A Probabilistic Framework for Vague Queries and Imprecise Information in Databases

Norbert Fuhr
Technische Hochschule Darmstadt, Fachbereich Informatik
Karolinenplatz 5, D-6100 Darmstadt, West Germany

Abstract

A probabilistic learning model for vague queries and missing or imprecise information in databases is described. Instead of retrieving only a set of answers, our approach yields a ranking of objects from the database in response to a query. By using relevance judgements from the user about the objects retrieved, the ranking for the actual query as well as the overall retrieval quality of the system can be further improved. For specifying different kinds of conditions in vague queries, the notion of vague predicates is introduced. Based on the underlying probabilistic model, also imprecise or missing attribute values can be treated easily. In addition, the corresponding formulas can be applied in combination with standard predicates (from two-valued logic), thus extending standard database systems for coping with missing or imprecise data.

1 Introduction

In most of today's data base management systems (DBMSs), the query language is based on two-valued logic (e.g. relational algebra). This concept implies that for every object stored in a data base, a binary decision can be made by the system whether the object is an answer to the current request or not. Based on this feature, efficient query processing strategies are possible.

On the other hand, handling of user requests that cannot be expressed in two-valued logic is difficult with current DBMSs:

- In engineering applications, when a new part has to be developed, it is often more effective to start with a similar part already constructed, instead of developing the new part from scratch. If the parts are stored in a database, it should be possible to search for parts similar to a given specification. The system described in [Schneider et al. 89] for this purpose is based on a relational database for the retrieval of a set of candidate objects. Then for each object in this set its similarity to the specification is computed.
- In materials data systems [Westbrook & Rumble 83], a large number of requests either seek for materials similar to a known material or for materials that are optimum with respect to a number of criteria. As there is also a large number of missing values for the materials attributes, systems based on Boolean query logic rarely can provide satisfactory answers [Ammersbach et al. 88].
- Business decision making very often means to find an optimum solution with respect to a number of criteria. If the decision relates to items stored in a database, then there should be a method that retrieves items close to the optimum (see the example below).

In all these applications, the query languages of current DBMSs offer little support. Mostly, users are forced to submit a series of queries in order to retrieve some objects that are possible solutions to their problem. Moreover, they often cannot be sure if they tried the query that retrieves the optimum solution.

In the field of information retrieval, similar prob-

Proceedings of the 16th VLDB Conference
Brisbane, Australia 1990

MODEL	CPU	MEMORY	CLOCK_RATE	DISK_SIZE	ACCESS_TIME	PRICE
A	80386	4	20	40	40	1500
B	80386	4	25	80	28	2000
C	80386	4	25	75	26	2000
D	80386	4	25	80	24	3000
E	80386	4	25	85	28	2500

Table 1: Example database about PCs

lems have been investigated for a long time.[1] For this kind of queries, ranking methods have been developed which yield a ranked list instead of a fixed set of documents as an answer to a query. It has been shown in experiments that ranking methods yield significantly better results than search methods based on Boolean logic [Salton et al. 83] [Fuhr 86].

Text retrieval systems with ranking first seek for documents that contain terms from the query. A relevance status value (RSV) is computed for each document, and then documents are ranked according to descending RSVs. For this task, two major theoretical models have been developed:

- In the vector space model [Salton 71] [Wong et al. 87], documents and queries are represented as vectors in a vector space spanned by the terms of the database. For a query-document pair, the RSV is computed by means of a similarity coefficient (e.g. dot product or cosine) of the corresponding vectors.
- In the probabilistic approach [Rijsbergen 79] [Fuhr 88], retrieval is regarded as a stochastic process. So documents are ranked according to their probability of being relevant to the current request. It can be shown that this kind of ranking is optimum [Robertson 77]. The probability of relevance of a document is computed from probabilistic weights of the terms, which in turn are derived from relevance information about (other) query-document pairs.

In this paper we describe how the probabilistic approach can be applied for retrieval of facts from databases. For a vague query, a system based on our approach first will yield an initial ranking of possible answers. Then the user is asked to give relevance judgements for some of the answers, that is, he must decide whether an answer is an acceptable solution to his problem. From this relevance feedback data, the system can derive an improved ranking of the answers for the current request. In addition to this kind of short-term learning, a major new concept of our approach is the collection of feedback data for a long-term learning process: Based on feedback information, an improved weighting of attribute values with respect to query conditions can be derived. This way, better retrieval results for future queries can be achieved.

In order to illustrate the concepts of our approach, we will use an example of a database about personal computers throughout this paper. The database contains information about PCs and consists of a single relation with the attributes processor type, CPU clock rate, memory size, disk size, disk access time and price (see table 1). Now a user may seek for a PC with an 80386 processor with a clock rate of at least 25 MHz, 4 MB of main memory and an 80 MB hard disk with an access time of less than 25 ms. Of course, he is interested in a cheap offer. This example illustrates the close relationship of this kind of problem to the field of information retrieval: It is obvious that the goal of the user, namely to select a single model for purchase, cannot be fully expressed in a query. His final decision will depend on a number of additional factors, which cannot be represented completely in the database. This means that the representation is uncertain and incomplete with respect to the application – the same situation as in text retrieval.

Now look at the sample data set in table 1. Here only model D fulfills all the criteria specified in the query. On the other hand, there are three more models (B, C, E) which do not fully meet the requirements, but which are significantly cheaper than model D. For this reason, the user should be informed about these models, too. It is obvious that interpreting all the criteria specified by the user as predicates in a two-valued logic would yield inappropriate results. Instead, at least some of the criteria should be regarded as vague predicates which can be fulfilled to a certain degree by attribute values.[2] This degree of fulfillment will be called indexing weights in the following.

A second kind of weighting (called query condition

[1] A general discussion of the similarities and differences between information retrieval and database management systems is presented in [Eastman 89], where the issues of evaluation, matching, interaction and clustering are considered.

[2] An approach based on ranking instead of Boolean logic, but using predicates from two-valued logic would e.g. not allow to distinguish between the access times of model B and C.

weighting below) refers to the different criteria specified by the user, which may not be of equal importance for him. For example, disk size may be more important than access time, so the answers should be ranked accordingly. We will describe two probabilistic retrieval models for this purpose which allow either an explicit weighting as specified by the user or an implicit one derived from his relevance judgements.

In addition to these two kinds of probabilistic weighting, our approach has the following features:

- Different vague predicates can be considered, like e.g. "about", "at most", "high", "low", "some".
- The approach can be applied to arbitrary data types, ranging from numbers or strings to complex objects like e.g. the shape of a geometric object.
- With the probabilistic foundation of our approach, imprecise data (e.g. disjunctive information) or missing data (i.e. null values) also can be handled easily, even for predicates from two-valued logic.

The paper is structured in the following manner. First, we present the foundations of our approach (section 2) and describe the extension to imprecise data (section 3). Two different retrieval functions for the computation of the RSVs are presented in section 4. Finally, our approach is compared with similar work (section 5).

2 Foundations of the probabilistic model

We propose our approach as an extension of database systems based on two-valued logic. A user query in such a system can be extended by a part in which one or more vague criteria can be specified.

Definition 1 *An extended query is a combination of a Boolean query and a vague query. The answer for the Boolean query is a set of objects from the database called preselected objects. The answer to the extended query is a ranked list of the preselected objects.*

In the following discussion, unless stated otherwise, we will restrict to the vague part of the query. Furthermore, no distinction between the whole database and the set of preselected objects will be made.

For our PC example, an extended query in a SQL-like notation could be specified as follows:

```
SELECT * FROM PC
   WHERE CPU    = '80386'
   AND   MEMORY = 4
   RANK_BY CLOCK_RATE  >= 25,
           DISK_SIZE   >= 80,
           ACCESS_TIME <  25,
           PRICE LOW.
```

Definition 2 *A database is a set of objects O. Let $A = \{a_1, \ldots, a_n\}$ denote the set of attributes in the database, and D_i the domain for attribute a_i. Then each object $o_m \in O$ is represented by a tuple $t_m =< t_m(a_1), \ldots, t_m(a_n) >$ with $t_m(a_i) \in D_i$.*

For an extension of the linear data model assumed here, see the discussion of probabilistic databases in section 5. It should be emphasized that our approach makes no assumptions about the domain of an attribute, so attribute values can be of a complex data type.

Now we give the definitions for the vague queries.

Definition 3 *A vague (or fuzzy) predicate f is either a unary predicate or a binary predicate. For each attribute $a_i \in A$ in a database, there is a set F_i^1 of unary attributes defined and a set F_i^2 of binary attributes defined.*

Note that a predicate is not a mapping of attribute values (or pairs of values) onto numbers, like e.g. in fuzzy logic [Zadeh 65]. In our approach, the weighting of attribute values takes place in a later stage of the indexing process. Examples for unary predicates are "low", "high", "medium" and also so-called fuzzy quantifiers like "some", "several", "many" (e.g. a user might ask for a PC with "several" I/O-ports). Most binary vague predicates will be vague interpretations of the standard predicates like e.g. "$=$", "$<$", "$\geq$".

Definition 4 *A vague query condition c_i can have one of the two forms*

- *(a_i, f_i) with $a_i \in A$ and $f_i \in F_i^1$*
- *(a_i, f_i, d_i) with $a_i \in A$, $f_i \in F_i^2$ and $d_i \in D_i$, where d_i is called the comparison value.*

Definition 5 *A vague query formulation q_k^c is a set of vague conditions c_i. Furthermore, each attribute $a_i \in A$ may occur in at most one condition $c_i \in q_k^c$.*

For our PC example, the vague query formulation is

$$q^c = \{(\text{CLOCK\_RATE}, \geq, 25), (\text{DISK\_SIZE}, \geq, 80), (\text{ACCESS\_TIME}, <, 25), (\text{PRICE}, \text{low})\}$$

The restriction that an attribute may occur in at most one query condition is due to the independence assumptions of the underlying probabilistic models (see [Fuhr 89a]). We do not allow Boolean operators in the query formulation, for two reasons:

- A Boolean structure would lead to rather complex probabilistic formulas, with parameters that could not be estimated in most cases.
- In the field of text retrieval, there is no experimental evidence that the consideration of a Boolean query structure yields any improvement over a pure linear structure [Croft 86] [Salton & Voorhees 85] [Fuhr & Müller 87]. On the other hand, it can be shown that many proposals for ranking in combination with Boolean queries yield significantly worse retrieval results than ranking for linear queries [Salton et al. 83] [Fuhr 86] [Fuhr 88].

However, we can extend our model to allow for the disjunction or conjunction of conditions for the same attribute (see next section).

Definition 6 *Let $\mathcal{R} = \{R, \overline{R}\}$ (relevant/nonrelevant) denote the set of possible relevance judgements for query-object pairs. Then a vague query q_k is a pair $q_k = (q_k^c, q_k^J)$ with $q_k^J \subset O \times \mathcal{R}$.*

For databases, the notion of relevance judgements may be inconvenient. Furthermore, in many applications a user may seek only for a single object as the optimum solution to his problem. Considering only this object as being relevant would prevent the application of relevance feedback techniques, at least for the weighting of query conditions. For this reason, it is more appropriate to use a concept like 'acceptability' for the user's judgements: In general, there will be several acceptable objects for a query, among which the user may finally decide for one (or more) as being the optimum solution. So we will assume that the users give binary judgements about the acceptability of objects. For convenience, however, we will keep on using the term 'relevance' for this event.

The event space of our probabilistic model is $Q \times O$ where Q denotes the set of all queries submitted to the database system. A single element of this event space is a query-object pair (q_k, o_m) with a binary[3] relevance judgement $\in \mathcal{R}$, a set q_k^c of query conditions and a tupel t_m of object attributes. Now we seek for an estimate of the probability $P(R|q_k, t_m)$ that an object with the attribute tuple[4] t_m will be judged relevant with respect to query q_k. The estimation of this probability by means of a retrieval function (as a basis for the ranking of the objects) is described in section 4.

The retrieval function needs indexing weights for all the conditions in q_k^c for this purpose. For the probabilistic definition of these weights, we introduce the additional concept of correctness as an attribute of an object-condition relationship: An object may be a correct answer to a single condition, or may not. We will denote these events by the symbols C and $\overline{C}$, respectively. The decision about the correctness of an object-condition pair can be specified explicitly, that is, in addition to the relevance jugdement, the user will have to judge an object with respect to each condition. However, it is also possible to derive these decisions from the relevance judgement of a query-object pair: If the pair is relevant, then the object is correct with respect to all query conditions. In the opposite case, the object is not a correct answer for any of the query conditions. The latter definition forms the basis of the binary independence indexing (BII) model [Fuhr 89a]. The retrieval-with-probabilistic-indexing (RPI) model described in the same paper – which we apply for the task of query condition weighting based on relevance feedback data (see section 4) – can be combined with both definitions. The RPI model only assumes a positive correlation between the events of relevance and correctness. Experiments in the field of automatic text indexing have shown that both definitions of the concept of correctness can be applied successfully and lead to similar results [Fuhr 89a]. Further experiments with the second definition can be found in [Fuhr & Buckley 90].

With the event of correctness defined in either of the two ways, we seek for estimates of the probability $P(C|c_i, t_m(a_i))$ that an object with attribute value $t_m(a_i)$ is a correct answer to the query condition c_i. The estimation of this probability is performed by the indexing task.

The indexing task consists of two steps, a description step and a decision step. In the first step, all information available about the relationship between the condition c_i and the attribute value $t_m(a_i)$ is collected in the so-called relevance description $x(c_i, t_m(a_i))$. Based on this data, the decision step yields an estimate of the probability $P(C|x(c_i, t_m(a_i)))$ that an object-condition pair de-

[3] For the discussion of multivalued relevance scales, see section 5.

[4] As the probabilistic model is not able to make a difference between two objects having the same tuple of attribute values, we use the notation t_m instead of o_m in $P(R|q_k, t_m)$.

scribed by relevance description x will be judged correct.

In both steps, the different vague predicates have to be treated separately (specific relevance descriptions and estimation of indexing weights for each predicate). To keep the following explanations simple, we assume that we always regard only a single predicate in the indexing process, and that this process has to be repeated for each predicate.

Definition 7 *A relevance description $x(c_i, t_m(a_i))$ is a data structure that describes properties of the relationship between the condition c_i and the attribute value $t_m(a_i)$.*

Here no specific assumptions about the structure and the elements of a relevance description are necessary. However, some algorithms used in the decision step are restricted to certain types of relevance descriptions. Furthermore, it is essential that only those properties are useful as elements of the relevance description for which there is a significant correlation with the event of correctness.

This concept of relevance description yields an abstraction from specific pairs (condition, attribute value). For the binary predicates in the PC example, one could define

$$x(c_i, t_m(a_i)) = \frac{t_m(a_i) - d_i}{d_i}.$$

This way, all pairs with the same relative difference between comparison value and attribute value would have identical relevance descriptions. For unary predicates, the relevance description can be defined with respect to the distribution of the attribute values in the database, e.g. the percentage of values smaller than $t_m(a_i)$.

The actual definition of the relevance description depends strongly on the application. Its elements can be values from interval scales as well as from ordinal or nominal scales (e.g. for string comparison a marker indicating whether there is a phonetic match between the two strings or not). Especially distances based on the different metrics described in [Motro 88] will provide useful information for relevance descriptions. It should be noted that a relevance description may consist of several components, that is, we can cope also with multi-dimensional metrics. This feature is important when dealing with complex data types.

In the decision step, estimates of the probabilities $P(C|x(c_i, t_m(a_i)))$ are computed. For this purpose, we need a learning sample of relevance descriptions and corresponding decisions about the correctness from previous user queries.

Now, one could estimate the probability $P(C|x(c_i, t_m(a_i)))$ as relative frequency from those elements of the learning sample that have the same relevance description (components of x with continuous values would have to be discretized before, see e.g. [Wong & Chiu 87]). At this point, we introduce the concept of an indexing function:

Definition 8 *Let X denote the set of relevance descriptions and $\mathbb{R}$ the set of real numbers. Then a probabilistic indexing function is a mapping $e : X \rightarrow \mathbb{R}$ such that $e(x)$ is an approximation of $P(C|x)$. We call $w(c_i, t_m(a_i)) = e(x(c_i, t_m(a_i)))$ the indexing weight of the attribute value $t_m(a_i)$ with respect to the condition c_i.*

As indexing function, different probabilistic classification (or learning) algorithms can be applied. The general advantage of these probabilistic algorithms over simple estimation from relative frequencies is that they yield better estimates from a learning sample given, because they use additional (plausible) assumptions about the indexing function.

Here we list some probabilistic algorithms that can be used as indexing functions. All these algorithms have been applied successfully to relevance descriptions in the field of text retrieval for the task of automatic indexing with a controlled vocabulary [Biebricher et al. 88]. With the exception of the first algorithm, these methods are restricted to a vector form $\vec{x}$ of the relevance description. For the first three algorithms, all the elements of the relevance description must have discrete values.

- The so-called Boolean approach developed by Lustig [Beinke-Geiser et al. 86] exploits prior knowledge about the relationship between single elements of the relevance description x and the corresponding probability $P(C|x)$ for the development of a discrete indexing function (e.g. in our PC example, it can be assumed that the probability for the predicate $\geq$ is a monotonic function of the relative difference between d_i and $t_m(a_i)$)).
- The probabilistic learning algorithm ID3 developed by Quinlan [Quinlan 86] seeks for significant components of $\vec{x}$ that form a probabilistic classification tree [Faißt 90].
- By assuming only pair-wise dependencies among the components of $\vec{x}$, one can apply the tree dependence model described in [Chow & Liu 68] [Rijsbergen 77] [Yu et al. 83] as indexing function [Tietze 89].
- For the application of regression methods, the components of $\vec{x}$ must be real numbers. Least

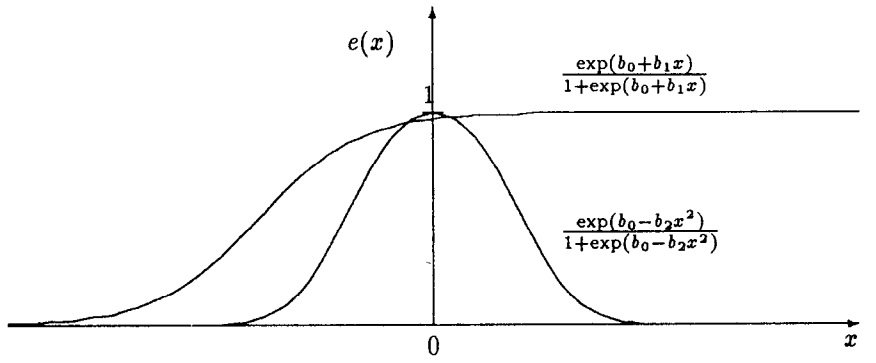

Figure 1: Examples of logistic indexing functions

square polynomials [Knorz 83] [Fuhr 89b] yield indexing functions of the form $e(\vec{x}) = \vec{b}^T \cdot \vec{x}$ (in the linear case), where $\vec{b}$ is a coefficient vector that minimizes the expectation of the squared error $|P(C|\vec{x}) - \vec{b}^T \cdot \vec{x}|^2$.

- By means of logistic regression [Freeman 87] [Fienberg 80], indexing functions of the form $e(\vec{x}) = \frac{exp(b(\vec{x}))}{1+exp(b(\vec{x}))}$ can be developed, where $b(\vec{x})$ is a polynomial, and the coefficients of $b(\vec{x})$ are estimated based on the maximum likelihood method [Pfeifer 90].

Logistic functions seem to be suited best to our indexing task. As an example, assume that the only component of a relevance description is defined as the relative difference between d_i and $t_m(a_i)$. Now one can define an indexing function for the vague predicate '$\geq$' as $e_1(x) = \frac{exp(b_0+b_1x)}{1+exp(b_0+b_1x)}$. In the case of the vague predicate '$=$', an appropriate indexing function would be $e_2(x) = \frac{exp(b_0-b_2x^2)}{1+exp(b_0-b_2x^2)}$. Both functions are illustrated in figure 1. As can be seen from this figure, a major advantage of logistic functions is their asymptotic behaviour. A second advantage (in comparison to polynomial functions) is related to the problem of parameter estimation: When only small learning samples are available, a prior distribution on the coefficients of $b(\vec{x})$ can be considered, thus yielding a Bayesian estimate.

As another nice property of logistic indexing functions, they support the view of vague predicates being deformations of predicates from two-valued logic: In our example, we get for $b_1 \rightarrow \infty$

$$e_1(x) = \begin{cases} 0 & \text{, if } x < 0 \\ 1-\varepsilon & \text{, if } x = 0 \\ 1 & \text{, if } x > 0 \end{cases}$$

with $\varepsilon = 1/(1 + exp(b_0))$. This result corresponds to the strict interpretation (from two-valued logic) of the predicate '$\geq$'. Similarly, we get for $b_2 \rightarrow \infty$ in $e_2(x)$ the strict interpretation of the equality predicate, namely $e_2(x) = 0$, if $x \neq 0$ and $e_2(0) = 1 - \varepsilon$.

3 Indexing for missing or imprecise data

Our probabilistic indexing approach can be extended to handle also imprecise or missing data as attribute values and sets of values as comparison value. We first discuss the case of attribute values, then comparison values, and finally we show how these methods can be applied for predicates from two-valued logic, too.

Imprecise data for attribute values can be disjunctive information (e.g. assume that we only know that the clock rate of a PC model is either 25 or 33 MHz). Another major reason for imprecise attribute values is the limited precision of measurement values in technical applications. In order to handle imprecise data, a probability distribution function must be given as attribute value.[5] For attributes with continuous values, imprecise data can be specified e.g.

[5] Probability distributions as attribute values are also

as an interval range or as a pair of mean and variance (for which a normal distribution is assumed). In the following we will only discuss discrete probability distributions, since the extension to continous probability distributions is obvious.

Missing data – that is, an attribute value exists, but is not known – is often stored as null values in databases. Here also an appropriate probability distribution has to be assumed, e.g. by taking the distribution of the corresponding attribute values from similar objects or from the whole database. In most applications, an unknown attribute value does not mean that nothing at all is known about this value. Therefore, the assumption of an appropriate distribution allows to store the information actually available in the database, and to use this information in retrieval. For this reason, we regard missing values as a variant of imprecise data.

We can further extend our approach to a second interpretation of null values discussed sometimes ([Vassiliou 79] [Codd 86]), namely "attribute value not existent". For example, in an address data base, a value of the attribute "telephone number" does not exist for persons who do not have a telephone.

Now we show how these values can be handled by probabilistic indexing functions.

Definition 9 *An imprecise attribute value $t_m(a_i)$ must be specified as a discrete probability distribution over D_i, that is*

$$t_m(a_i) = \{(z_j, p_j) | z_j \in D_i \text{ and } p_j \in [0,1]\}$$

with

$$\sum_{(z_j,p_j)\in t_m(a_i)} p_j = \alpha_{im}, \quad 0 \leq \alpha_{im} \leq 1.$$

This definition covers both interpretations of null values as well as the usual interpretation of imprecise data: If $\alpha_{im} = 1$, we certainly know that an attribute value exists, and with $\alpha_{im} = 0$, we represent the fact that no value exists for this attribute. In the case of $0 < \alpha_{im} < 1$, α_{im} gives the probability that an attribute value exists: For example, someone who is going to have a telephone soon gave us his number, but we are not sure if this number is valid already.

With imprecise values specified this way, their probabilistic indexing weight can be derived easily:

$$P(C|c_i, t_m(a_i)) = \sum_{(z_j,p_j)\in t_m(a_i)} p_j \cdot P(C|x(c_i, z_j)).$$

discussed in the field of numerical taxonomy, see e.g. [Jardine & Sibson 77].

As our indexing function yields approximations of the probabilistic indexing weights, we can define appropriate indexing formulas for imprecise values. In contrast to the case of precise values, we cannot show that these formulas yield optimum approximations (with respect to certain criteria). However, it is reasonable to assume that there is no significant difference between the approximation defined below and the optimum approximation.

Definition 10 *If $t_m(a_i)$ is an imprecise attribute value, then the indexing weight $w(c_i, t_m(a_j))$ is computed by the formula*

$$w(c_i, t_m(a_i)) = \sum_{(z_j,p_j)\in t_m(a_i)} p_j \cdot e(x(c_i, z_j)).$$

Our approach can be further extended to consider the disjunction or conjunction of conditions for the same attribute. Both variants are in fact syntactic elements for specifying a set of values, so we call this imprecise data as comparison value. A major need for imprecise comparison values comes from technical applications, where very often interval ranges for certain attributes are specified in the query [Dathe 84].

Definition 11 *An imprecise comparison value is a set $d_i \subseteq D_i$, where D_i is the domain of the corresponding attribute.*

The correct handling of imprecise data as comparison value depends on the specific predicate, the data type of the attribute and the type of the specification of the imprecise data (e.g. set of values vs. interval range). Two possible strategies can be applied here:

- In many cases, the indexing function for a single comparison value can be applied by object-specific selection of an appropriate value from the set or range of comparison values specified in the condition. For example, when an interval range is specified in combination with the predicate '=', then the value from the interval closest to the current attribute value is selected. As a counterexample, consider the query "List all the PC models from the manufacturer whose name is similar to 'Dandy' or 'Dundee'". For the manufacturer 'Tandy', both comparison values might yield nonzero indexing weights, and there is no simple, theoretically founded method for combining these weights.
- At a closer look, it becomes obvious that imprecise data is just another variant of a vague condition: For example, there is no systematic difference whether we specify a predicate like '>' for an attribute with continuous values or a set of constants in combination with the equality predicate for a nominal-scaled attribute. This view

leads us to the development of special relevance descriptions and indexing functions for imprecise data. With this method, we can also handle rather complex specifications of imprecise data (not covered by the definition from above). As an example, assume that a user wants a PC with a green or a black and white monitor, but he prefers black and white. However, this strategy is only applicable if there is enough learning data available, that is, imprecise comparison values are used frequently in query formulations. Otherwise, one can only attempt to define appropriate indexing functions, without adaptation to the user population.

These methods for coping with imprecise data as comparison or attribute value also can be applied to predicates from two-valued logic. For this case, a binary indexing function is defined:

Definition 12 *Let p denote a predicate from two-valued logic and $d_i \subset D_i$ an imprecise comparison value. Then the indexing function is defined as*

$$e((a_i, p, d_i), t_m(a_i)) = \begin{cases} 1, & \text{if } p(z, t_m(a_i)) \text{ for any } z \in d_i \\ 0, & \text{otherwise} \end{cases}$$

For imprecise or missing attribute values, the same formulas as for vague predicates can be applied. This strategy yields the answers that a procedure for disjunctive information in two-valued logic would retrieve. For example, for the query condition $c_i = (a_i, =, \{z_1, z_2\})$ and the attribute value $t_m(a_i) = \{(z_1, \alpha), (z_2, 1-\alpha)\}$, our procedure assigns an indexing weight of 1 to the attribute value. But our approach also yields valuable results when the object is not an answer in two-valued logic: If the query condition is only $(a_i, =, z_1)$, then in general the object with attribute value $t_m(a_i) = \{(z_1, \alpha), (z_2, 1-\alpha)\}$ will be ranked ahead of all objects with null values, which in turn are ranked ahead of objects with different attribute values.

4 Retrieval functions

The task of the retrieval function $\varrho(q_k, o_m)$ is to compute relevance status values (RSVs) for query-object pairs (q_k, o_m). Then the objects can be ranked according to descending RSVs for a query. This way, a user will find the objects probably relevant at the beginning of the ranked list.

Definition 13 *A retrieval function is a mapping $\varrho : Q \times O \rightarrow \mathbb{R}$.*

Here we will discuss two different probabilistic rerieval functions, one for the initial ranking and another for an improved ranking based on relevance feedback from the user. For both functions we give only a brief description, for the details of the underlying models the reader is referred to the original publications.

In order to compute the RSV, the retrieval functions use the indexing weights $w(c_i, t_m(a_i))$ for the conditions $c_i \in q_k^c$ (the set of conditions specified in the query). Now a simple linear retrieval function can be defined as

$$\varrho_{lin}(q_k, o_m) = \sum_{c_i \in q_k^c} u_{ik} \cdot w(c_i, t_m(a_i)).$$

Here u_{ik} is a factor which reflects the importance of the condition c_i within the query q_k. These factors can be specified explicitely by the user (For the combination with relevance feedback, see the discussion below). In [Wong & Yao 89], it is shown that the above formula can be given a utility-theoretic interpretation: If u_{ik} denotes the utility of the condition c_i with respect to the query q_k (and the underlying definition of correctness), then $\varrho_{lin}(q_k, o_m)$ gives the expected utility of object o_m for the query q_k.

After the initial ranking process, the user is asked to give relevance judgements for the top ranking objects. This relevance feedback data (which we regard as a part q_k^J of a vague query q_k, see Definition 6) can be used for a better weighting of the query conditions. In the field of text retrieval, evaluations of relevance feedback methods have shown significant improvements over the initial ranking (see e.g. [Robertson & Sparck Jones 76] [Yu & Salton 76] [Robertson et al. 81]). For our application with probabilistic indexing weights, the RPI model described in [Fuhr 89a] is most appropriate. This model yields a ranking according to descending values of the probability of relevance $P(R|q_k, t_m)$. The corresponding retrieval function is

$$\varrho_{RPI}(q_k, o_m) =$$

$$\prod_{c_i \in q_k^c} \left[\left(\frac{r_{ik}(1-s_{ik})}{s_{ik}(1-r_{ik})} - 1 \right) \cdot w(c_i, t_m(a_i)) + 1 \right]$$

In this formula, r_{ik} gives the expectation of the indexing weight of c_i for an arbitrary object that is jugded relevant with respect to q_k. Similarly, s_{ik} is the expectation of the indexing weight of c_i for

an arbitrary object in the set of preselected objects for q_k. These factors can be estimated by means of relevance feedback as follows: Let O_k denote the set of preselected objects for q_k (or a representative sample hereof), and O_k^R is the set of objects judged relevant for q_k, that is, $O_k^R = \{o_m | (o_m, R) \in q_k^J\}$. Then we get

$$r_{ik} \approx \frac{1}{|O_k^R|} \cdot \sum_{o_m \in O_k^R} w(c_i, t_m(a_i))$$

and

$$s_{ik} \approx \frac{1}{|O_k|} \cdot \sum_{o_m \in O_k} w(c_i, t_m(a_i))$$

In [Croft 81], the linear retrieval function ϱ_{lin} is used in combination with relevance feedback weights defined as

$$u_{ik} = \log \frac{r_{ik}(1 - s_{ik})}{s_{ik}(1 - r_{ik})}$$

It can be shown theoretically that this approach does not yield a probabilistic ranking [Fuhr 89a]. However, as far as experimental comparisons are available [Fuhr 89a] [Salton 87], no significant differences could be found.

With this kind of relevance feedback, we make double use of the relevance data: The user enters relevance judgements in order to get a better ranking for his query. In addition, we collect his judgements for a long-term improvement of the system by developing probabilistic indexing functions based on this data.

5 Comparison with other approaches

There is growing interest in the issue of imprecision in database. The collection of articles in [IEEE 89] gives a survey over past and current work in this area. In the following, we will compare some of these approaches with the model described in this paper.

A similar approach for vague queries in databases is the VAGUE system developed by Motro [Motro 88]. This approach is based on the vector space model in information retrieval [Salton 71] [Wong et al. 87], but extended with plausible definitions for coping with Boolean query formulations. In the VAGUE system, a number of different metrics for the comparison of attribute values and values specified in the query are available. These metrics can be used as elements of relevance descriptions in our approach. In addition, we can consider multidimensional metrics and values from nominal or ordinal scales, and all these values are finally mapped onto probabilistic weights. The VAGUE system does not use any relevance information, although it could be extended according to the vector space model in order to improve query-specific ranking. However, there is no method for the improvement of the indexing function (i.e. the metrics) based on relevance information in the vector space model. Different predicates have not been considered explicitly in the VAGUE approach, but the possibility of choosing a metrics (for the 'similar' predicate) could be extended easily for this purpose. With regard to missing or imprecise data, the VAGUE system can handle null values only by defining appropriate distances.

The issue of vague queries and imprecise or missing data has also been addressed in the context of fuzzy databases [Buckles & Petry 82] [Prade & Testemale 84]. This approach offers solutions to all the problems mentioned in this paper. Regarding the theoretical foundations, both approaches are orthogonal to each other: Whereas probability theory deals with the uncertainty about the occurrence of events (here: relevance and correctness), fuzzy logic focuses on the ambiguity in describing events. Thus, while our approach aims to estimate the probability of an object being relevant to a query, fuzzy logic would compute a value that resembles the degree of relevance of the object w.r.t. the query. In [Bookstein 83], it is shown how both approaches can be combined in the context of information retrieval by regarding multivalued relevance scales within a probabilistic model. But the experimental results described in [Fuhr 89b] give no evidence that multivalued relevance scales yield any improvement (in terms of retrieval quality) over binary scales. So the question remains: Which of the two approaches should be preferred for the kind of applications discussed here? We think that the following two facts support the probabilistic view:

- For the probabilistic approach, it can be shown theoretically that this method yields an optimum retrieval quality (under certain assumptions), even for multivalued relevance scales [Robertson 77] [Bookstein 83]. Such a proof does not exist for fuzzy theory. This theoretic statement is supported by experimental results from text retrieval, where the fuzzy model yields significantly worse results in comparison to the probabilistic and the vector space model [Salton et al. 83] [Fuhr 88]. Of course, the assumptions underlying the probabilistic model are only approximations to reality. But as they

are made explicit, they can be replaced by more appropriate assumptions, and refined models can be developed.

- The weights in the probabilistic approach all have the explicit notion of probabilities. This feature is the basis of our indexing and retrieval approach, where the weights are derived from relevance feedback information. Evaluations in the field of text retrieval have shown significant improvements of retrieval quality for relevance feedback methods. In contrast, this kind of adaptation to a single query as well as to a user population is not possible in the fuzzy approach. Similarly, the probabilistic interpretation of imprecise attribute values forms a guideline for the mapping of empirical data onto these values.

A simple ranking scheme for relational databases has been proposed in [Lacroix & Lavency 87]. This approach is based on predicates from two-valued logic. For the weighting of query conditions, preferences between the different conditions can be specified by the user: Two conditions can either have the same weight, or one is preferred over the other. As this kind of weighting can easily be considered by the linear retrieval function described in the previous section, the approach from Lacroix and Lavency can be regarded as a special case of our model.

For imprecise or missing data in combination with predicates from two-valued logic, an approach similar to ours is described in [Morrissey & Rijsbergen 87]. This model yields the same indexing weights as in our approach, although no explicit probabilistic model is described.

The problem of imprecise attribute values in the form of null values or disjunctive information has been discussed extensively in the database literature (see e.g. [Lipski 79] [Vassiliou 79] [Reiter 84] [Imielinski & Lipski 84]); [Imielinski 89] gives a brief survey over this work. As all these approaches are based on two-valued logic, the correct treatment of imprecise values is obvious (e.g. in the proof-theoretic approach [Reiter 84], it follows from the axioms of first-order logic). However, as shown in [Imielinski 86], the introduction of imprecise attribute values can have drastic consequences on the complexity of query processing. In [Codd 86], a three-valued logic is used instead, in order to retrieve 'maybe' answers in addition to the correct answers of a query. This approach can be regarded as a very simple ranking mechanism.

Probabilistic models for imprecise information in databases are described in [Cavallo & Pittarelli 87] and [Barbara et al. 90]. The latter article outlines a relational probabilistic data model (PDM) which is similar to our approach. Besides (stochastically) independent attributes as assumed here, Barbara et al. also regard interdependent attributes, which are not considered in the current formulation of our approach. In order to cope with the probabilistic weights, the relational operations projection, join and selection are redefined in the PDM, but vague queries are not regarded. With these features, a combination of both approaches seems to be feasible: The operations redefined for probabilistic databases can be used for handling imprecise data in relational databases and for deriving the answer relation. Then our probabilistic model, which can be regarded as an extension of the selection operation, is applied for the ranking of the tuples in the answer set.

6 Conclusions

In this paper, we have described a probabilistic approach for handling vague queries and imprecise information in databases. The old database paradigm of using two-valued logic is appropriate for the classical application areas of databases, and for the retrieval interface to batch programs. For interactive user interfaces of database systems, new retrieval strategies must be implemented. In parallel, new evaluation criteria for database systems should be considered: Instead of measuring only the efficiency of a system in retrieving answers to well-formed queries, now the effectiveness of the system with regard to supporting the user in solving his problems should be regarded. From this point of view, a fast system that retrieves only sets of answers for Boolean queries may be of little value, because it forces the user to submit a whole series of queries in order to solve his problem.

As new application areas for database systems arise, the concepts of vagueness and imprecise information become even more important. Technical values are almost always of limited precision, and approaches based on two-valued logic are not appropriate for solving the corresponding problems. When new technical solutions are to be developed, the concepts of vagueness and similarity play a central role in this search process. Database systems can offer an effective support for these new applications only when they provide the appropriate concepts.

References

Ammersbach, K.; Fuhr, N.; Knorz, G. (1988). Empirically Based Concepts for Materials Data

Systems. In: *Proceedings of the 1988 CODATA Conference.* Karlsruhe, Germany.

Barbara, D.; Garcia-Molina, H.; Porter, D. (1990). A Probabilistic Relational Data Model. In: Bancilhon, F.; Thanos, C.; Tsichritzis, D. (ed.): *Advances in Database Technology - EDBT '90*, pages 60–74. Springer, Berlin et al.

Beinke-Geiser, U.; Lustig, G.; Putze-Meier, G. (1986). Indexieren mit dem System DAISY. In: Lustig, G. (ed.): *Automatische Indexierung zwischen Forschung und Anwendung*, pages 73–97. Olms, Hildesheim.

Biebricher, P.; Fuhr, N.; Knorz, G.; Lustig, G.; Schwantner, M. (1988). The Automatic Indexing System AIR/PHYS - from Research to Application. In: Chiaramella, Y. (ed.): *11th International Conference on Research and Development in Information Retrieval*, pages 333–342. Presses Universitaires de Grenoble, Grenoble, France.

Bookstein, A. (1983). Outline of a General Probabilistic Retrieval Model. *Journal of Documentation 39(2)*, pages 63–72.

Buckles, B.; Petry, F. (1982). A Fuzzy Representation of Data for Relational Databases. *Fuzzy Sets and Systems 7*, pages 213–226.

Cavallo, R.; Pittarelli, M. (1987). The Theory of Probabilistic Databases. In: *Proceedings of the 13th International Conference on Very Large Databases*, pages 71–81. Morgan Kaufman, Los Altos, Cal.

Chow, C.; Liu, C. (1968). Approximating Discrete Probability Distributions with Dependence Trees. *IEEE Transactions on Information Theory 14(3)*, pages 462–467.

Codd, E. F. (1986). Missing Information (Applicable and Inapplicable) in Relational Databases. *SIGMOD RECORD 15(4)*, pages 53–78.

Croft, W. (1981). Document Representation in Probabilistic Models of Information Retrieval. *Journal of the American Society for Information Science 32*, pages 451–457.

Croft, W. B. (1986). Boolean Queries and Term Dependencies in Probabilistic Retrieval Models. *Journal of the American Society for Information Science 37(2)*, pages 71–77.

Dathe, G. (1984). Peculiarities and Problems of Materials Engineering Data. In: *Proceedings of the 9th International CODATA Conference, Jerusalem.* North-Holland Physics Publishing, Amsterdam.

Eastman, C. (1989). Approximate Retrieval: A Comparison of Information Retrieval and Database Management Systems. *IEEE Data Engineering 12(2)*, pages 41–45.

Faißt, S. (1990). *Development of Indexing Functions Based on Probabilistic Decision Trees (in German).* Diploma thesis, TH Darmstadt, FB Informatik, Datenverwaltungssysteme II (in preparation).

Fienberg, S. (1980). *The Analysis of Cross-Classified Categorial Data.* MIT Press, Cambridge, Mass., 2. edition.

Freeman, D. (1987). *Applied Categorial Data Analysis.* Dekker, New York.

Fuhr, N.; Buckley, C. (1990). *Probabilistic Document Indexing from Relevance Feedback Data.* To appear in: Proceedings of the 13th ACM-SIGIR International Conference on Research and Development in Information Retrieval.

Fuhr, N.; Müller, P. (1987). Probabilistic Search Term Weighting - Some Negative Results. In: van Rijsbergen, C.; Yu, C. (ed.): *Proceedings of the 1987 ACM Conference on Research and Development in Information Retrieval*, pages 13–18. ACM, New York.

Fuhr, N. (1986). Rankingexperimente mit gewichteter Indexierung. In: Deutsche Gesellschaft für Dokumentation (ed.): *Deutscher Dokumentartag 1985*, pages 222–238. K.G. Saur, München, New York, London, Paris.

Fuhr, N. (1988). *Probabilistisches Indexing und Retrieval.* Dissertation, TH Darmstadt, Fachbereich Informatik. Available from: Fachinformationszentrum Karlsruhe, D-7514 Eggenstein-Leopoldshafen, West Germany.

Fuhr, N. (1989a). Models for Retrieval with Probabilistic Indexing. *Information Processing and Management 25(1)*, pages 55–72.

Fuhr, N. (1989b). Optimum Polynomial Retrieval Functions Based on the Probability Ranking Principle. *ACM Transactions on Information Systems 7(3)*, pages 183–204.

IEEE. (1989). *IEEE Data Engineering 12(2).* Special Issue on Imprecision in Databases.

Imielinski, T.; Lipski, W. (1984). Incomplete Information in Relational Databases. *Journal of the ACM 31(4)*, pages 761–791.

Imielinski, T. (1986). *Query Processing in Deductive Databases with Incomplete Information.* Technical report, Department of Computer Science, Rutgers University.

Imielinski, T. (1989). Incomplete Information in Logical Databases. *IEEE Data Engineering 12(2)*, pages 29–40.

Jardine, N.; Sibson, R. (1977). *Mathematical Taxonomy.* Wiley, London et al.

Knorz, G. (1983). *Automatisches Indexieren als Erkennen abstrakter Objekte.* Niemeyer, Tübingen.

Lacroix, M.; Lavency, P. (1987). Preferences: Putting more Knowledge into Queries. In: *Proceedings of the 13th International Conference on Very Large Databases*, pages 217–225. Morgan Kaufman, Los Altos, Cal.

Lipski, W. (1979). On Semantic Issues Connected with Incomplete Information Databases. *ACM Transactions on Database Systems 4(3)*, pages 262–296.

Morrissey, J.; van Rijsbergen, C. J. (1987). A Formal Treatment of Missing and Imprecise Information. In: Yu, C. T.; van Rijsbergen, C. (ed.): *Proceedings of the Tenth Annual ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 149–156. ACM, New York.

Motro, A. (1988). VAGUE: A User Interface to Relational Databases that Permits Vague Queries. *ACM Transactions on Office Information Systems 6(3)*, pages 187–214.

Pfeifer, U. (1990). *Development of Log-Linear and Linear-Iterative Indexing Functions (in German).* Diploma thesis, TH Darmstadt, FB Informatik, Datenverwaltungssysteme II.

Prade, H.; Testemale, C. (1984). Generalizing Database Relational Algebra for the Treatment of Incomplete/Uncertain Information and Vague Queries. *Information Science 34*, pages 115–143.

Quinlan, J. (1986). The Effect of Noise on Concept Learning. In: Michalski, R.; Carbonell, J.; Mitchell, T. (ed.): *Machine Learning: An Artificial Intelligence Approach , Vol.II*, pages 149–166. Morgan Kaufmann, Los Altos, California.

Reiter, R. (1984). Towards a Logical Reconstruction of Relational Database Theory. In: Brodie, M.; Mylopoulos, J.; Schmidt, J. (ed.): *On Conceptual Modelling*, pages 191–233. Springer, New York et al.

van Rijsbergen, C. (1977). A Theoretical Basis for the Use of Co-Occurrence Data in Information Retrieval. *Journal of Documentation 33*, pages 106–119.

van Rijsbergen, C. (1979). *Information Retrieval.* Butterworths, London, 2. edition.

Robertson, S.; Sparck Jones, K. (1976). Relevance Weighting of Search Terms. *Journal of the American Society for Information Science 27*, pages 129–146.

Robertson, S. (1977). The Probability Ranking Principle in IR. *Journal of Documentation 33*, pages 294–304.

Robertson, S.; Van Rijsbergen, C.; Porter, M. (1981). Probabilistic Models of Indexing and Searching. In: Oddy, R.; Robertson, S.; Van Rijsbergen, C.; Williams, P. (ed.): *Information Retrieval Research*, pages 35–56. Butterworths, London.

Salton, G.; Voorhees, E. (1985). Automatic Assignment of Soft Boolean Operators. In: *Proceedings of the 8th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 54–69. ACM, New York.

Salton, G. (ed.)(1971). *The SMART Retrieval System - Experiments in Automatic Document Processing.* Prentice Hall, Englewood Cliffs, New Jersey.

Salton, G. (1987). *Private communication.*

Salton, G.; Fox, E.; Wu, H. (1983). Extended Boolean Information Retrieval. *Communications of the ACM 26*, pages 1022–1036.

Schneider, R.; Kriegel, H.-P.; Seeger, B.; Heep, S. (1989). Geometry-Based Similarity Retrieval of Rotational Parts. In: *Proceedings 2nd International Conference on Data and Knowledge Systems for Manufacturing and Engineering.*

Tietze, A. (1989). *Approximation of Discrete Probability Distributions by Dependence Trees and their Application as Indexing Functions (in German).* Diploma thesis, TH Darmstadt, FB Informatik, Datenverwaltungssysteme II.

Vassiliou, Y. (1979). Null Values in Database Management - a Denotational Semantics Approach. In: *Proceedings of the ACM SIGMOD International Conference on the Management of Data.* ACM, New York.

Westbrook, J.; Rumble, J. (ed.)(1983). *Computerized Materials Data Systems*, Gaithersburg, Maryland 20899, USA. National Bureau of Standards.

Wong, A.; Chiu, D. (1987). Synthesizing Statistical Knowledge from Incomplete Mixed-Mode Data. *IEEE Transactions on Pattern Analysis and Machine Intelligence 9(6)*, pages 796–805.

Wong, S.; Yao, Y. (1989). A Probability Distribution Model for Information Retrieval. *Information Processing and Management 25(1)*, pages 39–53.

Wong, S.; Ziarko, W.; Raghavan, V.; Wong, P. (1987). On Modeling of Information Retrieval Concepts in Vector Spaces. *ACM Transactions on Database Systems 12(2)*, pages 299–321.

Yu, C.; Salton, G. (1976). Precision Weighting. An Effective Automatic Indexing Method. *Journal of the ACM 23*, pages 76–88.

Yu, C.; Buckley, C.; Lam, K.; Salton, G. (1983). A Generalized Term Dependence Model in Information Retrieval. *Information Technology: Research and Development 2*, pages 129–154.

Zadeh, L. (1965). Fuzzy Sets. *Information and Control 8*, pages 338–353.