump what is left unaccomplished, and return to it when system usage is light. Such retrospective analysis of the query session is still useful since it provides defaults for the next time the user uses the database.

5.3 "Parsing" query sequences

To infer user parameters we need a theory of what the user is trying to do. Following work such as [Hobbs 1978], [Cohen, 1981], and [Cohen et al, 1981], we want to "parse" the query sequence, explaining query 4 as an elaboration of query 3, say, or query 8 as a "retreat" from a query 7 with an unhelpful answer.

5.3.1 Query ordering

Unfortunately, query sequences to a database are not well described by artificial intelligence planning models such as [Bobrow et al, 1977], [Sproull, 1977], and [Appelt, 1980]. People can acquire information in almost any order and still make sense of it. However, they do have certain predispositions in orders. They ask about the most important things first, or they ask about related things together, or they follow sequences in time or space. In database querying for decision making the first kind is primary, and our cost function model is just what is needed to address it.

5.3.2 Query answer set preferences

A key notion is the user's "preference" of the answers to one query to the answers to another. This will be a weighted sum of both "recall" (the total information in a set) and "precision" (its information density), after [Lancaster, 1979].

We give some heuristics for determining preferences, in approximate order of decreasing certainty factor. A user prefers answer set A to answer set B:

1. If a database is used for actually effecting real-world actions -- as when you can actually order a ship somewhere by updates to a file -- and A includes action choices when B doesn't

2. Almost as good, if one can retrospectively learn real-world choices -- as when later information reports that a ship was ordered somewhere, and when -- and A items were included when B items weren't

3. If you ask the user and he says yes. But users may not act consistently, or questioning may be an imposition.

4. If the user knows real-world referent connections (e.g. tuple names) for A items, but not B items. Rationale: the database is only a model of the real world, and explicit connections to reality are usually needed.

5. If A items are inter-distinguishable by properties, but B items are not (i.e. there is some basis on which to choose an A item, but not a B item). Rationale: people don't usually choose randomly.

6. If the user knows necessary real-world implementational data (e.g. call sign for sending orders to a ship at sea) for A items, but not B items. Rationale: same as .

7. If A is a subset of B, or A = B, and A follows B in the query sequence. This is the most useful condition. Its strength increases with every additional subset of A the user creates. Rationale: people continue searching in a place when they think they're on to something.

8. If A is last or "late" in a query sequence. Rationale: people stop searching when they find what they're looking for.

9. For an English-like query language environment, if the user gives more positive verbal cues for A than B, e.g. the difference between "Great. Now tell me . ..." and "Hmm. Now tell me . ..."

10. If A is a small but nonempty set. Rationale: people narrow choices when only a few choices are satisfactory to them.

Preferences are also transitive, with certainty factors combining to give a higher factor for the result. If we like, we can set a threshold criterion as to which and how many of these criteria must be satisfied in order to definitety conclude a preference exists.

5.3.3 An example parse

'Figure 5- 1 gives an example query sequence. Here the sets of ships in queries 3, 5, 7, and 9 are preferred to the set in query 1 by the fourth and tenth heuristics. Sets 5, 7, and 9 are preferred to the set in 3 by the fifth heuristic, and also by the seventh, but note that 7 is preferred to 5 by virtue of 9 being a subset of it. Finally, 9 is preferred to all the others by the sixth, seventh, and eighth heuristics.

5.4 Relevance feedback with the cost function

-Preferences are important because they can extend the idea of "relevance feedback" from information retrieval [Yu et al, 1976] to more general databases. That is, we can refine the parameters of a degrees-of-interest model based on which choices seemed more "relevant" for the user.

5.4.1 Inverting the formula

The main problems in parameter inference are finding weights on utilities and finding suitability values; other parameters follow from this. To do this, we must "invert" the formula (3.3); that is, treat the constants as variables and vice versa.

This is hard to do with formula (3.3). However, we can get an upper bound by (a) setting the inner suitability multipliers to 1, and (b) replacing the value of the product by its smallest term:

1. **User:** How many tankers are in the Mediterranean?

2. Database: 37.

3. **User:** List the American ones.

4. Database: Titanic, Bounty, Pequod, Lusitania, Pueblo, Mayaguez.

5. **User:** Give the tonnages for those more than 500 feet.

6. Database: None are that long.

7. **User:** Give the tonnages and positions for those over 300 feet.

8. Data base:

SHIP	TONNAGE	POSITION
Bounty	14000	**40N13E**
Pequod	8000	**45N5E**
Pueblo	17000	**43N18E**

9. **User:** Good. What are the captain and radio call sign of the Pequod?

10. **Database:** Ahab and WHL.

Figure 5-1: Query sequence illustrating set preferences

$$p_i = s_i \, \Phi((u_i - u_{best})/\sigma_{indiff}) \quad (5.1)$$

But this is still difficult to invert, since it leads to terms of products of pairs of unknowns. Instead we can eliminate the Φ warping for a reasonable estimate:

$$v_i = s_i (u_{worst} - u_i) / \sigma_{indiff} \quad (5.2)$$

$$p_i = \prod_{j=1}^{n} \Phi((v_i - v_j)/\sigma_u) \quad (5.3)$$

Note (5.2) can also be derived a different way, and thus may have some generality. Let the "usual" utility of a choice item be u_m. Then for any item i an "effective utility" akin to a v_i is equal to

$$s_i u_i + (1 - s_i) u_m = s_i (u_i - u_m) + u_m \quad (5.4)$$

which amounts to the same thing as (5.2) where $u_i - u_m$ here equals u_i there.

It is true that equation (5.2) represents a significant distortion of (3.3). But it can be almost as good for inference purposes, since the success of inference is not well-defined: a user need not be terribly consistent in his parameters. We believe often users are consistent enough that useful order-of-magnitude estimates of parameters are possible. Also, with data averaged for user and usage classes, much of the time we expect to be making small adjustments to what were initially rather close parameter values, and a crude adjustment scheme is sufficient.

5.4.2 Decoupling utilities and suitabilities

Total utility is the sum of utilities, and total suitability is the product of suitabilities, under the independence assumptions of formula (3.3). We can decompose the analysis to an examination of each separately by the following strategy based on equation *(5.2):*

1. Compute utility weights, assuming an estimate of total suitability for each item. Use previous weight estimates, if any, for a starting point.

2. Compute the total utilities for each item based on these computed weights.

3. Compute the *logarithms* of suitabilities (since they are a product), assuming an estimate of the logarithm of total utility for each item. Use previous suitability estimates, if any, for a starting point.

4. Compute the total suitability for each item based on these computed logarithms of each suitability term.

5. Go to step 1 unless values have changed less than some criterion from **last iteration.**

There may be many suitability unknowns in general, but usually many of these are correlated with

others and can be eliminated. For instance, after 3.2.4, the suitability of a ship for a particular amount of cargo is a sigmoidal (integral of normal curve) function with a mean and standard deviation; we thus have only two unknowns to find.

5.4.3 Generating inequalities

We have now reduced the problem to one of finding a set of unknowns in a weighted (with utilities) or unweighted (with suitability logarithms) sum. We now bring in the notion of set preference defined in 5.3.2. There are various ways to interpret it mathematically. We choose the simple criterion that the user significantly prefers set A to set B if any member of A is "significantly" better in p_i than any member of B. Since Φ in equation (5.3), the integral of the normal curve, is monotonically increasing, this condition is equivalent to saying that any member of A is significantly better in v_i as defined by equation (5.2).

Formally, we are given two known vectors with unknown weights on their components, plus the knowledge that one's weighted sum is larger than the others's. Then the weighted sum of the vector difference of the first vector minus the second must be greater than zero. We can generate such linear inequalities for every pair of items in the preference sets.

There are other sources of linear inequalities. Set preferences may be explainable directly from the queries themselves without examining the query answers, if those queries are sufficiently similar. Suppose we can define a T_A as the set of query restrictions in A that do not logically follow from those of B, and a T_B as vice versa. Then:

- If T_A and T_B represent single terms that affect the same single suitability or utility, $s_A > s_B$ or $u_A > u_B$. For instance, if T_A is "American ships" and T_B "Russian ships", that says the nationality sub-suitability of American ships is larger than that for Russians.

- If T_B is null and T_A affects a single suitability or utility, s_B or u_B is greater than the suitability or utility of any other possibility. For instance, query 3 vs. query 1 of Figure 5-1: if 3 is preferred to 1 then American ship sub-suitability is larger than the nationality sub-suitability for any other nationality.

- If T_A is null and T_B affects **a** single suitability or utility, s_B or u_B is less than the suitability or utility of any other possibility.

And more complicated rules can be written for more complicated T_A and T_B expressions, although the certainty of the generated inequality decreases quickly with complexity, and the tuple comparision method just mentioned soon becomes preferable.

Inequalities also come from knowledge of the particular database application. For instance, an

empty ship may be preferable for transport to one with cargo, a user may prefer using his corporation's own ships when possible, and the riskiness utility may be never more than one fifth of the financial cost for a given shipping situation. Inequalities can also state reasonable bounds on the search space for parameters.

5.4.4 Pruning inequalities

The above generates a lot of linear inequalities; note there are mn comparisions of two sets sizes m and n. Many of these inequalities will be redundant, and must be pruned for efficient analysis of the rest.

Write all inequalities as a weighted linear sum of terms plus a constant being greater than zero. Using negative logarithm transformations on suitabilities, and defining utilities so they are always nonnegative, we can consider all variables nonnegative.

We can eliminate inequalities with all zero coefficients, plus those with all positive coefficients and a positive constant, since they will always be true, and those with all negative coefficients and a negative constant, because they're contradictory and can never be satisfied (we assume the user accepts a few contradictions in order to ask simple queries). We can also eliminate inequalities derivable from others. Considering the coefficients and constant of an inequality as a vector, inequality V can be eliminated if there exists another inequality U and a constant K such that KU implies V. Determine this from analysis of pairwise ratios of the components of V and U.

There are three special cases of unknowns. Corresponding coefficients that are nonnegative in every inequality indicate unknowns that can be unboundedly large, so set their values to 1, all others to 0. Analogously, if there are coefficients which are always nonpositive, set their values to zero, and confine subsequent study to the other unknowns. And if there are unknowns with always zero coefficients, there is nothing to say about them, so give them some default value. These latter two situations may need to further simplications of the kind of the previous paragraph, and possibly furtheriterations.

5.4.5 Hill climbing

Inequalities only pathologically have a unique solution. So "solving" our inequalities can mean finding some representative point in the convex region of hyperspace defined by the inequalities -- some kind of "midpoint", say. Linear programming and least-squares methods are not very helpful for this. Monte Carlo methods are a possibility. However, the approach we take is to try to maximize a

nonlinear aggregate "distance-from-the-boundaries" function. This function derives from our sigmoidal "user indifference" function (3.2) for utility-value indifference. Since indifference is a probability, we combine values from different inequalities (representing the degree in which the particular inequality is satisfied) multiplicatively -- a good approximation even without guaranteed statistical independence. We use:

$$f(X) = \prod_j (\Phi([\mathbf{C}_j\mathbf{X} \quad b_j]/\sigma_{indiff})) \tag{5.5}$$

where X is the vector of unknowns, $\mathbf{C}_j$ the vector of coefficients of the jth inequality, $\mathbf{C}_j\mathbf{X}$ their inner product, b_j the constant in the jth inequality, and σ_{indiff} the user sensitivity in equation (3.2). If inequalities have certainty factors (perhaps derived from certainty factors on preferences), the terms of the above product may be weighted appropriately.

The gradient of this function is straightforward to calculate, but the Hessian (second derivative) matrix is hard. Thus we use a steepest-ascent hill-climbing technique on the inequalities, to maximize the above formula. Step size is adjusted dynamically from the rate of change of the gradient. Since except for pathological cases this function has a single maximum, choice of an initial vector is not critical, and results of previous runs on similar inequalities may provide it. Note a certain amount of error tolerance is possible in formula (5.5), and some inequalities may not actually be satisfied by the "solution".

5.5 Implementation considerations

It is hard to rigorously evaluate this inference method. Domains may differ much in the closeness of choices, and domain-specific inequalities may affect performance greatly. We have implemented a demonstration program and it seems to work for merchant shipping choices; we hope to do a more formal validation on a different database soon. We do note, however:

- Analogously to solving of linear equations, the number of inequalities must be at least the number of unknowns for consistently reasonable results.
- The alteration between utilities and suitabilities may cause divergence if the starting step size for the hill-climbing and the total number of steps per each invocation are poorly chosen. Simple rules-of-thumb can avoid this.
- Computation time is centered in the inequality pruning, and thus design should concentrate on making this efficient.
- Space is not as much a problem as time.

6. Further applications

6.1 System efficiency

6.1 .1 Management of previous query results

The p_i selection probabilities represent mutually exclusive probabilities that the user will choose an item. Thus they may be summed for a set of items to get a cumulative estimate of the completeness of the range of options. This is helpful information for systems that keep for a time the data pages retrieved by previous queries. For each data page one can estimate the probability of selecting some item on that page as the total of the p_i. Since p_i incorporate a detailed theory of degrees of interest, their sum can be a considerably more intelligent judgment of page value than a least-recently-used criterion.

6.1.2 Prefetching

The p_i also allow estimating the value of information before it is fetched. One can designate for each field a set of related fields that might be good to fetch at the same time, based on weight sensitivity to the value of that field; the user may be likely to explicitly request this information anyway subsequently. Or one can weaken restrictions the user gives in order to get extra data items, if his restrictions seem too strong to get much of a yield.

6.2 Cooperative respdnses to queries

6.2.1 Suggestive responses

Quantitative degrees of interest also facilitate "cooperative" indirect answers to queries, in particular when the answer set the user identifies is empty. One can note violated existence preconditions [Kaplan, 1979] and set-inclusion presuppositions [Mays, 1980] without degrees of interest. But those approaches do not work when there is no simple explanation of the empty set. For these cases one can use sensitivity analysis to decide which restrictions to relax with minimum impact on the user's cost function, formulate a weaker query and try again (what [Kaplan, 1979] calls a "suggestive response").

6.2.2 Approximate responses

Correlations (see 3.1.3) can be used to estimate data not yet retrieved from what has. Approximations are suggested whenever correlations are known to be strong and the impact of uncertainty in the unknown value on the cost function analysis is weak.

6.2.3 Noticing user planning errors

A very important application of item selection probabilities is in catching a heretofore undetectable class of user errors in query specification. These are errors in the values he assigns to things, including:

- *Extra items.* **E.g.,** the user asks about the Totor, Pequod, and Bounty, and their computed degrees of interest are .2, .3 and .0001 respectively. The user may have mistakenly included Bounty, perhaps confusing it with a similarly named ship.

- *Omitted items.* Similarly, if the user singles out certain ships, and a few high-weighted ships are left out while many ships with lower selection probabilities are included, the user may have missed or forgotten the former.

- *Errors in scope of symbolic restrictions.* E.g., the user requests ships of a very expensive ship type, roll-on roll-off carriers, in addition to regular ships. This suggests a problem specific to the utility or suitability of certain ship characteristics.

- *Errors in scope of numeric restrictions.* E.g., the user requests ships within 10,000 nautical miles of Naples. Since the degrees of interest for ships 10,000 miles away would be very small under any cost function (it would take forever for them to reach Naples), we suspect a mistake.

- *Restriction sense errors.* E.g., the user asks for ships of less than 10,000 tonnage rather than greater than 10,000. This can be detected as a nonstandard query form, or by an overly large range of answer p_i.

- *Premature termination of a query sequence.* If at the end of a session there is still large uncertainty in the selection probabilities, or they are all very closely spaced together, the user may have forgotten something, since the querying hasn't helped choose among options very much.

6.3 More query capabilities

6.3.1 Queries with examples

We can now handle a new set of queries exemplified by

- Find me fifty ships in the Mediterranean, ideally ones with 12,000 tonnage, over 10 knots cruising speed, at Naples, and available.

The restrictions here provide an "ideal ship" that can be compared to all others, generating inequalities. This is particularly valuable preference information because the user specifically chose the example for illustration.

This can be viewed as an extension of Query-by-Example [Zloof, 1977] where one uses the otherwise-discarded placeholders or field-value "examples" selected.

6.3.2 Fuzzy restrictions

We can also now handle certain queries with "fuzzy" linguistic expressions. For instance:

- What small tankers are at Naples?
- What ships at Naples are probably available?
- Which ships were late at Naples in the past month?

We can use our probabilistic suitability model for this, as a simpler alternative to the algebra of fuzzy sets [Zadeh, 1979]. Terms like "small", "probably", and "late" can be modelled as additional suitability uncertainty. For instance, "small" for a tanker could mean a suitability of .1 for 10,000 tonnage, .5 for 5,000, and .9 for 1,000. Generalizing from several such situations we may be able to describe different kinds of "smallness" in similar ways, differing only in size reference point and compression.

7. Conclusion

We have proposed a detailed methodology for modelling degrees of interest among items in a database. Our approach carefully distinguishes utility and probability information, to provide more sensitivity and flexibility than previous approaches such as term weighting, fixed metrics, and user subschemas. But our approach is more complex, and to handle this complexity certain inference methods -- slot inference, task inference, set preference inference, and parameter estimation from inequalities -- are needed. The balance of this tradeoff will vary from database to database, but we argue that the overhead required for our methods is small compared to the usually large time and space demands of databases and their uses.

References

Appelt, D. E. Problem Solving Applied to Natural Language Generation. In *Proceedings of 78th Annual Meeting.* Philadelphia, Penn.: Association of Computational Linguistics, 1980.

Bobrow, D. G., Kaplan, R. M., Kay, M., Norman, D., Thompson, H., and Winograd, T. GUS, A Frame-Driven Dialog System. *Artificial Intelligence,* 1977, 8, 155-l 73.

Cohen, R. Investigation of Processing Strategies for the Structural Analysis of Arguments. In *Proceedings of 79th Annual Meeting.* Stanford, CA: Association for Computational Linguistics, 1981.

Cohen, P., Perrault, C. R., and Allen, J. *Beyond Question-Answering* (Tech. Rep. 4644). Cambridge MA: Bolt Beranek and Newman Inc., May 1981.

Davis, R. and King, J. An Overview of Production Systems. In E. W. Elcock and D. Michie (Ed.), *Machine Intelligence 8.* New York: Wiley, 1976.

Duda, R. O., Hart, P. E., Nilsson, N. J., Reboh, R., Slocum, J., and Sutherland, G. L. *Development of a Computer-based Consultant for Mineral Exploration* (Annual Report Projects 5821 and 6415). Menlo Park, CA: SRI International Artificial Intelligence Center, October 1977.

Hobbs, J. R. *Why Is Discourse Coherent?* (Technical Note 176). Menlo Park CA: SRI International Artificial Intelligence Center, November 1978.

Kaplan, S. J. *Cooperative Responses from a Portable Natural Language Database Query System* (Tech. Rep. HPP-79-19). Stanford, CA: Stanford University Heuristic Programming Project, July 1979.

Lancaster, F. W. *Information Retrieval Systems.* New York: Wiley, 1979.

Luce, R. D. *Individual Choice Behavior. New* York: Wiley, 1959.

Mays, E. Correcting Misconceptions about Database Structure. In *Proceedings of Biennia/ Conference.* Victoria BC: Canadian Society for Computational Studies of Intelligence, 1980.

Miller, R. B. Archetypes in Man-Computer Problem Solving. *IEEE Transactions on Man-Machine Systems,* December 1969, *MMS*(10), 219-241.

Newsted, P. and Wynne, B. Augmenting Man's Judgement with Interactive Computer Systems. *International Journal of Man-Machine Studies,* 1976,8(l), 29-59.

L. Press. Toward Balanced Man-Machine Systems. *International Journal of Man-Machine Studies,* 1971, *3*, 61-73.

Raiffa, H. *Decision Analysis.* Reading, MA: Addison-Wesley, 1970.

W. Rouse. Design of Man-Computer Interfaces for On-Line Interactive Systems. *Proceedings of the IEEE,* June 1975, *63*(6), 847-857.

C. Spath and L. Schneider. A Generalized End-User Facility Architecture for Relational Database Systems. In *Proceedings.* Conference on Very Large Database Systems, 1977.

Sproull, R. F. *Strategy Construction Using a Synthesis of Heuristic and Decision-Theoretic Methods* (Tech. Rep. CSL-77-2). Palo Alto, CA: XEROX Palo Alto Research Center, July 1977.

Ullman, J. D. *Principles of Database Systems.* Potomac MD: Computer Science Press, 1980.

Wiederhold, G. *Database Design.* New York: McGraw-Hill, 1977.

Yang, C.-S. and **Salton,** G. Best-Match Querying in General Database Systems -- A Language Approach. In *Proceedings.* Chicago IL: IEEE COMPSAC, 1978.

Yu, C. T., Luk, W. S., and Cheung, T. Y. A Statistical Model for Relevance Feedback in Information Retrieval. *Journal of the Association for Computing Machinery,* April **1976, 23(2),** 210-225.

Zadeh, L. A Theory of Approximate Reasoning. In E. W. Elcock and D. Michie (Ed.), *Machine Intelligence 9.* New York: Wiley, 1979.

Zloof, M. M. Query-by-Example: A Data Base Language. *IBM Systems Journal,* 1977, *16*(4), 314-343.