# Integration of Probabilistic Fact and Text Retrieval

Norbert Fuhr
Universität Dortmund, Informatik VI, W-4600 Dortmund 50
Germany

## Abstract

In this paper, a model for combining text and fact retrieval is described. A query is a set of conditions, where a single condition is either a text or fact condition. Fact conditions can be interpreted as being vague, thus leading to nonbinary weights for fact conditions with respect to database objects. For text conditions, we use descriptions of the occurrence of terms in documents instead of precomputed indexing weights, thus treating terms similar to attributes. Probabilistic indexing weights for conditions are computed by introducing the notion of correctness (or acceptability) of a condition w.r.t. an object. These indexing weights are used in retrieval for a probabilistic ranking of objects based on the retrieval-with-probabilistic-indexing (RPI) model, for which a new derivation is given here.

## 1 Introduction

In the past, research in the field of information retrieval (IR) has focused on the problem of text retrieval. It has been shown that Boolean retrieval yields a rather poor retrieval quality, and the complexity of the user interface prevents many potential users from using this kind of retrieval systems. On the other hand, systems based on the vector space model [Salton 71] or probabilistic retrieval [Rijsbergen 79] improve retrieval effectiveness by producing a ranked list of answers instead of a set only as in Boolean retrieval. In both models, a query is a set of terms (without logical operators, as in Boolean retrieval). Each term can be given a search term weight which reflects its importance with respect to the query. By means of relevance feedback, better values for these weights can be estimated, thus yielding significant improvements of retrieval quality [Yu & Salton 76]. A term also can be assigned a so-called indexing weight with respect to a document, thus discriminating terms of different importance in a document [Salton & Buckley 88] [Fuhr & Buckley 91]. Experiments have shown that this kind of weighted indexing outperforms binary indexing for most applications. Besides the benefits in terms of retrieval quality, the non-Boolean models also improve the user-friendliness of the retrieval system. Much of the complexity of the Boolean model is reduced by omitting Boolean operators and treating queries as a set of terms only. Furthermore, ranking allows a user to select any number of documents that he wants to see, while with Boolean systems, he has to transform the query formulation until the appropriate size of the output is achieved.

However, in real IR applications, the objects to be retrieved hardly ever consist of text only. For example, document databases offered by public hosts also provide bibliographic information. In office information systems, documents comprise a number of attributes besides the text itself. Queries posed to these systems frequently relate to the attribute values of the documents, e.g. the name of the author (or sender) of a letter or the publication (or mailing) date. With regard to these requirements, text retrieval methods only offer a partial solution. So concepts for combining text and fact retrieval have to be devised.

In this paper, we will present a new probabilistic approach that treats texts and facts in a similar way. First, we give a survey over recent work in this area. Then we describe a formal model for the integration of probabilistic text and fact retrieval along with the underlying probabilistic model. We present an application of our approach together with a user-friendly interface. Finally, we give an outlook on further work in this area.


15th Ann Int'l SIGIR '92/Denmark-6/92
© 1992 ACM 0-89791-524-0/92/0006/0211...$1.50

## 2 Extending text retrieval methods for coping with facts

The most simple way for combining text and fact retrieval can be found in today's commercial information retrieval systems, where Boolean retrieval is performed for facts as well as for texts. If we concentrate on approaches that apply ranking methods at least for texts, then three different methods for the combination of text and fact retrieval can be distinguished:

1. Boolean retrieval for facts,
2. non-Boolean retrieval for facts with binary weighting for fact conditions,
3. non-Boolean retrieval for facts with non-binary weighting for fact conditions,

We discuss each of these approaches in the following.
The paper [Raghavan et al. 86] describes the combination of probabilistic text retrieval methods with Boolean fact retrieval. Here the fact conditions select a set of objects from the database, followed by a ranking of objects according to the text conditions. This hybrid approach does not allow for ranking according to fact conditions.
A simple approach for applying ranking methods to fact conditions as well as to text conditions is to treat fact conditions like index terms in combination with binary indexing. This idea has been implemented in several systems described in the literature:

- In the SMART system [Buckley 85], different so-called "concept types" can be distinguished; besides text terms, for example dates or names of persons or institutions can be used for retrieval. However, for these attributes, only tests on equality can be performed; furthermore, only a binary weighting of a fact condition w.r.t. a document is possible (however, the application of the SMART indexing procedures may lead to a slightly different weighting of the same concept type value in different documents in case the document vector is normalized).

- In [Saxton & Raghavan 90], a more elaborate approach based on the generalized vector space model is presented. Here a (IR-type) query is a disjunction of conditions, where a single condition can be a Boolean combination of subconditions. For fact conditions, the standard comparison operators found in database systems can be applied. Each condition in the query is given a weight, and then objects are ranked according to the weighted sum of the conditions they are fulfilling.

- The paper [Rabitti & Savino 90] describes an advanced system for retrieval of multimedia data (including facts). This system uses probabilistic retrieval for text as well as for facts. Both kinds of conditions can be assigned weights with respect to the query. In addition, text terms can be given probabilistic index terms weights w.r.t. an object, whereas conditions relating to attributes are either true or false for a specific object.

The major drawback of the approaches described above is the rigid treatment of query conditions relating to attributes. We think that for most of the applications discussed here, conditions specified by a user should be regarded as being vague in most cases. For example, in an office information system, when a user seeks for a letter that was sent last month, then there is a certain probability that the letter may date one or two weeks earlier. With proper names, there is a similar problem, since the user may have difficulties to give the correct spelling (or the name in the document may be misspelled). For this reason, in case there is no answer that fulfills all the criteria specified in the query, the system also should search for objects that are close to match these criteria. As a result, the system should present a ranking of objects, from which the user may select the one he is looking for.
In the systems described above that provide ranking for fact conditions, vague conditions can be simulated by splitting the fact condition into several conditions with different attribute values and assigning different weights to these conditions. However, this approach can only be applied for certain attributes (e.g. by specifying ranges in numerical or date conditions); if for example the similarity of strings is to be considered, then the standard database operators are insufficient for achieving the desired ranking.
Besides the examples mentioned above which deal with relatively simple data types like dates or names, there is an increasing need for information systems that can cope with vague conditions relating to rather complex data types:

- In engineering, when a new part has to be constructed, it may be more effective to modify an existing, similar part (stored in the database) than to start from scratch [Schneider et al. 89].

- Materials data systems contain property values of a large number of materials. Most requests posed to these systems either ask for a new material with properties similar to those of a known material, or they seek for materials that are optimum with respect to a number of criteria (which may be in conflict with each other) [Ammersbach/et al:88].

- In software engineering, approaches for software reuse seek for modules similar to a given specification. In [Pintado & Tsichritzis 90], similarity measures based on the distance in the object-oriented inheritance hierarchy are discussed for this purpose.

This problem of vague queries in databases has gained more attention recently [IEEE 89]. So far, three basic approaches have been proposed:

- The VAGUE system described in [Motro 88] is based on the vector space model. For each attribute for which a vague condition is specified in the query, the user may choose between a number of different metrics for the comparison of attribute values with the corresponding value from the query. Then the distance between the query and a database object is computed as a function (which can be interpreted as a distance measure in a vector space) of the distances for the different query conditions. Finally, objects are ranked according to increasing query-object distances.

- Fuzzy databases have been proposed in [Prade & Testemale 84] and [Zemankova & Kandel 85]. In fuzzy logic, the notion of a fuzzy set is a generalization of standard set theory where an element either belongs to a set or not. For a fuzzy set, a membership function gives the degree of set membership for a specific element, which can be a real number ranging from 0 to 1. When applied to databases, the sets of objects fulfilling single (vague) conditions as well as the set of objects that are answers to a whole query may be fuzzy sets. For Boolean operators, appropriate definitions are given for combining the values of membership functions, such that fuzzy logic is a generalization of Boolean logic. So this approach also yields a ranking of objects in the form of a fuzzy set.

- Recently, a probabilistic model for vague fact retrieval has been developed [Fuhr 90]. In this paper, we describe an extension of this model for integrating text and fact retrieval. In contrast to the other approaches mentioned above, the probabilistic model can exploit empirical data from an application in order to improve the quality of the system's output.

Many applications of fact retrieval also have to cope with a second problem, namely imprecise data. Unlike business or administration applications (for which today's database management systems have been developed), technical and scientific data is frequently inclomplete or imprecise. For this problem, standard database management systems offer only poor support (e.g. null values). In the probabilistic approach [Fuhr 90], imprecise or missing attribute values can be stored as probability distributions over the set of possible attribute values. This way, we can make optimum use of empirical data (e.g. about the precision of a measuring device). Fuzzy databases also allow for the consideration of imprecise data; however, there is no obvious way for considering empirical data. We will not discuss the issue of imprecise data here any further, since the extensions required for coping with this problem (as described in [Fuhr 90]) can simply be added to the integrated approach presented in the following.

The discussion above has shown that for many applications, fact retrieval should be handled in a similar way as text retrieval. So we seek for an approach that integrates text and fact retrieval by regarding both conditions relating to text or facts as being vague. So far, the only approach for combining vague fact and text retrieval is the office information system described in [Croft & Krovetz 88]. Whereas probabilistic indexing and search term weighting is applied for the text part of the query, the theory of endorsements is used for conditions relating to facts. However, there is no obvious way how the results of the two parts of the query can be combined for producing a single ranking.

In the following, we show how a probabilistic text retrieval model can be combined with a probabilistic model for vague fact retrieval. This combination treats texts and facts in a uniform way. The new model can be applied in many areas where texts and facts are stored together, and where at least parts of the queries relating to facts should be interpreted as vague conditions (e.g. uncertainty about dates, correct spelling of names, similarity of objects like products or software modules).

# 3 A unified model for text and fact retrieval

Our unified model is an extension of the probabilistic model for fact retrieval presented in [Fuhr 90]. Since we concentrate on the retrieval aspect of databases here, we assume a fairly simple data model, where a database consists of a set of objects to which a query may relate to; we do not consider schema transforming operations here (especially join operations), but these operations can be assumed to take place before our approach is applied, that is, the database that we regard here may be derived from the original database via some standard database operations.

**Definition 1** *A database is a set of objects $\underline{O}$. A single object $\underline{o}_m \in \underline{O}$ is represented by a pair $o_m = (o_m^F, o_m^T)$, where $o_m^F$ is called the fact part and $o_m^T$ is called the text part of $\underline{o}_m$.*

As in all probabilistic models, we distinguish between an object itself and its representation (e.g. assume a document represented by the set of terms occurring within the document). In our approach, we distinguish between the text part and the fact part of objects, since these parts are treated differently in the retrieval process.

**Definition 2** *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 the fact part of an object instance $\underline{o}_m$ is a tuple $o_m^F =< o_m(a_1), \ldots, o_m(a_n) >$ with $o_m(a_i) \in D_i$.*

Although only a linear data model is assumed here, it should be emphasized that we pose no restrictions on the domains of attributes; so complex data types are allowed here, too.

**Definition 3** *Let $T$ denote the set of terms $t_i$ occurring within the text parts of the objects $\underline{o}_m \in \underline{O}$. Furthermore, let $Z$ denote the set of all possible descriptions for the occurrence of a term in a text. Then the text part of an object instance $\underline{o}_m$ is a set $o_m^T = \{(t_i, z(t_i, o_m)) | t_i \in T \wedge z(t_i, o_m) \in Z\}$ of terms and their corresponding descriptions.*

Here we make no specific assumptions about the type of terms that we are regarding; in principle, index terms can be single words, noun phrases, terms from a controlled vocabulary or even content identifiers based on some knowledge-based method. Especially for phrases, the set of all terms may not be known completely in advance. For this reason, it is not assumed here that a text is mapped directly onto a set of weighted index terms (unlike most other text indexing approaches). Instead, each term is associated with a description. The weighting process is postponed until retrieval time (at least conceptually). The details of the weighting process are described below. With this approach, terms and fact attributes are quite similar, since each term can be regarded as an attribute, and the descriptions represent the corresponding attribute values. In many applications, the text of an object will have a certain structure (e.g. title, abstract, additional keywords etc., see the example in section 6). In our probabilistic model, we do not distinguish explicitly between the different parts of a text; instead, the location information is mapped into the description of the occurrence of the term, which in turn is used for computing a weight for the term w.r.t. the text following the description-oriented indexing approach as described in [Fuhr & Buckley 91] (see below).

Now we describe the structure of the queries.

**Definition 4** *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, unless stated otherwise, we will restrict to the vague part of the query. As an example for a combined query, assume a user of an office information system seeking for all letters with offers for office furniture that were received within the last six months:

```
FIND DOCUMENT
  WHERE DOCTYPE='OFFER'
  RANK BY DATE > 6/91,
          TEXT: 'OFFICE FURNITURE'
```

Here the WHERE-clause specifies the Boolean query which is a Boolean expression in general. The vague query starts with the RANK clause and consists of a set of conditions, each one being either a text condition or a fact condition.

**Definition 5** *A vague query is a pair $q_k = (q_k^V, q_k^J)$, where $q_k^V$ denotes the set of vague conditions forming the query and $q_k^J$ is the relevance feedback data for the query. With $\mathcal{R} = \{R, \overline{R}\}$ (relevant/nonrelevant) denoting the set of possible relevance judgements for query-object pairs, the relevance feedback data is a subset $q_k^J \subset \underline{O} \times \mathcal{R}$. The (nonempty) set of vague conditions $q_k^V = q_k^F \cup q_k^T$ is the union of the set of fact conditions $q_k^F$ and the set of text conditions $q_k^T$. A text condition is a term $t_i \in T$.*

In our example query, there is one fact condition (DATE > 6/91) and one text condition (TEXT: 'OFFICE FURNITURE'). In addition to the query formulation comprised of the set of vague conditions, we also consider the set of relevance feedback judgements that we got for this query so far. This data may be exploited by the retrieval function (see below) in order to yield an improved output ranking.
For the definition of fact conditions, we first have to introduce the notion of vague predicates.

**Definition 6** *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 predicates defined and a set $F_i^2$ of binary predicates defined.*

In contrast to fuzzy logic, a predicate is just a label here and not a mapping of attribute values onto values of a membership function. Examples for unary predicates are "low", "high", "medium" or "recent" (for dates) and also so-called fuzzy quantifiers like "some", "several", "many" . Most binary vague predicates will be vague interpretations of the standard predicates like e.g. "=", "<", "≥".

**Definition 7** *A fact condition $c_i$ can have one of the three 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.*
- $(a_i, f_i, a_j)$ *with* $a_i \in A$, $a_j \in A$ *and* $f_i \in F_i^2 \cap F_j^2$

So a fact condition may be an attribute value with a unary attribute (e.g. RECEPTION_DATE RECENT), an attribute value with a binary predicate and a comparison value (e.g. PRICE < 1000), or two attributes compared by a binary attribute (e.g. RECEPTION_DATE > EXPIRING_DATE).

The goal of our probabilistic model is the estimation of the probability $P(R|q_k, o_m)$ that an object with the representation $o_m$ will be judged relevant by a user who submitted the vague query $q_k$. For the estimation of this probability, we want to allow nonbinary weights for conditions w.r.t. objects. For text conditions, this means that weighted indexing is used instead of a binary one. The same approach is to be taken for fact conditions. As described in the previous section, fact conditions are to be interpreted as vague conditions, thus allowing for the distinction between objects fullfilling a condition to different extents (e.g. when seeking for a letter dating from last week, we would like to distinguish between those sent one day earlier and others from a week or more before).

In order to integrate text and fact retrieval, we have introduced the notion of vague conditions, where a condition $v_i$ can be either a text condition $t_i$ or a fact condition $c_i$. For facts, this approach is obvious, since standard database query languages also use conditions of the form <attribute> <predicate> <attribute> or <attribute> <predicate> <value>. In text retrieval, usually a document is represented by a set of index terms with associated weights. However, for more advanced text analysis methods, this approach seems to be inadequate. For example, when phrases are regarded in addition to single words, then the number of phrases from queries that occur within a document may become very large. For this reason, a preprocessing of all possible phrases for a document may not be feasible; instead, a phrase should be regarded as a condition that has to be evaluated for the document at retrieval time. This approach is taken in [Schwarz 90]; as an example, a text condition "analysis of amino acids in cheese" would yield a partial match for the phrase "acid analysis", but not for "cheese analysis". So a text retrieval system supporting this kind of search should regard query terms as text conditions, for which the construction of the descriptions $z(t_i, o_m)$ for the occurrence of terms in documents is performed at retrieval time (at least conceptually, since we would not like to exclude the possibility of preprocessing the indexing for certain types of terms, e.g. single words). From this point of view, there is no fundamental difference between text and fact conditions, since both have to be compared with the actual values of the database objects at retrieval time.

# 4 Indexing

Now we discuss the weighting process for (text or fact) conditions w.r.t. objects. For brevity, we will call these weights indexing weights, irrespective of the type of condition. As a theoretic basis for the computation of the indexing weights, a probabilistic indexing approach is applied. For this purpose, we introduce the additional concept of correctness as a property of an object-condition pair. The term 'correctness' is used here for historical reasons. A more appropriate term would be 'acceptability': Since conditions are to be regarded as being vague, the term acceptability would express the fact whether an object is still acceptable w.r.t. a specific condition; for example, whether an item with a price of 1050 is still acceptable when the user specified the condition PRICE $\leq$ 1000. The decision about the correctness as a condition-object pair can be specified explicitly, in addition to the relevance judgement (which relates to the overall query-object relationship). However, it is also possible to derive these decisions from the relevance judgements of query-object pairs: 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. For text indexing, experiments with both definitions are described in [Fuhr 89] and [Fuhr & Buckley 91].

Let $C$ denote the event when a condition-object pair is correct, and $\bar{C}$ the opposite case. The probabilistic indexing weight that we want to estimate is defined as $P(C|v_i, o_m)$. This is the probability that a random object with representation $o_m$ will be judged correct w.r.t. condition $v_i$. For the estimation of this probability, the description-oriented indexing approach presented in [Fuhr & Buckley 91] is applied. Here we want to describe this method only briefly.

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 $v_i$ and the object (representation) $o_m$ is collected in the so-called relevance description $x(v_i, o_m)$. Based on this data, the decision step yields an estimate of the probability $P(C|x(v_i, o_m))$ that an object-condition pair

described by relevance description $x$ will be judged correct. The indexing task has to be performed separately for different attribute-predicate pairs (and additionally for text conditions). Whereas the description step depends on the type of the condition, the decision step is identical for all types.

**Definition 8** *A relevance description $x(v_i, o_m)$ is a data structure that describes properties of the condition $v_i$, the object representation $o_m$ and their relationship.*

The concept of relevance descriptions yields an abstraction from specific condition-object pairs. The actual definition of a relevance description has to be chosen heuristically. Here we want to give just some ideas of how the descriptions for different types of conditions may be chosen:

- For text conditions, the relevance description consists of the occurrence description of the term w.r.t. the object plus additional collection-related information about the term. For single words as terms, the occurrence description may contain the within-document frequency, the total number of words within the text of the object and additional location information (i.e. the parts of the text in which the term occurs) [Fuhr & Buckley 91]. For phrases, the output of text analysis methods as outlined above should be included. Descriptors from a controlled vocabulary also may be regarded here, as described in [Fuhr et al. 91], [Fuhr & Pfeifer 91]. Collection-related information is for example the inverse document frequency, but also dictionary data could be considered (e.g. when only a synonym of a query term occurs within a document).

  For dynamic collections, the subdivision of the relevance description into an occurrence description and collection-related information offers the advantage that the former is stable, while the latter may change over time. The aspect of collection dynamics is another reason for viewing text indexing weights as being computed at retrieval time.

- For fact conditions with binary predicates, similarity measures are candidates for elements of the relevance description. As an example, if $a_i$ denotes an attribute with numeric values in the condition $c_i = (a_i, f_i, d_i)$, one can define:

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

  This way, all pairs with the same relative difference between comparison value and attribute value are mapped onto identical relevance descriptions.

  For string-valued attributes, string distance measures can be applied (e.g. by counting the number of common trigrams or by including a marker indicating whether there is a phonetic match between the two strings or not).

- 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 $o_m(a_i)$.

In the decision step, estimates of the probabilities $P(C|x(v_i, o_m))$ 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 $P(C|x(v_i, o_m))$ 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 9** *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 $u_{im} = e(x(v_i, o_m))$ the indexing weight of the object $o_m$ with respect to condition $v_i$.*

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. Examples for probabilistic classification algorithms are described in [Fuhr & Buckley 91], [Fuhr et al. 91].

# 5 Retrieval

With the indexing weights computed as described in the previous section, retrieval can be performed. The retrieval function $\varrho(q_k, o_m)$ computes 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.

**Definition 10** *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 the initial ranking, the user may assign weights to the conditions of his query, in order to denote their different importance w.r.t. his need. For this purpose, let $w_{ik}$ give the weight of condition $v_i$ in query $q_k$.
Based on the indexing weights $u_{im}$ and the query condition weights $w_{ik}$, the decision-theoretic linear retrieval function [Wong & Yao 90] yields

$$\varrho_{lin}(q_k, o_m) = \sum_{v_i \in q_k^V} w_{ik} \cdot u_{im}. \quad (1)$$

If relevance feedback data is available, that is, the set $q_k^J$ is not empty, the initial ranking can be further improved by applying the RPI model (see the detailed description of the model and the precise definition of the parameters in the appendix). Let $O_k^R = \{o_j | (o_j, R) \in q_k^J\}$ denote the set of objects). With $q_i$ denoting the average indexing weight for condition $v_i$ in all objects of the database, and $p_{ik}$ as the average indexing weight for condition $v_i$ in the objects judged relevant w.r.t $q_k$ so far, the retrieval formula yields:

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

$$\prod_{v_i \in q_k^V} \left[ \left( \frac{p_{ik}(1-q_i)}{q_i(1-p_{ik})} - 1 \right) \cdot u_{im} + 1 \right] \quad (2)$$

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.

# 6 An application example

In order to illustrate the concepts outlined in the previous section, we present an application of integrated text and fact retrieval here. First, the database and the indexing functions for the different attributes are described briefly, followed by the presentation of the PFTR system developed for this application.
As database, we have used a number of documents from the NTIS database which contains references to research reports of scientific projects in the USA. An example document is shown in figure 1.
From the different categories of these documents, the title, the summary, the controlled terms (CT) and the text of the classification codes (CC) can be accessed via text retrieval. In order to compute the indexing weights for the text part, the SMART indexing procedures have been applied ([Salton & Buckley 88]) for this specific application. These procedures have the advantage that no relevance feedback information is required for indexing. However, as shown in [Fuhr & Buckley 91], when feedback data is available, a better retrieval quality can be achieved with probabilistic indexing methods.
For the other categories of the document as well as for the classification codes itself, vague fact retrieval methods are supplied. Except for the attribute 'date' (see below), string distance measures are used as relevance descriptions within the indexing functions of the different attributes. For these attributes, only the vague equality predicate is supported. The definition of the distance measure actually used depends on the attribute:

- For the attribute "Author" (AU), the proportion of common trigrams is computed.
- With the classification codes (CC), the length of the longest common prefix is regarded.
- For "Number of Report" (NR) and "Other Sources", the proportion of matching characters is computed.
- With the attribute "Corporate Source" (CS), the number of common words is counted.

For dates, the relative difference between the comparison value $d_i$ and the attribute value $o_m(a_i)$ is taken as

**AN** 87(01):411 NTIS Order Number: AD-A172 502/7/XAD
**TI** Controlling Inference. (Doctoral thesis)
**AU** Smith, David E.
**CS** Stanford Univ., CA. Dept. of Computer Science
**NC** Contract : N00014-81-K-0004
**NR** AD-A172 502/7/XAD; STAN-CS-86-1107
197 p. NTIS Prices : MF A01
Availability : Microfiche copies only.
**PD** 860400
**LA** English CY United States
**OS** GRA&I8701
**AB** Effective control of inference is a fundamental problem in Artificial Intelligence. Unguided inference leads to a combinatoral explosion of facts or subgoals for even simple domains. To overcome this problem, ...
**CC** 95F Bionics and artificial intelligence
**CT** *Artificial intelligence; Data bases; Decision making; Global; Information exchange; Problem solving; Efficiency *Inference; Control; Theses; Expert systems
**UT** NTISDODXA

Figure 1: Example NTIS document

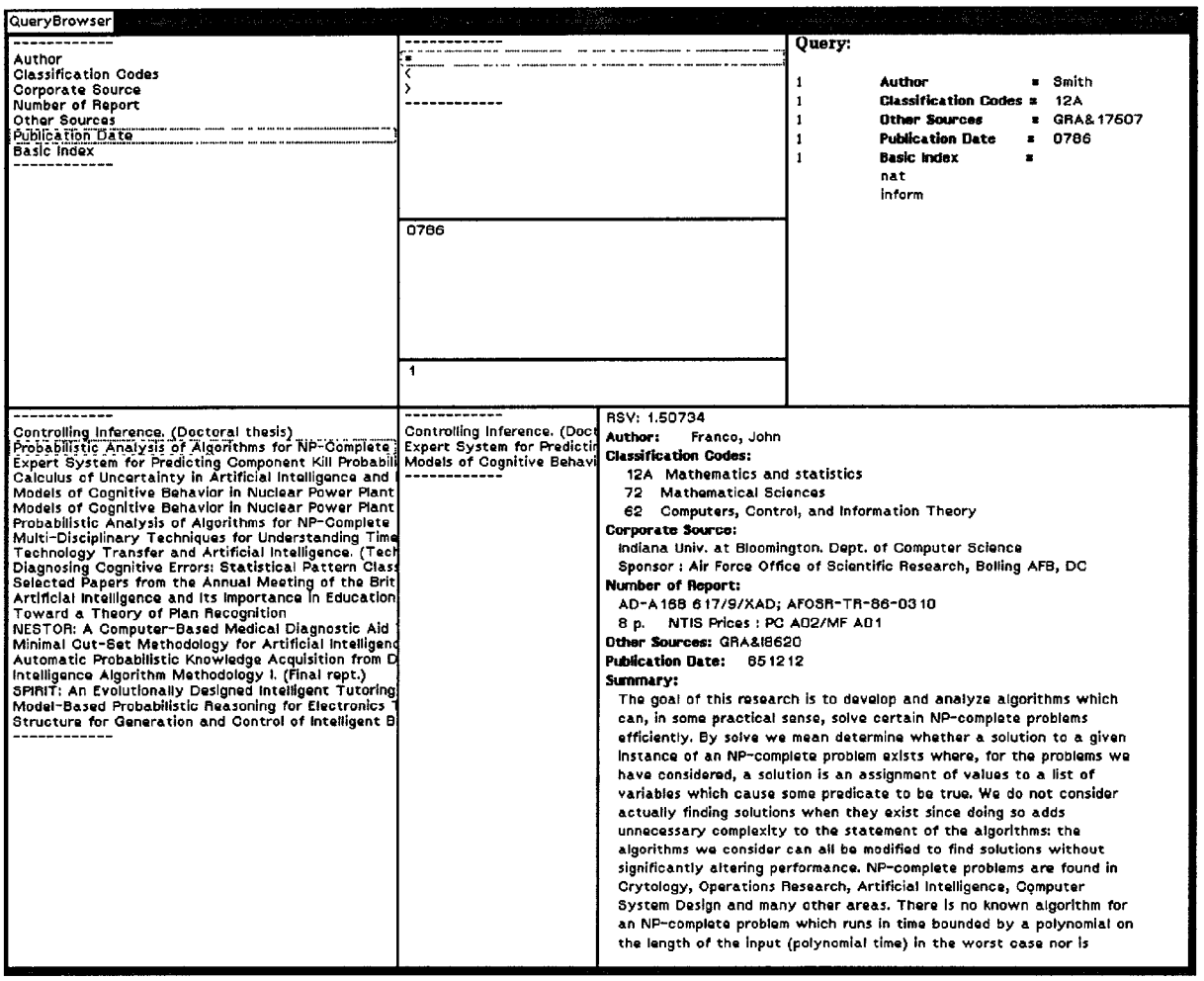


Figure 2: User interface of the PFTR system

relevance description. As predicates, vague interpretations of the predicates '=', '<' and '>' are provided.

Now we describe the prototype system PFTR for probabilistic fact and text retrieval for the application presented above. This prototype serves mainly for the illustration of the user interface and the basic retrieval functions, whereas certain information system aspects (like e.g. the management of large collections of objects) have not been considered yet. The system is implemented in Smalltalk-80 similar to a Smalltalk browser; all documents are held in main memory.

Figure 2 shows the user interface of the PFTR system. The windows in the upper row serve for the formulation of vague attributes to which queries may relate to ("Basic Index" = text conditions). After selecting an attribute with the mouse, the next window shows the predicates that are applicable for this attribute. Having chosen a predicate, the user may enter a comparison value and a query condition weight within the two small windows below the predicate window. Then the rightmost window shows the complete vague query constructed so far. With the operation "remove", single conditions can be removed from the query formulation. When the user has completed his query formulation, he issues an "update" command which tells the system that the windows in the bottom row should be updated such that they present the answers to the query formulated in the top half. In the leftmost window of the bottom row, the ranked list of items is shown by displaying the titles. By selecting one of these titles, the complete object is shown in the rightmost window. Furthermore, a user may mark some of the objects a being relevant to his query. In turn, these titles are shown in the middle window. When the user wants to see the effect of the relevance feedback information given to the system, he again issues the "update" command; in turn, the system

computes the new ranking and displays the new list in the leftmost window. The user may also decide to modify his query by adding or removing conditions. After finishing the modification of the query, the "update" command produces the new ranking.
The basic idea behind the "update" command is to provide a generic operation for performing the retrieval step. In principle, a new ranked list of objects could be computed after each modification of the query or after ranking an object as being relevant. However, an implementation of an instant re-ranking would be rather inefficient and also confusing for the user (he would have to wait for the result, and furthermore he may not want instant changes of the ranked list). For this reason, the system indicates that an update operation would be possible by displaying a grey frame around the whole PFTR browser. So the user knows that the result displayed in the bottom half does not reflect the current state of the query and/or the feedback judgements, and that an update command would produce the latest state of the search. This strategy is rather different from that of today's systems, where user commands correspond one-to-one to system operations, and where additional commands are necessary after a retrieve command in order to show elements of the answer.

# 7 Conclusions and outlook

In this paper, we have described a probabilistic approach for integrating text and fact retrieval. We have shown that the application of probabilistic models for fact retrieval also gives us new insights into the problem of text retrieval; for advanced text analysis methods or dynamic collections, the current approach of indexing a document when it is added to the database does not seem to be feasible. Instead, one should treat texts similar to facts, where the final indexing weight is computed at retrieval time. Of course, certain parts of the indexing task can be performed already at input time, such that the actions to be taken at retrieval time are minimized.
Here we have chosen a fairly simple probabilistic model underlying the retrieval function, namely an independence model applied to a set of preselected objects. However, this approach can be extended easily towards dependence models (see e.g. the description of a tree dependence version of the RPI model in [Fuhr 91]); furthermore, the formalism of Bayesian inference networks also can be applied, thus allowing for more complex query structures (e.g. arbitrary combinations of Boolean and probabilistic operators) as described in [Turtle & Croft 90].
An important problem in the further development of integrated text and fact retrieval systems is the design of a system that can handle large collections of objects. (In the PFTR system, all objects are held in main memory, and the retrieval status values of all objects are computed for a query.) In a large information system, only a certain proportion of all objects should be considered w.r.t. a query, namely those that will form the head of the ranked list. For this purpose, special access paths and the corresponding ranking algorithms have to be developed.

# A Derivation of the RPI model

Here we give a new derivation of the RPI model; in comparison to the original description in [Fuhr 89], the following dertivation is an improvement w.r.t. two aspects:

- The problem of incompatible independence assumptions (see [Cooper 91]) is overcome.
- We consider the fact that only a portion of the objects in the database (namely the set of preselected objects) is to be ranked for a query.

For modelling probabilistic indexing, we assume a fixed number of binary indexings for all objects. Let $I$ denote the set of indexers, then the event space of the model is $\underline{Q} \times \underline{O} \times I$. A single element of this event space is a query-object pair with a binary indexing. Furthermore, a number of these pairs has binary relevance judgements associated with it (namely those that belong to the sets of preselected objects). In order to distinguish the correctness of different conditions, we will denote the condition a correctness decision relates to by a subscript, that is, $C_i$ stands for the event that $v_i$ is judged to be correct, and $\bar{C}_i$ denotes the contrary case.
For estimating the probability of relevance of an object, we regard the query w.r.t. the relevance descriptions $\vec{x}$ of the object (objects with the same relevance descriptions will be given the same relevance status value). First, we apply Bayes' theorem:

$$O(R|q_k,\vec{x}) = O(R|q_k)\frac{P(\vec{x}|R,q_k)}{P(\vec{x}|\bar{R},q_k)}$$

Here $O(R|q_k)$ denotes the odds that an arbitrary object considered for $q_k$ (from the set of preselected objects) will have the judgement "relevant". $P(\vec{x}|R,q_k)$ is the probability that from the set of relevant objects for $q_k$, a randomly selected element will have the relevance descriptions $\vec{x}$, and $P(\vec{x}|\bar{R},q_k)$ is the corresponding probability for the nonrelevant objects.
Next, the linked dependence assumption [Cooper 91]

$$\frac{P(\vec{x}|R,q_k)}{P(\vec{x}|\bar{R},q_k)} = \prod_{v_i \in q_k^V} \frac{P(x_i|R,q_k)}{P(x_i|\bar{R},q_k)}$$

yields

$$O(R|q_k,\vec{x}) \;=\; O(R|q_k)\prod_{v_i\in q_k^V}\frac{P(x_i|R,q_k)}{P(x_i|\bar{R},q_k)}$$

$P(x_i|R,q_k)$ is the probability that a random relevant document of $q_k$ will have the relevance description $x_i$ for condition $v_i$, and $P(x_i|\bar{R},q_k)$ is the corresponding probability for nonrelevance. Now, two more assumptions are made:

- The relevance description $x_i$ of a condition $v_i$ depends only on the correctness of $v_i$, independent of the correctness of other conditions and independent of the relevance of the object.
- The correctness $C_i$ of a condition $v_i$ (w.r.t. an object) depends only on relevance, and it is independent of the correctness of other conditions.

With these assumptions, we get:

$$O(R|q_k,\vec{x}) = O(R|q_k)\cdot$$

$$\prod_{v_i\in q_k^V}\frac{P(x_i|C_i)\cdot P(C_i|R,q_k)+P(x_i|\bar{C}_i)\cdot P(\bar{C}_i|R,q_k)}{P(x_i|C_i)\cdot P(C_i|\bar{R},q_k)+P(x_i|\bar{C}_i)\cdot P(\bar{C}_i|\bar{R},q_k)}$$

So far, the derivation is similar to that of the 2-Poisson independence model described in [Robertson et al. 81]. Now Bayes' theorem is applied once more:

$$O(R|q_k,\vec{x}) = O(R|q_k)\cdot$$

$$\prod_{v_i\in q_k^V}\frac{\frac{P(C_i|x_i)}{P(C_i)}\cdot P(C_i|R,q_k)+\frac{P(\bar{C}_i|x_i)}{P(\bar{C}_i)}\cdot P(\bar{C}_i|R,q_k)}{\frac{P(C_i|x_i)}{P(C_i)}\cdot P(C_i|\bar{R},q_k)+\frac{P(\bar{C}_i|x_i)}{P(\bar{C}_i)}\cdot P(\bar{C}_i|\bar{R},q_k)}$$

Next, we introduce the following notations for the probabilistic parameters:

$u_{im} = P(C_i|x_i)$ denotes the probabilistic indexing weight of $o_m$ w.r.t. $v_i$, the probability that a randomly selected indexer judged $o_m$ to be correct w.r.t. $v_i$.

$q_i = P(C_i)$ is the probability that a random indexer judged $v_i$ to be correct for a random object.

$p_{ik} = P(C_i|R,q_k)$ is the probability that a random indexer judged $v_i$ to be correct for a random object relevant to $q_k$.

$r_{ik} = P(C_i|\bar{R},q_k)$ is the probability that a random indexer judged $v_i$ to be correct for a random object nonrelevant to $q_k$.

With these notations, we get

$$O(R|q_k,o_m) = O(R|q_k)\cdot$$

$$\prod_{v_i\in q_k^V}\frac{\frac{u_{im}}{q_i}p_{ik}+\frac{1-u_{im}}{1-q_i}(1-p_{ik})}{\frac{u_{im}}{q_i}r_{ik}+\frac{1-u_{im}}{1-q_i}(1-r_{ik})} \qquad (3)$$

If we assume that $r_{ik}\approx q_i$, that is, $P(C_i|\bar{R},q_k)\approx P(C_i)$, we get the original RPI formula:

$$O(R|q_k,o_m)\approx O(R|q_k)\cdot$$

$$\prod_{v_i\in q_k^V}\frac{u_{im}}{q_i}p_{ik}+\frac{1-u_{im}}{1-q_i}(1-p_{ik}) \qquad (4)$$

In our model, the parameters $u_{im}$ and $q_i$ relate to the whole database, whereas the query-specific parameters $p_{ik}$ and $r_{ik}$ are only valid for the set of preselected objects. So the probabilistic indexing weights are independent of specific queries (and their Boolean parts), and they can be estimated for the whole database by collecting relevance judgements from different queries. The question whether the assumption $r_{ik}\approx q_i$ underlying the simplified RPI formula holds or not may depend on the Boolean query part; if this assumption seems to be too strong, the full formula (3) should be used instead.

The parameters from above can be estimated in the following way: Let $O_k^R = \{\underline{o}_j|(\underline{o}_j,R)\in q_k^J\}$ denote set of objects judged relevant w.r.t. $q_k$, and $O_k^N = \{\underline{o}_j|(\underline{o}_j,\bar{R})\in q_k^J\}$denote the corresponding set of nonrelevant objects; then the following estimation formulas can be used:

$$q_i \;\approx\; \frac{1}{|\underline{O}|}\sum_{\underline{o}_j\epsilon\underline{O}}u_{ij},$$

$$p_{ik} \;\approx\; \frac{1}{|\underline{O}_k^R|}\sum_{\underline{o}_j\epsilon\underline{O}_k^R}u_{ij},$$

$$r_{ik} \;\approx\; \frac{1}{|\underline{O}_k^N|}\sum_{\underline{o}_j\epsilon\underline{O}_k^N}u_{ij}.$$

# B References


**Buckley, C.** (1985). *Implementation of the SMART Information Retrieval System.* Technical Report 85-686, Department of Computer Science, Cornell University, Ithaca, NY.

**Cooper, W.** (1991). Some Inconsistencies and Misnomers in Probabilistic IR. In: Bookstein, A.; Chiaramella, Y.; Salton, G.; Raghavan, V. (eds.):

*Proceedings of the Fourteenth Annual International ACM/SIGIR Conference on Research and Development in Information Retrieval*, pages 57–61. ACM, New York.

**Croft, W.; Krovetz, R.** (1988). Interactive Retrieval of Office Documents. In: *Proceedings of the Conference on Office Information Systems*. ACM, New York.

**Fuhr, N.; Buckley, C.** (1991). A Probabilistic Learning Approach for Document Indexing. *ACM Transactions on Information Systems 9(2)*.

**Fuhr, N.; Pfeifer, U.** (1991). Combining Model-Oriented and Description-Oriented Approaches for Probabilistic Indexing. In: Bookstein, A.; Chiaramella, Y.; Salton, G.; Raghavan, V. (eds.): *Proceedings of the Fourteenth Annual International ACM/SIGIR Conference on Research and Development in Information Retrieval*, pages 46–56. ACM, New York.

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

**Fuhr, N.** (1990). A Probabilistic Framework for Vague Queries and Imprecise Information in Databases. In: McLeod, D.; Sacks-Davis, R.; Schek, H. (eds.): *Proceedings of the 16th International Conference on Very Large Databases*, pages 696–707. Morgan Kaufman, Los Altos, Cal.

**Fuhr, N.** (1991). *Probabilistic Retrieval for Imprecise Queries*. Report DV II 91-3, TH Darmstadt, FB Informatik, Datenverwaltungssysteme II.

**Fuhr, N.; Hartmann, S.; Knorz, G.; Lustig, G.; Schwantner, M.; Tzeras, K.** (1991). AIR/X - a Rule-Based Multistage Indexing System for Large Subject Fields. In: *Proceedings of the RIAO'91, Barcelona, Spain, April 2-5, 1991*, pages 606–623.

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

**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.

**Pintado, X.; Tsichritzis, D.** (1990). SaTellite: A Navigation Tool for Hypermedia. In: *Proceedings of the Conference on Office Information Systems*. ACM, New York.

**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.

**Rabitti, F.; Savino, P.** (1990). Retrieval of Multimedia Documents by Imprecise Query Specification. In: Bancilhon, F.; Thanos, C.; Tsichritzis, D. (eds.): *Advances in Database Technology - EDBT '90*, pages 203–218. Springer, Berlin et al.

**Raghavan, V.; Saxton, L.; Wong, S.; Ting, S.** (1986). A Unified Architecture for the Integration of Data Base Management and Information Retrieval Systems. In: Kugler, H.-J. (ed.): *Information Processing 86*, pages 1049–1054. Elsevier, Amsterdam.

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

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

**Salton, G.; Buckley, C.** (1988). Term Weighting Approaches in Automatic Text Retrieval. *Information Processing and Management 24(5)*, pages 513–523.

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

**Saxton, L.; Raghavan, V.** (1990). Design of an Integrated Information Retrieval / Database Management System. *2(2)*, pages 210–219.

**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*.

**Schwarz, C.** (1990). Automatic Syntactic Analysis of Free Text. *Journal of the American Society for Information Science 41(6)*, pages 408–417.

**Turtle, H.; Croft, W. B.** (1990). Inference Networks for Document Retrieval. In: Vidick, J.-L. (ed.): *Proceedings of the 13th International Conference on Research and Development in Information Retrieval*, pages 1–24. ACM, New York.

**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.** (1990). Query Formulation in Linear Retrieval Models. *Journal of the American Society for Information Science 41(5)*, pages 334–341.

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

**Zemankova, M.; Kandel, A.** (1985). Implementing Imprecision in Information Systems. *Information Sciences 37*, pages 107–141.