PIC Matrices: A Computationally Tractable Class of Probabilistic Query Operators

WARREN R. GREIFF and W. BRUCE CROFT
University of Massachusetts, Amherst

and
HOWARD TURTLE
West Publishing Co.

The inference network model of information retrieval allows for a probabilistic interpretation of query operators. In particular, Boolean query operators are conveniently modeled as link matrices of the Bayesian network. Prior work has shown, however, that these operators do not perform as well as the the *pnorm* operators used to model query operators in the context of the vector space model. This motivates the search for alternative probabilistic formulations for these operators. The design of such alternatives must contend with the issue of computational tractability, since the evaluation of an arbitrary operator requires exponential time. We define a flexible class of link matrices that are natural candidates for the implementation of query operators and an $O(n^2)$ algorithm for the computation of probabilities involving link matrices of this class. We present experimental results indicating that Boolean operators implemented in terms of link matrices from this class perform as well as *pnorm* operators in the context of the INQUERY inference network.

Categories and Subject Descriptors: H.3.3 [**Information Storage and Retrieval**]: query formulation

General Terms: Algorithms; Performance; Theory

Additional Key Words and Phrases: Bayesian networks, Boolean queries, computational complexity, inference networks, link matrices, piecewise linear functions, pnorm, probabilistic information retrieval, query operators.

This material is based on work supported in part by the National Science Foundation, Library of Congress and Department of Commerce under cooperative agreement number EEC-9209623. This material is also based on work supported in part by United States Patent and Trademark Office and Defense Advanced Research Projects Agency/ITO under ARPA order number D468, issued by ESC/AXS contract number F19628-95-C-0235. Any opinions, findings and conclusions or recommendations expressed in this material are the authors and do not necessarily reflect those of the sponsors.
Author's addresses: W. R. Greiff and W. B. Croft, Computer Science Department, University of Massachusetts, Amherst, Massachusetts, [greiff,croft]@cs.umass.edu; Howard Turtle, West Publishing Co., Eagan, Minnesota, turtle@research.westlaw.com

1. INTRODUCTION

In his doctoral dissertation [Tur90], Howard Turtle developed a probabilistic model for information retrieval formulated in terms of a Bayesian Network. The inference network is a general framework which makes possible the consideration of multiple sources of evidence in the process of ranking documents in response to a user's information need. Evidence due to multiple document representations (e.g. titles, abstracts, document bodies, manually produced indices); multiple query formulations (e.g. Boolean, natural language) and even multiple belief systems can be combined in a principled way. An attractive aspect of the inference network approach is that it provides a direct, natural, computationally efficient, probabilistically motivated method for modeling query operators.

In this paper, we present a class of query operators which, for reasons to be explained, we have called PIC operators. The PIC operators comprise a subclass of the general class of query operators that are expressible within the inference network framework. The subclass has been chosen to satisfy two important design criteria: 1) the operators belonging to the subclass are computationally tractable, and 2) they are intuitively plausible candidates for the modeling of query operators.

A wide variety of query operators can be defined as PIC operators. The work reported has been principally motivated by the search for more effective Boolean operators. Other approaches to the modeling of Boolean query operators have been tried. As with the inference network, the goal has been to generalize the classical, strict Propositional Logic interpretation of the query operators. The objective of this generalization is twofold. On the one hand, to allow for graduated inputs to the Boolean operators so that the representation of documents in terms of vectors of Boolean characteristics can be extended to vectors of feature weights. Second, to generate real valued operator output so that dichotomous relevance judgments can be replaced by document ranking in response to user's queries.

A particularly notable success in this pursuit was reported by Salton, Fox and Wu [SFW83; SBF83]. Grounded in the geometric metaphor of the vector space model, they defined a general class of "*pnorm*" operators which extend the traditional operators in a natural way. The experimental results achieved were quite positive. Recent experimentation has shown that system performance is improved by replacing the inference network Boolean operator calculation used in INQUERY by the *pnorm* AND and OR operator computations. Despite these results, the *pnorm* operator computation has not been incorporated in the INQUERY system for want of a convincing probabilistic justification.

This situation has prompted the search for an alternative to the current scheme for modeling operators within the inference network framework that might improve system performance. A major consideration in this work is the need for the computations associated with the model adopted to respect the computational limitations imposed by modern large scale information retrieval systems. As we shall see, this can be problematic within the confines of the inference network framework.

We begin with a brief review of how query operators are defined in the inference network framework and of the use of the *pnorm* calculation to model Boolean operators. This is followed by the definition of PIC operators and the presentation of an efficient algorithm for the evaluation of PIC operators for the scoring of structured queries. We then discuss the results of experiments that indicate that the modeling of Boolean queries with PIC operators can be as effective as the use of the *pnorm* operators.

2. INFERENCE NETWORK

The INQUERY inference network is a Bayesian Network [KP83; Pea88; Cha91] designed for supporting information retrieval. The nodes of the network correspond to propositions and are divided into two parts: the *document sub-network*, and the *query sub-network*. In the document sub-network, the propositions associated with the nodes pertain to the observation of: documents; representations of the documents; and representations of document content. Nodes of the query sub-network correspond to propositions regarding: the presence of query concepts; the satisfaction of queries; and the satisfaction of information needs. The subject of this paper, the modeling of Boolean queries, is directly concerned with only the *concept* and *query* nodes, and so our description will focus on this part of the network. The reader is referred to [TC90] for a more general description of the inference network framework; [CCH92] discusses the implementation of the inference network in the INQUERY retrieval system.

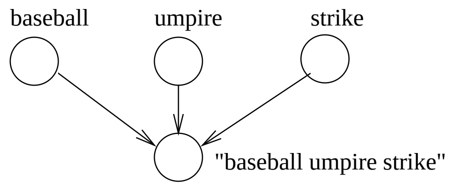

Fig. 1. a query node dependent on three concept nodes

Within the inference network formulation, a concept node is associated with the presence of a "concept" in the document currently being analyzed. For unstructured queries, there is exactly one query node in the network. This query node is connected to the concept nodes corresponding to the terms of the query. For example, figure 1 shows the relevant part of the network for the query "`baseball umpire strike`". The leftmost node corresponds to a proposition asserting the presence in the document of the concept associated with the word `baseball`. Similarly, there are nodes for the `umpire` and `strike` concepts. The query node is associated with the proposition that the user's query is satisfied.

2.1 Bayes Nets

In general, a Bayesian Network encodes a joint probability distribution. The nodes of the network correspond to random variables. In the INQUERY inference network, each random variable may assume a value of either true or false. The topology of the network is interpreted as encoding a set of conditional independence relations among the variables. If the nodes corresponding to the variables, $P_1, \ldots, P_n$, are the immediate predecessors (*parents*) of a node, Q, and $Z_1, ..., Z_s$ are all other nodes that are not *descendents* of (i.e. are not reachable from) Q, as shown in figure 2,

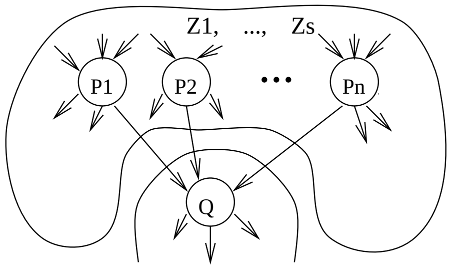

Fig. 2. conditional independence encoded in a Bayesian Network

then Q is considered to be conditionally independent of $Z_1, ..., Z_s$ given $P_1, \ldots, P_n$:

$$pr(Q \mid P_1, \ldots, P_n, Z_1, \ldots, Z_s) = pr(Q \mid P_1, \ldots, P_n)$$

Given probabilities for the root nodes (i.e. nodes with no parents), the network may be processed in a top-down fashion in order to produce the probabilities relevant to each of its nodes. As a consequence of the conditional independence assumptions implicit in the topology of a Bayesian Network, once the probabilities, $p_1, \ldots, p_n$, have been produced for the parents of a node, Q, the probability that Q assumes the value $y \in D_q$ is given by:

$$pr(Q = y) = \sum_{x_1 \in D_1, \ldots, x_n \in D_n} pr(Q = y \mid P_1 = x_1, \ldots, P_n = x_n) pr(P_1 = x_1) \cdots pr(P_n = x_n)$$

where $D_1, \ldots, D_n$, D_q are the sets of values that may be assumed by the variables, $P_1, \ldots, P_n$, and Q, respectively. For the INQUERY inference network, $D_1 = D_2 = \ldots D_n = D_q = \{\text{true,false}\}$.

2.2 Binary Valued Random Variables

In INQUERY each node corresponds to a proposition; that is, a variable that may take on one of two values: *true* or *false*. For example, in figure 2, each P_i might correspond to the proposition that some document under consideration is about some concept, c_i, while Q corresponds to the proposition that the query is satisfied.

Since all the variables are binary valued in INQUERY, the dependence of a child on its parents can be given via the specification of:

$$\begin{array}{ll} pr(Q \text{ is true} \mid P_1 = b_1, \ldots, P_n = b_n) & \quad \text{and} \\ pr(Q \text{ is false} \mid P_1 = b_1, \ldots, P_n = b_n) & \end{array}$$

for each:

$$< b_1, \ldots, b_n > \in \{\text{true}, \text{false}\}^n$$

Equivalently, the values:

$$\begin{array}{rcll} \bar{\alpha}_R & = & pr(Q \text{ is false} \mid & i \in R \Rightarrow P_i \text{ is true} \\ & & & i \notin R \Rightarrow P_i \text{ is false}) \quad \text{and} \\ \alpha_R & = & pr(Q \text{ is true} \mid & i \in R \Rightarrow P_i \text{ is true} \\ & & & i \notin R \Rightarrow P_i \text{ is false}) \end{array}$$

must be specified for every possible subset, R, of $\{1, \ldots, n\}$. In terms of these conditional properties, the probability that a child node is true can be calculated once the probabilities of truth, $p_1, \ldots, p_n$, are known for each of its parent nodes:

$$pr(Q \text{ is false}) = \sum_{R \subseteq \{1,\ldots,n\}} \bar{\alpha}_R \prod_{i \in R} p_i \prod_{i \notin R} (1 - p_i) \qquad \text{and}$$

$$pr(Q \text{ is true}) = \sum_{R \subseteq \{1,\ldots,n\}} \alpha_R \prod_{i \in R} p_i \prod_{i \notin R} (1 - p_i)$$

The set of coefficients involved is conveniently organized as a 2×2^n matrix:

	$P_{0\ldots000}$	$P_{0\ldots001}$	$P_{0\ldots010}$	$\cdots$	$P_{1\ldots111}$
Q false	$\bar{\alpha}_{0\ldots000}$	$\bar{\alpha}_{0\ldots001}$	$\bar{\alpha}_{0\ldots010}$	$\cdots$	$\bar{\alpha}_{1\ldots111}$
Q true	$\alpha_{0\ldots000}$	$\alpha_{0\ldots001}$	$\alpha_{0\ldots010}$	$\cdots$	$\alpha_{1\ldots111}$

where $\alpha_{b_1,b_2,\ldots b_n}$ is the probability that Q is true subject to the condition that the parents P_i such that $b_i = 1$ are true, and the parents P_i such that $b_i = 0$ are false; $\bar{\alpha}_{b_1,b_2,\ldots b_n}$ is the corresponding probability that Q is false and is equal to $1 - \alpha_{b_1,b_2,\ldots b_n}$. This matrix, known as a *link matrix*, may be visualized as linking the child node with the parent nodes as is shown in figure 3

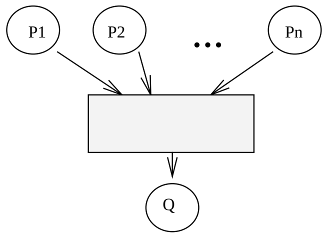

Fig. 3. *link matrix* links child to parents

2.3 Link Matrices As Query Operators

INQUERY examines documents one by one. For each, the inference network is used to evaluate evidence that the document satisfies an information need expressed by the user. A given link matrix form can be viewed as defining an operator for combining evidence. For example, suppose the propositions P_1, P_2, P_3, state that three queries q_1, q_2, q_3, respectively, have been (in some sense) *satisfied* by the document currently under scrutiny. A link matrix connecting the parents nodes, P_1, P_2, P_3, with the child node, Q, can be viewed as a way of forming a query, q, that is a composite of the individual sub-queries. The child node, Q, would correspond to the proposition that the combined query, q, has been satisfied. The specification of the coefficients of the link matrix defines the way the sub-queries are *combined* in that it gives all the information necessary for determining the probability that q has been satisfied given the probabilities, p_1, p_2, p_3, that the individual subqueries have been satisfied.

In the example of figure 1, this means that once the three probabilities, p_b, p_u, p_s that the document under scrutiny is about `baseball`, `umpire` and `strike` respectively, have been determined, the probability that the document is relevant to the query can be calculated. The computation requires that the requisite conditional probabilities have been specified. One such probability, for example, is $pr(Q \mid \bar{B}U\bar{S})$, the probability that the query is satisfied, given that the document is about `umpire`, but not about either `baseball` or `strike`. There are 8 possible truth assignments for the set of three variables associated with the concept nodes of this example. The probability that the query is satisfied is then given by:

$$\begin{aligned} pr(Q) = & \quad pr(Q \mid \bar{B}\bar{U}\bar{S}) \cdot (1-p_b) \cdot (1-p_u) \cdot (1-p_s) \\ & + pr(Q \mid \bar{B}\bar{U}S) \cdot (1-p_b) \cdot (1-p_u) \cdot p_s \\ & + pr(Q \mid \bar{B}U\bar{S}) \cdot (1-p_b) \cdot p_u \cdot (1-p_s) \\ & \quad \vdots \\ & + pr(Q \mid BUS) \cdot p_b \cdot p_u \cdot p_s \end{aligned} \tag{1}$$

One, admittedly arbitrary, way of defining a 3-ary query composition operator for forming the query, q, from the sub-queries q_1, q_2, q_3, might be to specify that q is to be considered satisfied with:

—80% probability if q_1 is satisfied, independent of the whether or not q_2 and q_3 are satisfied;
—50% probability if q_1 is not satisfied, but q_2 and q_3 are both satisfied
—10% probability in any other situation.

This particular operator corresponds to the link matrix[1]:

P_{000}	P_{001}	P_{010}	P_{011}	P_{100}	P_{101}	P_{110}	P_{111}
.1	.1	.1	.5	.8	.8	.8	.8

Where the column under P_{011}, for example, gives the probability that Q is true given that P_1 is false, P_2 is true, and P_3 is true. Clearly, this link matrix has the desired effect. Typically, none of the parents will be known to be either true or false with certainty. Rather, evidence corresponding to each of P_1, P_2, P_3 will, in general, be estimated to be present with certain probabilities: p_1, p_2, p_3. Given these probabilities, the probability that Q is true can be calculated as:

$$\begin{aligned} pr(Q \text{ is true}) = & \quad .1 \cdot \bar{p}_1\bar{p}_2\bar{p}_3 + .1 \cdot \bar{p}_1\bar{p}_2p_3 + .1 \cdot \bar{p}_1p_2\bar{p}_3 + .5 \cdot \bar{p}_1p_2p_3 \\ & + .8 \cdot p_1\bar{p}_2\bar{p}_3 + .8 \cdot p_1\bar{p}_2p_3 + .8 \cdot p_1p_2\bar{p}_3 + .8 \cdot p_1p_2p_3 \end{aligned}$$

2.4 Boolean Queries

In [Tur90], Turtle describes how the inference network can be used to model Boolean queries. The section of network shown in figure 4, for example, could be constructed for the query:

```
baseball and (player or umpire) and strike
```

The link matrix for a node labeled AND would contain a one in the column corresponding to all parent nodes being true. All other columns would be zeros. This

[1]since each column must sum to 1.0, only the row corresponding to $Q =$ true will be shown from this point on.

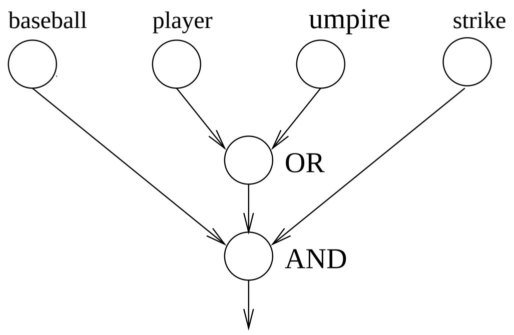

Fig. 4. a query node dependent on three concept nodes

is equivalent to saying that it is certain that an AND query is satisfied if all of its parents are (certain to be) true, and it is certain not to be satisfied when one or more of the parents is (certain to be) false. A matrix for a 3-ary #AND operator would therefore be:

L_{and}

P_{000}	P_{001}	P_{010}	P_{011}	P_{100}	P_{101}	P_{110}	P_{111}
0	0	0	0	0	0	0	1

An arbitrary member of the AND family (binary AND, 3-ary AND, 4-ary AND, etc.) would have 1.0 in the column corresponding to all parents being true, indicating that we are certain that the AND query is satisfied if all of the parents are true. All other columns would be 0.0, indicating that we are certain that the AND query is not satisfied if any one of the parents is false.

Analogously, an OR operator would have 1.0 in all cells except for a 0.0 for all parents false; the 3-arity OR being given by:

L_{or}

P_{000}	P_{001}	P_{010}	P_{011}	P_{100}	P_{101}	P_{110}	P_{111}
0	1	1	1	1	1	1	1

These link matrices are natural interpretations for the Boolean operators; and they can be evaluated very efficiently. They may be questioned on two counts, however.

First, the interpretation given to the AND and OR operators, while natural, might be considered overly strict. We ignore, for the moment, the questionable practice of ascribing certainty, in the form of 0.0 and 1.0 probabilities, to any belief under any circumstances. That excepted, we might still find it somewhat unsettling that an AND with ten parents makes no distinction between an event in which nine of these ten are true and another in which only eight of the ten are true. In fact no distinction is made between nine true parents and no true parents! Faced with a user need expressed in terms of AND, it is likely that most people's belief system would be inclined toward assigning greater, albeit perhaps only slightly greater, probability to the query being satisfied for those events for which a greater number of parents is true. An analogous argument can clearly be made for the case of the OR operator.

With respect to the issue of the strict interpretation of Boolean queries, a subtle distinction merits discussion. Although the interpretation discussed above can be said to be strict, it is nonetheless a generalization of the interpretation traditionally applied to Boolean queries. Under the conventional interpretation, the input to a Boolean operator is either 0 or 1. Were this the case in the inference network, the operators discussed above would indeed be equivalent to this strict interpretation. In the inference network, however, inputs to the Boolean operators are probabilities of truth, which may assume arbitrary values in the $[0.0, 1.0]$ interval, rather than dichotomous conditions which are either strictly true or false. The Boolean operators produce probabilities of truth which also range over the entire $[0.0, 1.0]$ interval. The AND operator, for example, will not produce a hard $0/1$ value, unless one of the inputs is exactly 0.0, or all of the inputs are exactly 1.0. If all of the inputs to the operator are probabilities in the open interval $(0.0, 1.0)$, the output will also lie in this open interval.

These observations do not obviate the comments made above with respect to the strictness of the operators as they are currently implemented, however. The computations performed for inputs $p_1, \ldots, p_n$, though none are exactly 0.0 nor 1.0, are logical consequences of the specification of the probabilities conditioned on certain knowledge of the values of the parent nodes. There is motivation, therefore, to consider the possibility of an interpretation for the AND operators that is not strict in the sense that it differentiates between all parent nodes being false and only some of the parent nodes being false, giving some extra credence to the probability of the query when a greater number of parents is true. Similarly, an OR operator may be considered that is to some, perhaps minor, degree sensitive to the number of parents that are true. It must be borne in mind that changes to the specification of the link matrices will affect the result of computations for all combinations of input values.

The second cause for concern with regard to the operators, as they are currently defined, is that experiments have shown that they are not as effective as might be hoped for. Experiments performed by Dr. Larkey have shown that the *pnorm* operators yield retrieval performance superior to that of the AND and OR operators as they are currently implemented [Lar96].

3. PNORM

A significant advance in the processing of Boolean queries was presented by Salton *et al.* in [SBF83] and [SFW83] Working within the context of the vector space model, they were able to extend conventional Boolean retrieval in a way that allowed for: 1) the weighting of document features, and 2) the production of ranked output.

Salton *et al.* normalized weights so that all vector components were in the range of 0.0 to 1.0 and, hence, all vectors lie in the unit hypercube. Intuitions based on a geometric view of information retrieval led them to consider the output of an operator as a measure of *distance* from the point $< 0.0, 0.0, \ldots, 0.0 >$. So, for the query

$$\texttt{\#or}(t_1, \ldots, t_n)$$

a document with term weights $w_1, \ldots, w_n$ for terms $t_1, \ldots, t_n$, would receive a score of,

$$(\frac{w_1^2 + \ldots + w_n^2}{n})^{\frac{1}{2}} \tag{2}$$

This operator can be considered or-like. When all weights are 0.0, it will produce a value of 0.0. When all weights are 1.0, it will produce a value of 1.0. When a variety of weights are presented as inputs, the output is more heavily influenced by the weights at the higher end of the scale. The sum produced, prior to the application of the square root, can be viewed as a weighted average of the input values, with each value receiving a weight equal to its own value:

$$(\frac{w_1 \cdot w_1 + \ldots + w_n \cdot w_n}{n})^{\frac{1}{2}}$$

Hence, higher values can be viewed as receiving greater weights. The result is that lower values can be *overpowered*, to a degree, by a single large value. This is reasonable behavior for a generalization of an OR operator. We may presume that, on specifying an OR, the user is indicating that lack of evidence with respect to most of the terms can be downplayed in the presence of strong evidence with regard to just one of the terms.

The beauty of this approach to extending Boolean operators is that the distance measure shown in eq. 2 is that it can be viewed as a special case of the more general vector norm measure[Ort72] [2]:

$$(\frac{w_1^p + \ldots + w_n^p}{n})^{\frac{1}{p}}$$

The Euclidean distance measure is, then, the vector norm measure with $p = 2$, but other values of p can be considered. For smaller values of p, the operator is less or-like. At $p = 1$, the operator degenerates to a simple average, and the Boolean structure of the query is essentially ignored. For larger values of p, the calculation can still be understood as effecting a type of weighted averaging. As p grows, more and more weight is given to the larger input components. We can view each input w_i as being weighted by w_i^{p-1} relative to the others.

$$(\frac{w_1^{p-1} \cdot w_1 + \ldots + w_n^{p-1} \cdot w_n}{n})^{\frac{1}{p}}$$

For very large values of p, the contribution of all but the largest component value are negligible. In the limit, the calculation is equivalent to that of the MAX operator, which is the standard operator used in models of information retrieval based on fuzzy-set theory [NKM77; WK79; Boo80; Boo81; BK81] and corresponds to a strict Boolean OR when the input components are limited to strict Boolean values.

In the *pnorm* model, the AND operator is defined as the *distance* of the input vector to the point $< 1.0, 1.0, ..., 1.0 >$, which is given by:

$$1 - (\frac{(1 - w_1)^p + \ldots + (1 - w_n)^p}{n})^{\frac{1}{p}}$$

[2]It should be said that the *pnorm* formula is more general than the one shown in that it allows for the weighting of query terms as well as document terms. As this aspect of the formula is not relevant to the work discussed here, we focus our attention on a somewhat simplified version of the formula.

For concreteness, we have concentrated on the OR operator, but the AND operator can be seen as the dual of OR, and all aspects of the above discussion have their counterpart with respect to AND as well.

[SFW83] reports experimental results in which use of the *pnorm* model consistently improves performance over basic Boolean retrieval on four relatively small collections. Further experimentation comparing *pnorm* with fuzzy logic models is discussed by Fox *et al.* in [FBK92]. More recently, Joon Ho Lee has run experiments on the larger TREC (Text REtreival Conference) sub-collection, the Wall Street Journal Disk 2 [Lee95]. He shows that *pnorm* performs favorably compared to a variety of other formalisms. A series of experiments at Virginia Tech combining unstructured queries with boolean formulations interpreted using *pnorm* are reported in [FS94; SF95]. Further work along these lines, in conjunction with work done at Rutgers, is discussed in [BKFS19].

In all of this work *pnorm* has performed exceedingly well. For those interested in investigating probabilistic IR models, however, the need to explore alternatives remains. Our interest is in improving the performance of the boolean operator interpretation within the framework of the inference network. The performance of *pnorm* when applied to the INQUERY system gives us reason to believe that this is possible at the same time that it sets a performance target to be aimed for.

4. PIC MATRICES AND THE PIC-EVAL ALGORITHM

A wide range of functions can be represented as link matrices. Unfortunately, the evaluation of an arbitrary link matrix for an arbitrary set of input probabilities requires $O(2^n)$ floating point operations. In [Tur90], Turtle shows that closed form expressions exist for some matrices, such that they may be evaluated in time linear in the number of parents, for arbitrary inputs. The matrices used for Boolean operators have that characteristic. It is not difficult to show that the probability that the child node is true can be given by:

$$pr(Q \text{ is true}) = p_1 p_2 \cdots p_n$$

for L_{and}, and

$$pr(Q \text{ is true}) = 1 - (1-p_1)(1-p_2)\ldots(1-p_n)$$

for L_{or}, where p_i is the probability that parent P_i is true.

4.1 The PIC matrices

The desire to explore new possibilities for Boolean query operators motivates the search for wider classes of computationally tractable link matrices. A natural candidate for consideration is the class of matrices for which the conditional probability that the child node is true is determined solely by the number of parents that are true, independent of which of them happen to be true and which are false.

We will say that a link matrix satisfies the *parent indifference criterion*, or simply that it is a PIC matrix if, given the parent nodes, $P_1, \ldots, P_n$:

$$\forall R_1, R_2 \subseteq \{P_1, \ldots, P_n\} : |R_1| = |R_2| \Rightarrow \alpha_{R_1} = \alpha_{R_2}$$

where for each $R \subseteq \{P_1, \ldots, P_n\}$:
α_R is the matrix coefficient associated with the event that the P_i belonging to R are true and the P_i not in R are false

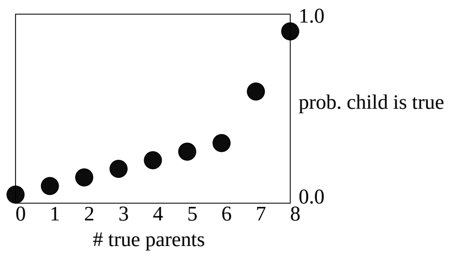

Fig. 5. PIC matrix as a non-decreasing function of number of true parents

A PIC matrix for n parents requires the specification of $n + 1$ parameters, $\alpha_0, \ldots, \alpha_n$, as compared to the 2^n parameters required for an arbitrary link matrix. Each α_i specifies the common value given by α_R for each $R \subseteq \{P_1, \ldots, P_n\}$ such that $|R| = i$. The sequence of link matrix coefficients, $\alpha_0, \ldots, \alpha_n$, can be viewed as a function from the integers $\{0, 1, \ldots, n\}$ to the interval $[0, 1]$.

It is appropriate to note that we have in mind, and will be exploring, coefficients corresponding to non-decreasing functions, such as that shown in figure 5. Viewing the *pic* matrices as operators for the combination of evidence, our interest is in those operators for which more pieces of individual evidence (i.e. greater number of true parents) translates to greater probability that the combined evidence is present. Nonetheless, the PIC matrix class is not limited to such functions.

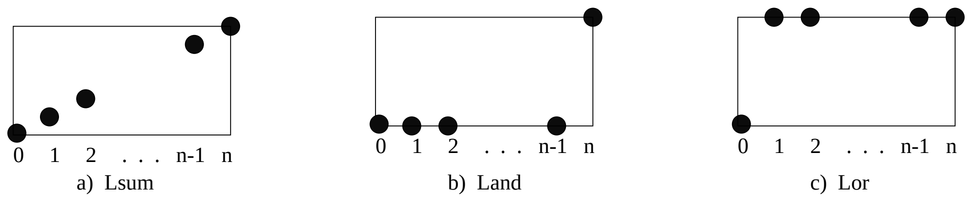

Fig. 6. view of SUM, AND, OR as functions of number of true parents

The L_{sum} matrix discussed earlier satisfies the parent indifference criterion. The coefficients for the general n-ary L_{sum} matrix are $0, \frac{1}{n}, \frac{2}{n}, \ldots \frac{n-1}{n}, 1$ as shown graphically in figure 6a. If the probabilities are viewed as weights of evidence, the matrix can be viewed as an operator that averages these weights. That is:

$$pr(Q \text{ is true}) = \frac{p_1 + p_2 + \ldots + p_n}{n}$$

The L_{and} and L_{or} matrices used for the Boolean operators also meet the parent indifference criterion, as shown in figures 6b and 6c.

In [Tur90], a fourth type of matrix, the *weighted-sum* matrix is shown to be computable in linear time as well. This matrix does not satisfy the parent indifference criterion. It will, however, be shown in section 4.4 to satisfy a generalization of the criterion. This generalization contemplates a relative weighting of the importance of the parent nodes in determining the probability that the child node is true. A simple extension of the PIC-EVAL algorithm allows for the calculation of probabilities involving these *weighted* PIC matrices in $O(n^2)$ time as well.

```
inputs:
  • $n+1$ coefficients, $\alpha_0, \ldots, \alpha_n$, of a pic matrix
  • probabilities $p_1, \ldots, p_n$, for the $n$ parent nodes.
output:
  • the probability that the child node is true.

PIC-EVAL:
    for $j = 0, \ldots, n$
        $\alpha[0, j] = \alpha_j$
    for $i = 1, \ldots, n$
        for $j = 0, \ldots, n-i$
            $\alpha[i, j] \leftarrow \alpha[i-1, j] * (1 - p_i) + \alpha[i-1, j+1] * p_i$
    return $\alpha[n, 0]$
```

Fig. 7. PIC-EVAL algorithm

4.2 The PIC-EVAL algorithm

The PIC matrices would not constitute a useful class if they could not be evaluated in better than exponential time. Figure 7 shows an algorithm for the efficient evaluation of matrices that satisfy the parent indifference criterion. Starting with the original link matrix, L_0, PIC-EVAL, in effect, generates a sequence of smaller and smaller link matrices, $L_1, L_2, \ldots, L_n$. In the process, it eliminates from consideration each probability in turn (figure 8).

To begin, the probability associated with parent node, P_1, is fixed and the matrix, L_0, with $n+1$ coefficients and n parents (figure 8a) is converted to an n coefficient link matrix with $n-1$ parents (figure 8b). As shown in figure 8, each matrix, L_i, connects with one fewer parent node than the previous one. Corresponding to this, the coefficients, $\alpha_{i,0}, \alpha_{i,1}, \ldots, \alpha_{i,n-1}$, of each matrix are fewer as well. The matrices, L_i, that result from this sequence of transformations are equivalent to the original matrix, L_0, in the following sense:

for any set of probabilities for parents,

$$P_{i+1}, \ldots, P_n,$$

L_i yields the same pr(Q is true) that L_0 would produce given the same probabilities for

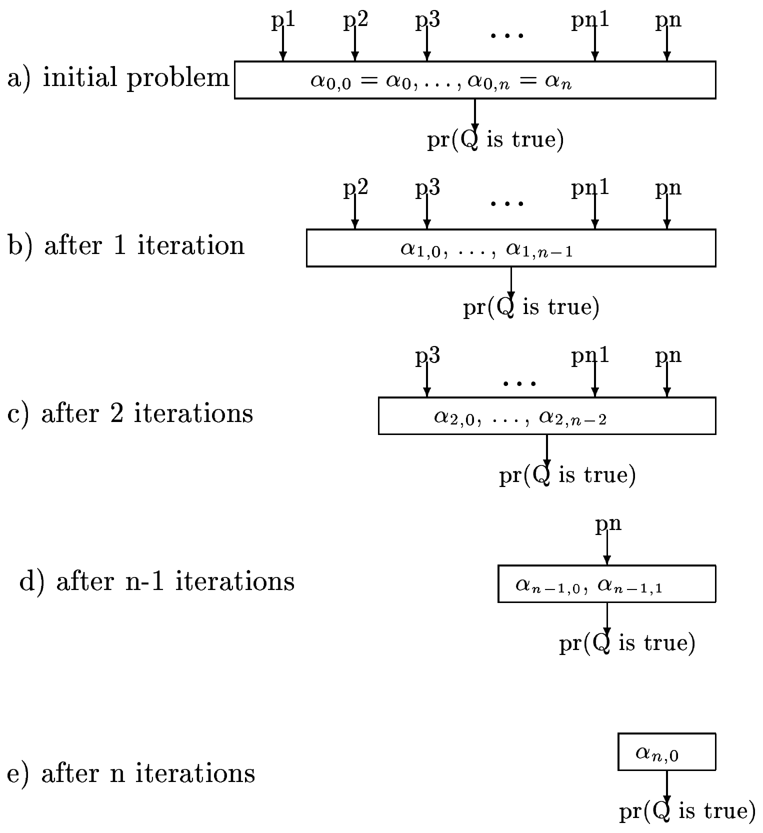

Fig. 8. iterations in the evaluation of a *pic* matrix

$$P_{i+1},\ \ldots,\ P_n,$$

together with the probabilities

$$p_1, \ldots, p_i \textit{ for } P_1, \ldots, P_i.$$

After n iterations we arrive at L_n with exactly one coefficient, $\alpha_{n,0}$, as seen in figure 8e. Now that all parent probabilities have been accounted for, this coefficient can be interpreted as the desired probability that the child is true. A formal proof of the correctness of the PIC-EVAL algorithm can be found in Appendix 1.

The PIC-EVAL algorithm is executed in $O(n^2)$ time. The initialization requires $O(n)$ and the main loop is executed precisely $\sum_{i=0}^{n}\sum_{j=0}^{n-i} 1 = O(n^2)$ times. Also, the constant factor is small. On top of the base iteration control overhead, two multiplications and one addition are required in each iteration. Although, for the purposes of exposition, the algorithm has been shown as requiring $O(n^2)$ space as well as time, $O(n)$ space is easily achieved since only one row's worth of cells need be maintained at any one time.

4.3 Piecewise Linear Functions

The PIC-EVAL algorithm evaluates a matrix with arbitrary PIC coefficients in $O(n^2)$ time. In this section we look at certain PIC matrices for which the evaluation algorithm can be made more efficient.

The PIC-EVAL algorithm can be viewed as filling a triangular portion of a square matrix as shown in figure 9a. The first row is initialized with the α_j values and then each row from 1 to n is processed in turn. Within each row, the cells are set from left to right, with each cell set to the sum of:

—the cell immediately above it, and
—the cell above it and to the right.

As a consequence of this, the value of each cell, $\alpha_{i,j}$, is dependent only on the values of cells in a triangle extending directly above it on the left and at a 45^o angle to the right, as is shown in figure 9b. That is, $\alpha_{i,j}$ depends on only those $\alpha_{k,l}$ such that:

$$\begin{array}{rcl} 0 \le & k & < i \\ j \le & l & < j+m-i \end{array}$$

When a subsequence of the PIC matrix coefficients forms an arithmetic progression,

$$\alpha_m, \alpha_m + \Delta, \alpha_m + 2\Delta, \ldots, \alpha_m + s\Delta$$

the cell values in the triangular subsection immediately below these coefficients can be expressed directly in terms of α_m, Δ, and the parent probabilities, $p_1, \ldots, p_s$. Of these cell values, the only ones that are needed for calculations outside of the triangle are those on the leftmost edge:

$$\alpha_{0,j}, \ldots, \alpha_{i,j}$$

Hence, if there were a direct method for determining these cell values, the rest of the cells in the triangle need not be calculated at all. Exactly, how this may be accomplished follows from the following Lemma, a proof of which is given in Appendix 2.

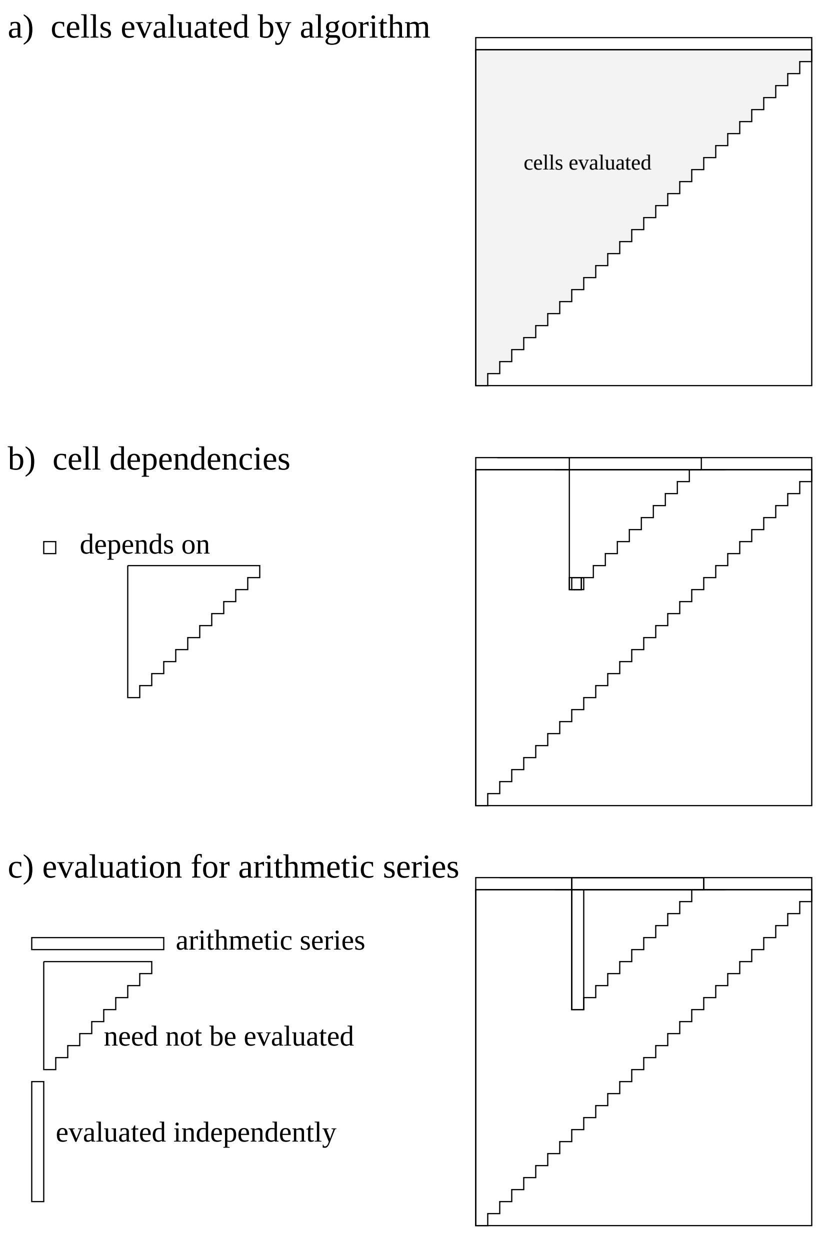

Fig. 9. cell evaluation during execution of PIC-EVAL

LEMMA 4.3.1. *Given a* PIC *matrix whose coefficients,*

$$\alpha_m, \alpha_{m+1}, \alpha_{m+2}, \ldots, \alpha_{m+s}$$

are of the form:

$$\alpha_m, \alpha_m + \Delta, \alpha_m + 2\Delta, \ldots, \alpha_m + s\Delta$$

i.e., are such that:

$$\alpha_{m+j} = \alpha_m + j\Delta \quad \forall j = 0, ..., s$$

then, $\forall i = 0, \ldots, s \quad \forall j = 0, \ldots, s - i$*:*

$$\alpha_{i,m+j} = \alpha_m + j\Delta + \Delta \sum_{l=1}^{i} p_j$$

In particular, each of the cells at the left edge of the triangle can be computed as:

$$\begin{aligned} \alpha_{i,m} &= \alpha_m + 0\Delta + (\Delta \sum_{k=1}^{i} p_k) \\ &= \alpha_m + 0\Delta + (\Delta \sum_{k=1}^{i-1} p_k) + \Delta p_i = \alpha_{i-1,m} p_i + \Delta p_i \end{aligned}$$

which requires a total of only s additions and s multiplications, replacing the $\sum_{i=1}^{s} = s(s+1)/2$ general cell computations which would normally be executed.

This technique can result in substantial savings if the number of coefficients involved, and hence the size of the triangle involved, is large. For example, figure 10a, shows a function with three linear pieces. The matrix coefficients in this case comprise three arithmetic series, each associated with a corresponding savings in cost of evaluation. Since the domain of the function is discrete, it is not strictly necessary that the pieces *connect*. Therefore, when speaking of piecewise linear functions, we shall also consider functions such as that shown in figure 10b.

The form of the 2-piece piecewise linear functions shown in figures 10c and 10d are of interest because they can be interpreted as generalizations of the L_{and} and L_{or} link matrices. We will see in a moment that evaluation of these functions is particularly efficient. The function shown in figure 10c, for instance, generalizes the L_{and} matrix in that the conditional probability that Q is true may:

—take on a (presumably small) value greater than 0 when all parents are false.

—rise (presumably slowly) at a constant rate as more parents are known to be true.

—rise suddenly for some number of true parents m less than n – although m must be less than n by some (presumably small) constant value, independent of n.

—rise at a constant rate as the number of parents known to be true goes from m to n

—take on a value less than 1 when all parents are true.

In a similar fashion, the form shown in figure 10d, generalizes the L_{or} matrix.

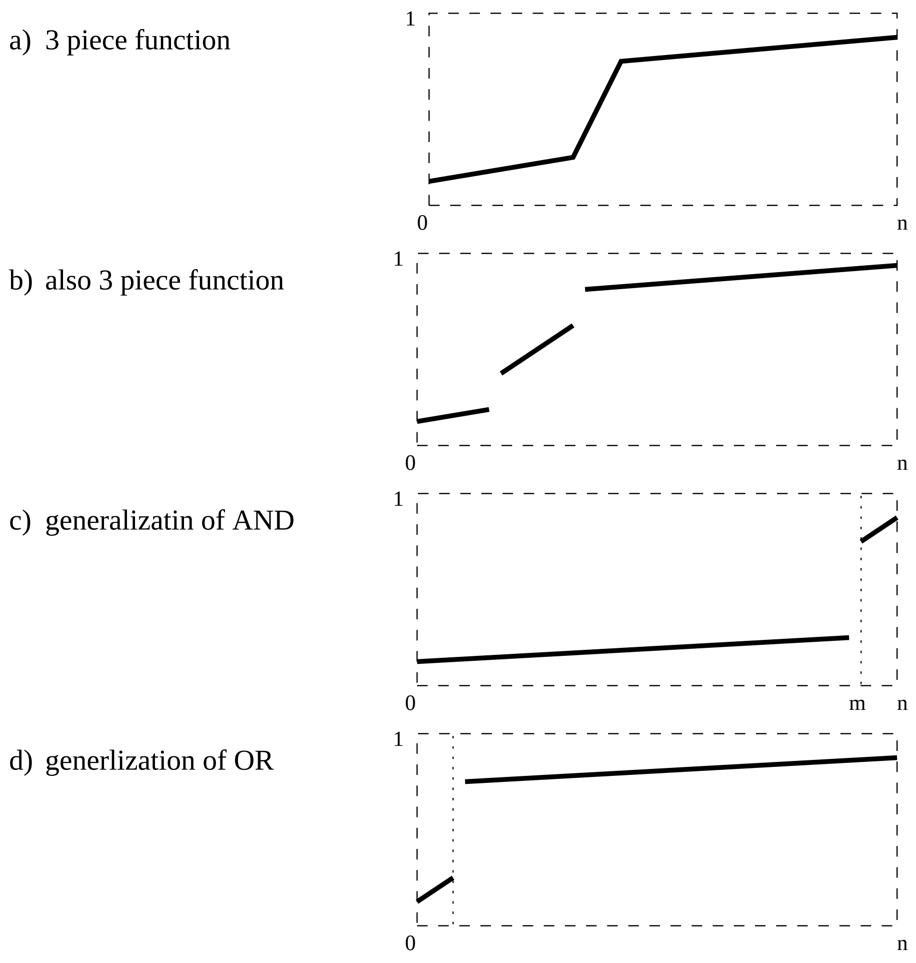

Fig. 10. piecewise linear functions

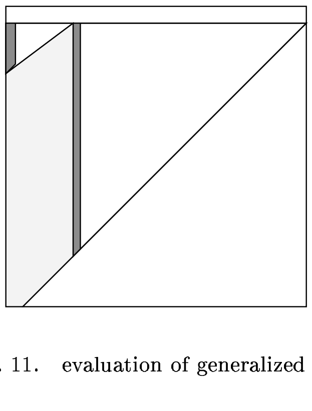

Fig. 11. evaluation of generalized OR

Although the savings realized for a piecewise linear function may be significant, the asymptotic cost of execution will not be affected unless all of the pieces save one cover a constant number of coefficients [3]. If $\alpha_m, \alpha_{m+1}, \ldots, \alpha_s$, is an arithmetic series, then $s(s+1)/2$ cell operations can be eliminated in favor of s additions. This leaves,

$$\begin{aligned} n(n+1)/2 - s(s+1)/2 &= (n^2 + n - s^2 - s)/2 \\ &= ns - s^2 + (n^2 - ns + n + s^2 - ns - s)/2 \\ &= (n-s)s + (n(n-s+1) - s(n-s+1))/2 \\ &= (n-s)s + (n-s)(n-s+1)/2 \end{aligned}$$

When $(n-s)$ is constant, the second term is constant and the first term grows with s. Since s must grow with n, if $n-s$ is to be kept constant, the cost of evaluation is $O(n)$. Figure 11 shows, pictorially, how the array for the generalized OR function is evaluated. The cells that are evaluated are restricted to a fixed width strip at the left of the array whose length grows with n.

4.4 Weighted Parents

In section 4.2 we defined and developed an $O(n^2)$ algorithm for the class of link matrices for which the conditional probabilities are dependent only on the number of parents that are true. In this section, we generalize this result, beginning with the class of link matrices under consideration.

The goal of this generalization is to allow for the possibility that the truth of each proposition, P_i, may have a different impact on the probability, $pr(Q$ is true). There will always be one parent whose impact is at least as great as any other. For the purposes of this exposition we will assume, without loss of generality, that parent, P_1, has this property. The impact of each other parent may, then, be *weighted* by some factor, $0 \leq w_i \leq 1.0$. For generality, we will say that all parents are weighted and that the weight, w_1, of parent P_1 is equal to 1. We shall say that a link matrix, satisfies the *weighted-parent indifference criterion* when the probability that the child is true is strictly a function of the number of parents that are true, once the weights of the parents have been taken into consideration. Formally,

Definition 4.4.1. We shall say that the *weighted-parent indifference criterion* is met when:

$$\begin{aligned} \exists \ (&w_1 = 1.0, \\ &0 \leq w_2, \ldots, w_n \leq 1.0 \\ &0 \leq \alpha_0, \ldots, \alpha_n \leq 1.0) \\ &\quad \forall \ R \subseteq \{1, \ldots, n\} \qquad \alpha_R = \alpha_j \prod_{i \in R} w_i \end{aligned}$$

We shall refer to a link matrix satisfying the weighted-parent indifference criterion as a WPIC matrix. A WPIC matrix is completely determined when two sets of parameters have been given: $\alpha_0, \ldots, \alpha_n$ and $w_1, \ldots, w_n$. When all the parents

[3]Prof. David Barrington has pointed out that computational improvement may be realizable even if this condition is not fully met. For example, if one piece grows with n, while all others grow only as $log(n)$, the asymptotic running time will be only $O(n \log(n))$.

```
inputs:
  • $n$ weights, $w_1, \ldots, w_n$
  • $n+1$ coefficients, $\alpha_0, \ldots, \alpha_n$, of a *pic* matrix
  • probabilities $p_1, \ldots, p_n$, for the $n$ parent nodes.
output:
  • the probability that the child node is true.

PIC-EVAL:
    for $j = 0, \ldots, n$
        $\alpha[0, j] = \alpha_j$

    for $i = 1, \ldots, n$
        for $j = 0, \ldots, n-i$
            $\alpha[i, j] \leftarrow \alpha[i-1, j] * (1 - p_i) + \alpha[i-1, j+1] * w_i * p_i$

    return $\alpha[n, 0]$
```

Fig. 12. PIC-EVAL algorithm

have a weight of, $w_i = 1$, the WPIC matrix reduces to a simple PIC matrix, where the coefficient, α_j, specifies the probability that Q is true when it is known that j parents are true.

For the general WPIC matrix, however, rather than specify what the probability that Q is true *is*, for j true parents, the coefficient, α_j, specifies what that probability *would be* for j true parents, if all the parents had the same impact; that is, *if all the weights were 1.* A weight, $w_i < 1$, indicates that when P_i is one of the true parents, the probability, $pr(Q$ is true), is less than it would be were P_i to have the same weight as P_1. The weight, w_i, gives the factor by which $pr(Q$ is true) must be *discounted* when P_i, rather than a parent whose impact is equal to that of P_1, is one of the true parents [4].

The basic algorithm requires only a minor modification in order that WPIC matrices be processed correctly. The WPIC-EVAL algorithm incorporating the necessary modification is given in figure 12.

The notation, lemma and theorem introduced in appendix A to demonstrate the correctness of the PIC-EVAL algorithm can generalized to contemplate weights. The reasoning used in the proof given there applies with very minor modifications to the situation pertaining to the WPIC-EVAL algorithm. A complete proof of the correctness of the weighted version of the PIC-EVAL algorithm can be found in [Gre96].

It should be observed here that, with respect to computational efficiency, the only

[4]Another way of understanding the WPIC matrices is in terms of *gated* inputs. Each parent is viewed as one input to an AND gate whose other input corresponds to an independent *activation* variable. Only when both the parent and the activation variable are both true is a value of true seen by the Q node. The parent weight becomes the probability that the activation variable is true. From this, we see that the class of WPIC matrices can be viewed as a generalization of the *noisy or* matrices often utilized in Bayesian Network applications.

difference between the two versions of the algorithm is that the weighted algorithm requires an extra multiplication during each execution of the main loop.

5. EXPERIMENTATION

In order to test the potential of the PIC matrices as candidates for the implementation of Boolean query operators, a number of experiments were run with the INQUERY system. For these experiments, two test collections were used: the INSPEC collection and volume 1 of the TIPSTER collection. For the TIPSTER collection, tests were run using two Boolean versions of queries 51 to 100. In this paper, we shall refer to these two sets of queries as TIPSTER-1 and TIPSTER-2. Two Boolean query sets, which we will call INSPEC-1 and INSPEC-2, were also used for testing with the INSPEC collection. For each collection, the two query sets were created by having two different people take the same set of unstructured natural language queries and reformulate them using Boolean operators [BCCC93]. For all experiments discussed in this paper, the INQUERY system using the AND and OR operators as defined in [Tur90] shall be considered the baseline system for purposes of comparison. For these query sets, experiments have shown that *pnorm* operators perform from 7% to 28% better than the baseline system in terms of average precision. Our goal was to achieve similar performance using PIC matrices.

A number of different types of PIC matrices were tried using the standard INQUERY formula for estimating the probability that a document is *about* a concept:

$$\begin{array}{rcl}
\multicolumn{3}{l}{0.4 + 0.6 \times \frac{tf}{0.5+1.5\times\frac{dl}{avg\_dl}} \times \frac{log(\frac{dc+0.5}{df})}{log(dc+1.0)}} \\
\text{where:} \quad tf &=& \text{term freq. (no. of occur of term in doc.)} \\
dl &=& \text{doc. length (no. of tokens in doc.)} \\
avg\_dl &=& \text{avg. doc. length (over collection)} \\
dc &=& \text{doc. count} = \text{no. of docs in collection} \\
df &=& \text{doc. freq.} = \text{no. of docs containing term}
\end{array} \qquad (3)$$

Although we were able to realize improvements over the baseline system, it was not possible to obtain results consistently on a par with those produced using the *pnorm* operators. Analysis of the link matrix computations led us to believe that the difficulty might lie in the value used for the *default belief.* In eq. 3, 0.4 is this default belief: the probability of a document being about the concept of interest that will be assigned when the associated term count is zero.

Intuitively, one might expect that a lack of evidence in favor of a subquery, in the form of a zero term count, should correspond to the same *default* belief value, independent of whether or not an operator is involved; and when an operator is specified, independent of which operator it is. Take the query,

```
#OR(t1, #AND(t2, t3), #AND(t4, t5, t6))
```

for example, and let us assume that all term counts are zero. If the probability associated with `t1` is β, we could argue that the probabilities associated with `#AND(t2, t3)` and `#AND(t4, t5, t6)` should both be β also; and, further, that the belief that the entire query is satisfied should be β, as well. In general, we might posit the following principle:

$$\forall n : \beta = \text{default belief } \wedge p(P_1) = p(P_2) = \ldots = p(P_n) = \beta$$
$$\Rightarrow p(\#\textsc{and}(P_1, \ldots, P_n)) = p(\#\textsc{or}(P_1, \ldots, P_n)) = \beta$$

Although, the applicability of this principle to the interpretation of Boolean queries is not beyond question, it seemed natural to us to explore belief systems that obey it.

The PIC class is sufficiently rich so that, for any preset value of the default belief, matrix families can be engineered such that the above principle is satisfied. Experiments with the default belief left at 0.4 were somewhat disappointing. With the default belief set to 0.0, however, it seemed that robust behavior had been obtained, at a level of performance comparable to that of the *pnorm* operators. We return to the point of the default belief value setting in section 6.

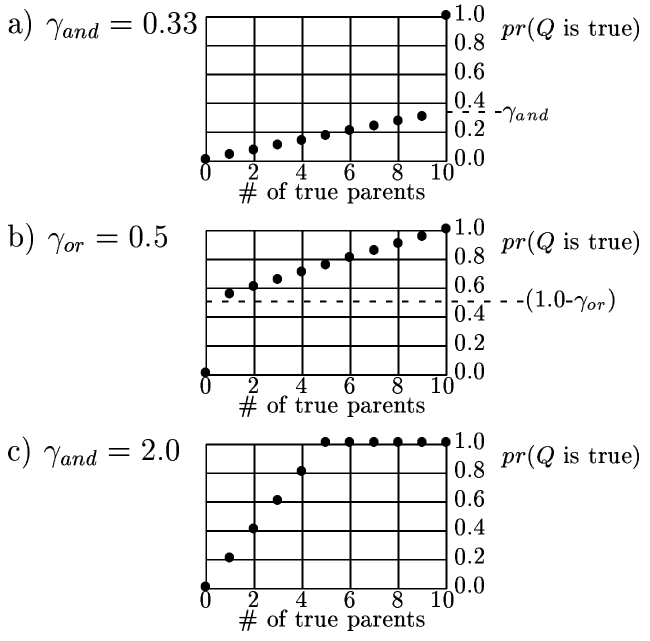

Fig. 13. *sloped* PIC matrices for Boolean operators

The experiments described here concentrated on a family of operators, initially described in [Tur90], that varied in a simple way from the AND and OR operators used in the current INQUERY system. The standard AND operator can be viewed as a linear sequence of $n-1$ points, $\alpha_0, \alpha_1, \ldots, \alpha_{n-1}$, together with the coefficient $\alpha_n = 1.0$. We can think of the sequence of $n-1$ coefficients as rising with a slope of 0.0. In these experiments we consider PIC matrices for the AND operator that are of the same form, but for which the slope may rise with a slope of $\gamma_{and} > 0$. Presumably, the rise will be gradual, with γ_{and} set to a fairly small fraction, as shown in figure 13a where $\gamma_{and} = 0.33$. To simplify the experimentation, the slope was maintained constant over the entire family of AND operators. That is, for a given experiment, the coefficients for each different arity of the AND operator were determined based on the same setting of γ_{and}.

The PIC matrix candidates for the OR operators were defined in an analogous manner as is shown in figure 13b. As with the AND operator, α_0, and α_n are fixed at 0.0 and 1.0, respectively. In the case of the OR operator, γ_{or} corresponds to the rate of decrease in the values of the coefficients as the number of true parents decreases in the sequence, $\alpha_n, \alpha_{n-1}, \ldots, \alpha_2$.

We shall see that, in the case of the γ_{and} parameter, we experimented with values greater than 1.0. As shown in figure 13c, for these values the matrix is defined such that the PIC coefficients rise with the slope, γ_{and}, up to a maximum of 1.0. Values which would otherwise be greater than 1.0 are truncated to 1.0.

	$\gamma_{or} = 0.2$	0.4	0.6	0.8	1.0
$\gamma_{and} = 0.2$	+14.3	+15.3	+16.6	+16.0	+14.5
0.4	+16.9	+17.8	+18.7	+17.5	+14.4
0.6	+17.6	+18.2	+19.0	+17.6	+14.9
0.8	+18.7	+18.9	+19.4	+17.6	+15.4
1.0	+20.0	+20.3	$\boxed{+20.4}$	+18.4	+15.5

(□ greatest improvement)

Table I. INSPEC-1 queries on INSPEC: PIC with $\gamma_{and}, \gamma_{or} \leq 1.0$

Experiments were run first with the INSPEC collection. Table I shows the percent improvement in 11-pt average precision with respect to the baseline system, for different settings of the γ_{and} and γ_{or} parameters. For all values of γ_{and} tested (including values not shown in the above table), optimal performance was achieved with γ_{or} set to 0.6. For all values of γ_{and}, performance improves monotonically with increasing values of γ_{and}, not reaching a maximum value until γ_{and} is at 1.0. This is surprising because AND operators with greater values of γ_{and} can be interpreted as operators that increasingly ignore the semantics usually associated with AND, namely, that AND is a simple conjunction. When $\gamma_{and} = 1$, the AND operator behaves exactly like a SUM operator. When $\gamma_{and} > 1$ the operator takes on a distinctly OR-like flavor; the presence of a relatively small number of conjoined terms leads to relatively large belief scores.

	$\gamma_{or} = 0.2$	0.4	0.6	0.8	1.0
$\gamma_{and} = 1.0$	+20.0	+20.3	+20.4	+18.4	+15.5
2.0	+23.9	$\boxed{+26.1}$	+24.9	+22.3	+17.6
3.0	+25.6	$\boxed{+26.1}$	+25.3	+22.4	+17.1
4.0	+25.1	+25.9	+24.4	+21.1	+15.3
5.0	+25.1	+25.9	+24.5	+21.2	+15.4
6.0	+25.1	+25.9	+24.5	+21.1	+15.4
7.0	+25.1	+25.9	+24.5	+21.1	+15.4

Table II. INSPEC-1 queries on INSPEC: PIC with $\gamma_{and} \geq 1.0$

The consistent peak of performance at $\gamma_{and} = 1$ suggested tests with $\gamma_{and} > 1$. For all values of γ_{or}, performance continues to improve as γ_{and} increases until

γ_{and} reaches 2.0 (Table II). This suggests that users do not necessarily interpret AND as a logical operator. The use of the AND connector appears to indicate that the likelihood that a document will be judged relevant grows quickly with the number of terms present in the document. This is in stark contrast with the normal propositional logic interpretation in which high likelihoods are assigned only to documents containing all of the terms.

	p_{or} = 1.0	2.0	3.0	4.0	5.0	6.0	7.0
p_{and} = 1.0	+15.5	+16.1	+16.0	+15.7	+15.6	+16.0	+16.4
2.0	+16.4	+17.0	+17.0	+17.1	+17.7	+17.1	+16.1
3.0	+16.6	+17.0	+17.2	+17.8	+17.9	+17.1	+16.2
4.0	+16.7	$\boxed{+18.2}$	+18.0	+17.7	+17.5	+16.8	+16.7
5.0	+17.6	+17.8	+17.2	+16.6	+16.4	+15.8	+15.5
6.0	+16.4	+16.5	+15.7	+15.5	+15.5	+15.0	+14.3

Table III. INSPEC-1 queries on INSPEC: *pnorm*

Table III shows the performance of the *pnorm* operator for various values of p_{and} and p_{or}, the settings of the parameter, p, for AND and OR respectively. We see that, as with the PIC matrix operators, the *pnorm* operators perform robustly over a wide range of parameter settings. Performance of the PIC matrix operator is clearly superior to that of the *pnorm* operators for this set of queries against the INSPEC collection. Peak performance improvement for the PIC matrix version of the operators reaches 26.1% as compared to 18.2% for the *pnorm* version. Tables IV and V show the comparative performance for INSPEC-2, an alternative Boolean formulation of the same queries, against the same collection. Performance is similar to that of the previous query set with the optimal performance of the PIC operators slightly lower at 22.8% improvement as compared to 18.0% for *pnorm.*

	γ_{or} = 0.0	0.2	0.4	0.6	0.8	1.0
γ_{and} = 1.0	+18.6	+19.5	+20.0	+20.1	+18.9	+14.3
2.0	+21.4	+22.0	+22.4	+22.8	+20.7	+14.1
3.0	+21.5	+22.3	+22.6	+22.7	+20.1	+13.3
4.0	+21.7	+22.5	$\boxed{+22.8}$	$\boxed{+22.8}$	+20.1	+12.5
5.0	+21.6	+22.5	$\boxed{+22.8}$	$\boxed{+22.8}$	+20.1	+12.5
6.0	+21.6	+22.5	$\boxed{+22.8}$	$\boxed{+22.8}$	+20.1	+12.5

Table IV. INSPEC-2 queries on INSPEC: PIC

	p_{or} = 1.0	2.0	3.0	4.0	5.0	6.0	7.0
p_{and} = 1.0	+14.3	+15.7	+16.3	+16.8	+17.1	+17.5	+16.9
2.0	+15.2	+16.4	+17.5	+17.6	+17.9	+17.8	+16.8
3.0	+15.5	+16.3	+16.9	+17.2	+17.9	+17.7	+16.9
4.0	+15.9	+16.9	+17.2	+17.5	$\boxed{+18.0}$	$\boxed{+18.0}$	+17.1
5.0	+15.2	+16.4	+16.7	+16.9	+17.4	+17.5	+16.7
6.0	+14.9	+15.4	+15.8	+16.5	+16.7	+16.8	+16.1

Table V. INSPEC-2 queries on INSPEC: *pnorm*

A similar set of experiments were run on the, larger, TIPSTER collection. Table VI summarizes the performance of the two approaches. We see here that both classes of operators are capable of significantly outperforming the baseline system on both query sets. On the first query set, the best parameter setting for the *pnorm* operators slightly outperforms the best settings for the PIC operators. On the second query set, it is the best settings of the PIC operators that yield superior performance. The smaller improvements for both systems on the second query set is due to the better performance of the baseline system. It is unclear why the baseline system is apparently so much more sensitive to the differences in the two query formulations than the *pnorm* and PIC versions of the system. Surprisingly, in light of the previous results, this best performance for the PIC system on the second query set is obtained with the γ_{and} coefficient set as low as 0.1. To date, we have been unable to explain this phenomenon.

Table VII compares the two operators with *pnorm* settings of $p_{and} = 6.0$, $p_{or} = 3.0$ and PIC settings of $\gamma_{and} = 2.0$, $\gamma_{or} = 0.6$. These parameter settings were chosen so as to give a best overall performance profile for each of the operator classes. In general, the optimal settings for the *pnorm* operator in these experiments tend to be somewhat higher than those reported in [SFW83]. For comparable experiments they found that values between 1.0 and 2.0 produced best results.

	baseline	*pnorm*	% impr.	PIC	% impr.
INSPEC-1	26.5	4.0/2.0	18.2	2.0/0.4	26.1
INSPEC-2	25.6	4.0/5.0	18.0	4.0/0.4	22.8
TIPSTER-1	20.8	9.0/1.0	28.8	2.0/0.8	27.8
TIPSTER-2	25.0	10.0/1.0	7.8	0.1/0.6	9.8

Table VI. comparison of PIC and *pnorm* on all four query sets - optimal parameter settings

	baseline	*pnorm*	% impr.	PIC	% impr.
INSPEC-1	26.5	30.7	15.7	33.1	24.9
INSPEC-2	25.6	29.6	15.8	31.4	22.8
TIPSTER-1	20.8	26.5	27.3	26.4	26.9
TIPSTER-2	25.0	26.7	6.8	26.4	5.5

Table VII. $p_{and}/p_{or} = 6.0/3.0$; $\gamma_{and}/\gamma_{or} = 2.0/0.6$

6. CONCLUSIONS

From the work reported here, we may conclude that combining functions with a well-defined probabilistic interpretation can be associated with Boolean query operators which:

—appear to perform as well as the *pnorm* operators,

—can be realized at reasonable computational cost.

6.1 Probabilistic interpretation

The definition of the *pnorm* functions is an outgrowth of intuitions grounded in the vector space model of information retrieval [Sal89]. The PIC matrices, on the other hand, are the result of viewing the scoring of documents from a probabilistic standpoint. The i^{th}coefficient of an n-ary PIC matrix corresponds to the conditional probability that the compound query is satisfied by an arbitrary document, given that exactly i of the n component sub-queries are satisfied by that document.

The existence of such an interpretation for the combining functions means that these functions may be analyzed within the probabilistic framework. Their comparative behavior on varying collections and/or varying query sets can be studied from a probabilistic vantage point. Attempts to improve overall performance of the operators can be guided by researchers' intuitions as to the probabilities involved. Coherent techniques for combining Boolean operators with others, such as proximity or phrase operators, can be developed in accord with the laws of probability theory.

The incorporation of these Boolean operators, in the context of an encompassing probabilistic IR system, allows for the meaningful analysis of the overall system in probabilistic terms. If the ranking scores produced by a system, as well as the intermediate scores manipulated by that system in arriving at these final values, can justifiably be interpreted as probabilities, measurements can be made of how well these numbers correspond to observed frequencies [Daw89; Bri50]. Analysis of these measurements can result in insights into the strengths and weaknesses of the system; unwarranted (possibly, hidden) assumptions; suspect parameter settings; and possibilities for improvement. Such insights may not be easily forthcoming in the absence of semantics for the numbers involved and a theoretical framework within which the manipulation of these numbers can be understood.

6.2 Performance

From the four different query set formulations studied, it appears that combining functions based on PIC matrices perform as well as the *pnorm* functions. For the researcher inclined toward the probabilistic approach, this is comforting. The definition of the *pnorm* operators is an excellent example of how a mathematical model, in this case the vector space model, can guide the researcher toward the development of fruitful ideas. We have shown here, that, at least as far as the current state of the art with respect to Boolean operators is concerned, a probabilistic theory of information retrieval can be equally beneficial in this regard. It is our belief, and apparently that of other "probabilists" (for example, see [Coo94] for an interesting exposition of the issues), that in the long run, models founded in probability theory will prove to be more fruitful in this regard. We believe that probability theory will be found to be well suited to the task of modeling the complexities of the information needs of users, the information content of documents, and the matching of one to the other. We think it will ultimately be found to be more useful than a geometric model where the mapping between the formal constructs and the aspects of information retrieval that are being modeled is, to our way of thinking at least, less obvious.

The results obtained here also suggest that some previous experimental results might merit a second look. For example, [Tur94] reports that natural language queries consistently outperform boolean queries for experiments run on legal collections. The difference observed, however, is within the range of the improvements we have been able to obtain, suggesting that more effective implementations of the boolean query operators might close the apparent gap.

6.3 Efficiency

Probability calculation involving PIC matrices can be realized in $O(n^2)$ time. The CPU time for the PIC matrix version of INQUERY was compared against the baseline system for each of the query sets discussed in this paper. An increase of from 35% to 65% in CPU time was observed. This would seem to be a reasonable price to pay for the corresponding increases in performance, especially since the percentage increase in overall response time can be expected to be significantly smaller when I/O time is factored in. Exactly what that overall increase would be will depend, of course, on the characteristics of the particular hardware used to run the system[5].

6.4 Future work

The experimentation reported here indicates that PIC matrices can be advantageously applied to the modeling of Boolean queries. Three ways in which we are considering extending this work are:

alternative sub-classes of the PIC *matrices.* The PIC matrices cover a wide range of functions which can be chosen for modeling belief-combination operators. The definition of an operator family requires the specification of $n+1$ parameters for each arity, $n = 2, 3, \ldots$. We have had success with a subclass of the PIC matrices that can be specified with two parameters: the increasing slope of the AND operator and the decreasing slope of the OR operator. This class may not be optimal. For one thing, the definition of the slope parameter constrains the rate of rise from 0 to $n-1$ for the different members of the AND family to be related in a given way. Specifically, the increase in belief is γ_{and}/n for each additional true parent. Equivalently, the belief at the *knee* ($n-1$ true parents) is $(n-1)\gamma_{and}/n$. Similarly for the OR family. Other relationships among the members of the family could be explored. For example, the position of the probability at the knee could be held constant over the entire family; its value being an empirically determined parameter setting. Alternatively, some simple, but non-linear, function for the increase in probability from 0 to $n-1$ for AND (and decrease for OR) might give better results. Another possibility worth exploring would be to allow for the setting of AND/2 and OR/2 to be independent of the settings chosen for arities of $n = 3, 4 \ldots$.

experimenting with default probabilities. We have obtained our best results by eliminating the default setting of 0.4 in favor of a 0.0 setting. It is possible that default settings at some value somewhat above 0.0 will yield small, but consistent, performance gains. When allowing for a non-zero default setting, β, it would probably be best to restrict the operator families such that each operator produces an

[5]As mentioned in section 4.3, a linear time version of the PIC-EVAL algorithm is applicable when the PIC matrix is a piecewise linear function. Although this more efficient version of the algorithm would be applicable to the PIC matrices utilized for these particular experiments, the general PIC matrix algorithm was used. The times reported correspond to the general version of the algorithm.

output of β when all inputs are at β. This would impose an algebraic constraint that could be satisfied in a number of ways. There is as little justification for assuming certainty of relevance ($pr(Q)$=1.0) as there is for assuming certainty of non-relevance ($pr(Q)$=0.0) even in the (hypothetical) case of certainty with respect to the truth or falsity of the parent propositions.

correlating beliefs with long term frequencies. If the design of the Boolean operators follows probabilistic intuitions, it is reasonable to expect that their success in practice will be correlated with the extent to which the values being manipulated correspond to probabilities in the real world. It should be instructive to analyze the behavior of the operators after performing some kind of regression to convert the document scores into scores more accurately reflecting observed frequencies of relevance for large text collections.

REFERENCES

N. J. Belkin, C. Cool, W. B. Croft, and J. P. Callan. The effect of multiple query representations on information retrieval system performance. In Robert Ksorfhage, Edie Rasmussen, and Peter Willett, editors, *Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 339–346, June 1993.

D. A. Buell and D. H. Kraft. Threshold values and Boolean retrieval system. *Information Processing & Management*, 17(3):127–136, 1981.

N. J. Belkin, P. Kantor, E. A. Fox, and J. A. Shaw. Combining the evidence of multiple query representations for information retrieval. *Information Processing & Management*, 31(3):431–448, 19.

A. Bookstein. Fuzzy requests: an approach to weighted Boolean searches. *Journal of the American Society for Information Retrieval*, 31(4):240–247, July 1980.

A. Bookstein. A comparison of two systems of weighted Boolean queries. *Journal of the American Society for Information Retrieval*, 32(4):275–279, July 1981.

Glenn W. Brier. Verificatyion of forecasts expressed in terms of probability. *Monthly Weather Review*, 78(1):1–3, January 1950.

J. P. Callan, W. B. Croft, and S. M. Harding. The inquery retrieval system. In *Proceedings of the 3rd International Conference on Database and Expert Systems Applications*, pages 78–83, 1992.

E. Charniak. Bayesian networks without tears. *AI Magazine*, 12(4):50–63, April 1991.

William S. Cooper. The formalism of probability theory in IR: A foundation or an encumbrance. In W. Bruce Croft and C. J. van Rijsbergen, editors, *Proceedings of the 17th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*, pages 242–248, Dublin, Ireland, July 1994.

A. P. Dawid. Probability forecasting. In Samuel Kotz and Norman L. Johnson, editors, *Encyclopedia of Statistical Sciences*, volume 7, pages 210–218. Wiley, New York, 1989.

E. Fox, S. Betrabet, and M. Koushik. Extended boolean models. In William B. Frakes and Ricardo Baeza-Yates, editors, *Information Retrieval: Data Structures and Algorithms*, pages 393–418. Prentice-Hall, Englewood Cliffs, NJ, 1992.

Edward A. Fox and Joseph A. Shaw. Combination of Multiple Searches. In D. K. Harmon, editor, *The Second Text REtreival Conference (TREC-2)*, pages 243–248, Gaithersburg, Md., March 1994. NIST Special Publication 500-215.

Warren R. Greiff. Computationally tractable, conceptually plausible classes of link matrices for the inquery inference network. Technical Report CmpSci TR-96-66, University of Massachusetts, Amherst, Massachusetts, September 1996.

J. H. Kim and J. Pearl. A computational model for combined causal and diagnostic reasoning in inference systems. In *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, pages 190–193, Karlsruhe, West Germany, August 1983.

Leah S. Larkey. personal communication, 1996.

J. H. Lee. Analyzing the effectiveness of extended Boolean models in Information Retrieval. Technical Report TR95-1501, Cornell University, 1995.

T. Noreault, M. Koll, and M. J. McGill. Automatic ranked output from Boolean searches in SIRE. *Journal of the American Society for Information Retrieval*, 28(6):333–339, 1977.

J. M. Ortega. *Numerical Analysis: A Second Course.* Academic Press, New York, 1972.

Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann, San Mateo, CA, 1988.

Gerard Salton. *Automatic text processing : the transformation, analysis, and retrieval of information by computer.* Addison-Wesley Publishing Company, Reading, MA, 1989.

G. Salton, C. Buckley, and E. A. Fox. Automatic query formulations in information retrieval. *Journal of the American Society for Information Retrieval*, 34(4):262–280, July 1983.

Joseph A. Shaw and Edward A. Fox. Combination of Multiple Searches. In D. K. Harmon, editor, *The Third Text REtreival Conference (TREC-3)*, pages 105–108, Gaithersburg, Md., April 1995. NIST Special Publication 500-225.

Gerard Salton, Edward A. Fox, and Harry Wu. Extended Boolean information retrieval. *Communications of the ACM*, 26(12):1022–1036, December 1983.

Howard Turtle and W. Bruce Croft. Inference networks for document retrieval. In Jean-Luc Vidick, editor, *Proceedings of the 13th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*, pages 1–24, Brussels, Belgium, September 1990.

Howard R. Turtle. *Inference Networks for Document Retrieval.* PhD thesis, University of Massachusetts, 1990.

Howard Turtle. Natural language vs. boolean query evaluation: A comparison of retrieval performance. In W. Bruce Croft and C. J. van Rijsbergen, editors, *Proceedings of the 17th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*, pages 212–220, Dublin, Ireland, July 1994.

W. G. Waller and D. H. Kraft. A mathematical model for a weighted Boolean retrieval system. *Information Processing & Management*, 15(5):235–245, 1979.

APPENDIX

1. CORRECTNESS OF THE PIC-EVAL ALGORITHM

The correctness of the pic algorithm follows directly from Lemma 1.4, which is more easily expressed by adopting the following notation.

Notation 1.1. With respect to the execution of the *pic* matrix evaluation algorithm: $\boldsymbol{L_i}$ shall refer to the link matrix generated during the i^{th} iteration, and the coefficient of L_i associated with j parents being true shall be referred to as $\boldsymbol{\alpha_{i,j}}$. We note that for the initial matrix, L_0:

$$\alpha_{0,j} = \alpha_j \quad \forall i = 0, \ldots, n$$

Notation 1.2. Given an arbitrary subset, R, of $\{i_1, \ldots, i_2\}$, $\boldsymbol{\pi_R^{\{i_1,\ldots,i_2\}}}$ shall denote the probability that precisely the parents, P_i such that $i \in R$, are true, while those parents, P_i such that $i \notin R$, are false. Equivalently:

$$\pi_R^{\{i_1,\ldots,i_2\}} = \prod_{i \in R} p_i \prod_{i \in \{i_1,\ldots,i_2\}-R} \bar{p}_i$$

Notation 1.3. The notation $\boldsymbol{\sigma_j^{\{i_1,\ldots,i_2\}}}$ will be used to denote the probability that exactly j of the propositions of $\{P_{i_1}, \ldots, P_{i_2}\}$ are true. We observe that:

$$\sigma_j^{\{i_1,\ldots,i_2\}} = \sum_{\substack{R \subseteq \{i_1,\ldots,i_2\} \\ |R|=j}} \pi_R^{\{i_1,\ldots,i_2\}}$$

It is worth noting that the probability that Q is true can be given by:

$$\begin{aligned} pr(Q \text{ is true}) &= \sum_{R \subseteq \{1,\ldots,n\}} [\alpha_R \prod_{i \in R} p_i \prod_{i \in \{1,\ldots,2\}-R} \bar{p}_i] \\ &= \sum_{R \subseteq \{1,\ldots,n\}} [\alpha_R \pi_R^{\{1,\ldots,n\}}] \end{aligned}$$

And that when the parent indifference criterion is met:

$$\begin{aligned} pr(Q \text{ is true}) = \sum_{R \subseteq \{1,\ldots,n\}} [\alpha_R \pi_R^{\{1,\ldots,n\}}] &= \sum_{j=0}^{n} \sum_{\substack{R \subseteq \{1,\ldots,n\} \\ |R|=j}} [\alpha_R \pi_R^{\{1,\ldots,n\}}] \\ &= \sum_{j=0}^{n} \sum_{\substack{R \subseteq \{1,\ldots,n\} \\ |R|=j}} [\alpha_j \pi_R^{\{1,\ldots,n\}}] \\ &= \sum_{j=0}^{n} [\alpha_j \sum_{\substack{R \subseteq \{1,\ldots,n\} \\ |R|=j}} \pi_R^{\{1,\ldots,n\}}] \\ &= \sum_{j=0}^{n} \alpha_j \sigma_j^{\{1,\ldots,n\}} \end{aligned}$$

Proof of the correctness of the PIC-EVAL algorithm will be a direct consequence of the following lemma.

LEMMA 1.4. *Assuming all coefficients $\alpha_{i,j}$ are those produced by the* PIC-EVAL *algorithm, then* $\forall i = 0, 1, \ldots, n$

$$\sum_{j=0}^{n-i} \alpha_{i,j} \sigma_j^{\{i+1,\ldots,n\}} = pr(Q \textit{ is true})$$

Lemma 1.4 effectively states that the set of coefficients, $\alpha_{i,0}, \ldots, \alpha_{i,n-i}$, can be interpreted as specifying a *pic* matrix, L_i, connecting parent nodes $P_{i+1}, \ldots, P_n$, to Q, and that this matrix is, in a sense, equivalent to the original matrix, L_0.

PROOF. This lemma is proved by induction on the value of i. For $i = 0$, we have:

$$\begin{aligned}
\sum_{j=0}^{n} \alpha_{0,j} \sigma_j^{\{1,\ldots,n\}} &= \sum_{j=0}^{n} \alpha_j \sum_{\substack{R \subseteq \{1,\ldots,n\} \\ |R|=j}} \pi_R^{\{1,\ldots,n\}} \\
&= \sum_{j=0}^{n} \sum_{\substack{R \subseteq \{1,\ldots,n\} \\ |R|=j}} [\alpha_R \prod_{i \in R} p_i \prod_{i \notin R} \bar{p}_i] \\
&= \sum_{R \subseteq \{1,\ldots,n\}} [\alpha_R \prod_{i \in R} p_i \prod_{i \notin R} \bar{p}_i] \\
&= pr(Q \text{ is true})
\end{aligned}$$

Assume the theorem to be true for $i = k$:

$$\sum_{j=0}^{n-k} \alpha_{k,j} \sigma_j^{\{1,\ldots,n\}} = pr(Q \text{ is true})$$

Then the sum for $i = k + 1$ is:

$$\begin{aligned}
&\sum_{j=0}^{n-k-1} \alpha_{k+1,j} \sigma_j^{\{k+2,\ldots,n\}} \\
&= \sum_{j=0}^{n-k-1} [\alpha_{k,j} \bar{p}_{k+1} + \alpha_{k,j+1} p_{k+1}] \sigma_j^{\{k+2,\ldots,n\}} \\
&\quad \text{(by step (5) of the algorithm)} \\
&= \sum_{j=0}^{n-k-1} \alpha_{k,j} \bar{p}_{k+1} \sigma_j^{\{k+2,\ldots,n\}} + \sum_{j=0}^{n-k-1} \alpha_{k,j+1} p_{k+1} \sigma_j^{\{k+2,\ldots,n\}} \\
&= \sum_{j=0}^{n-k-1} \alpha_{k,j} \bar{p}_{k+1} \sigma_j^{\{k+2,\ldots,n\}} + \sum_{j=1}^{n-k} \alpha_{k,j} p_{k+1} \sigma_{j-1}^{\{k+2,\ldots,n\}} \\
&\quad \text{(as a result of changing the variable for the second summation)} \\
&= \alpha_{k,0} \bar{p}_{k+1} \sigma_0^{\{k+2,\ldots,n\}} \qquad (4) \\
&\quad + \sum_{j=1}^{n-k-1} \alpha_{k,j} \bar{p}_{k+1} \sigma_j^{\{k+2,\ldots,n\}} + \sum_{j=1}^{n-k-1} \alpha_{k,j} p_{k+1} \sigma_{j-1}^{\{k+2,\ldots,n\}}
\end{aligned}$$

$$+\alpha_{k,n-k}p_{k+1}\sigma_{n-k-1}^{\{k+2,\ldots,n\}}$$

The term, $\sigma_0^{\{k+2,\ldots,n\}}$, expresses the probability that none of the propositions, $P_{k+2},\ldots,P_n$, is true. It is composed of only one product. Hence, the first term of 4 reduces to:

$$\begin{aligned}\alpha_{k,0}\bar{p}_{k+1}\sigma_0^{\{k+2,\ldots,n\}} &= \alpha_{k,0}\bar{p}_{k+1}\prod_{l=k+2}^{n}\bar{p}_l \\ &= \alpha_{k,0}\prod_{l=k+1}^{n}\bar{p}_l \\ &= \alpha_{k,0}\sigma_0^{\{k+1,\ldots,n\}} \qquad (5)\end{aligned}$$

Similarly, $\sigma_{n-k+1}^{\{k+2,\ldots,n\}}$, which expresses the probability that all $n-k+1$ of the propositions $\{P_{k+2},\ldots,P_n\}$ are true, is composed of only one product. Therefore, the last term of 4 reduces to:

$$\begin{aligned}\alpha_{k,n-k}p_{k+1}\sigma_{n-k+1}^{\{k+2,\ldots,n\}} &= \alpha_{k,n-k}p_{k+1}\prod_{l=k+2}^{n}p_l \\ &= \alpha_{k,n-k}\prod_{l=k+1}^{n}p_l \\ &= \alpha_{k,n-k}\sigma_{n-k}^{\{k+1,\ldots,n\}} \qquad (6)\end{aligned}$$

Combining the middle two terms of 4:

$$\begin{aligned}&\sum_{j=1}^{n-k-1}\alpha_{k,j}\bar{p}_{k+1}\sigma_j^{\{k+2,\ldots,n\}} + \sum_{j=1}^{n-k-1}\alpha_{k,j}p_{k+1}\sigma_{j-1}^{\{k+2,\ldots,n\}} \\ &= \sum_{j=1}^{n-k-1}\alpha_{k,j}[(\bar{p}_{k+1}\sigma_j^{\{k+2,\ldots,n\}}) + (p_{k+1}\sigma_{j-1}^{\{k+2,\ldots,n\}})] \\ &= \sum_{j=1}^{n-k-1}\alpha_{k,j}[(\bar{p}_{k+1}\sum_{\substack{R\subseteq\{k+2,\ldots,n\}\\|R|=j}}\pi_R^{\{k+2,\ldots,n\}}) + (p_{k+1}\sum_{\substack{R\subseteq\{k+2,\ldots,n\}\\|R|=j-1}}\pi_R^{\{k+2,\ldots,n\}})] \\ &= \sum_{j=1}^{n-k-1}\alpha_{k,j}[(\sum_{\substack{R\subseteq\{k+1,\ldots,n\}\\|R|=j}}\bar{p}_{k+1}\pi_R^{\{k+2,\ldots,n\}}) + (\sum_{\substack{R\subseteq\{k+1,\ldots,n\}\\|R|=j-1}}p_{k+1}\pi_R^{\{k+2,\ldots,n\}})] \\ &= \sum_{j=1}^{n-k-1}\alpha_{k,j}[\sum_{\substack{R\subseteq\{k+1,\ldots,n\}\\|R|=j\wedge k+1\notin R}}\pi_R^{\{k+1,\ldots,n\}} + \sum_{\substack{R\subseteq\{k+1,\ldots,n\}\\|R|=j\wedge k+1\in R}}\pi_R^{\{k+1,\ldots,n\}}] \\ &= \sum_{j=1}^{n-k-1}[\alpha_{k,j}\sum_{\substack{R\subseteq\{k+1,\ldots,n\}\\|R|=j}}\pi_R^{\{k+1,\ldots,n\}}] \\ &= \sum_{j=1}^{n-k-1}\alpha_{k,j}\sigma_j^{\{k+1,\ldots,n\}} \qquad (7)\end{aligned}$$

Finally, applying equations 5, 6, and 7 to equation 4, we have:

$$\begin{aligned}
&\sum_{j=0}^{n-k-1} \alpha_{k+1,j}\sigma_j^{\{k+2,\ldots,n\}} \\
&= \alpha_{k,0}\sigma_0^{\{k+1,\ldots,n\}} + [\sum_{j=1}^{n-k-1} \alpha_{k,j}\sigma_j^{\{k+1,\ldots,n\}}] + \alpha_{k,n-k}\sigma_{n-k}^{\{k+1,\ldots,n\}} \\
&= \sum_{j=0}^{n-k}\alpha_{k,j}\sigma_j^{\{k+1,\ldots,n\}} \\
&= pr(Q \text{ is true}) \qquad\qquad \text{(by the induction hypothesis)}
\end{aligned}$$

□

Setting $i = n$ in Lemma 1.4 immediately yields the following theorem which states that the PIC-EVAL algorithm is correct.

THEOREM 1.5. *Given that $\alpha_{n,0}$ has been produced by the* PIC-EVAL *algorithm:*

$$\alpha_{n,0} = pr(Q \textit{ is true})$$

2. PROOF OF LEMMA 4.3.1

LEMMA 4.3.1. *Given a* PIC *matrix whose coefficients,*

$$\alpha_m, \alpha_{m+1}, \alpha_{m+2}, \dots, \alpha_{m+s}$$

are of the form:

$$\alpha_m, \alpha_m + \Delta, \alpha_m + 2\Delta, \dots, \alpha_m + s\Delta$$

i.e., are such that:

$$\alpha_{m+j} = \alpha_m + j\Delta \quad \forall j = 0, ..., s$$

then, $\forall i = 0, \dots, s \quad \forall j = 0, \dots, s - i$*:*

$$\alpha_{i,m+j} = \alpha_m + j\Delta + \Delta \sum_{l=1}^{i} p_j$$

PROOF. By induction on i.
The base case follows directly from the hypothesis of the lemma.

$$\begin{aligned}\alpha_{0,m+j} = \alpha_{m+j} &= \alpha_m + j\Delta \\ &= \alpha_m + j\Delta + \Delta \sum_{l=1}^{0} p_j\end{aligned}$$

Assuming the lemma to be true for i=k-1:

$$\begin{aligned}
&\alpha_{k,m+j} \\
&= \alpha_{k-1,m+j}(1 - p_k) \qquad + \; \alpha_{k-1,m+j+1} p_k \\
&= (\alpha_m + j\Delta + \Delta \sum_{l=1}^{k-1} p_l)(1 - p_k) \; + \; (\alpha_m + (j+1)\Delta + \Delta \sum_{l=1}^{k-1} p_l) p_k \\
&= \alpha_m \; + \; (j\Delta + \Delta \sum_{l=1}^{k-1} p_l)(1 - p_k) \; + \; ((j+1)\Delta + \Delta \sum_{l=1}^{k-1} p_l) p_k \\
&= \alpha_m \; + \; j\Delta + (\Delta \sum_{l=1}^{k-1} p_l) - j\Delta p_k - (\Delta \sum_{l=1}^{k-1} p_l) p_k \\
&\qquad\quad + \; (j+1)\Delta p_k + (\Delta \sum_{l=1}^{k-1} p_l) p_k \\
&= \alpha_m \; + \; j\Delta + (\Delta \sum_{l=1}^{k-1} p_l) - j\Delta p_k + (j+1)\Delta p_k \\
&= \alpha_m \; + \; j\Delta + (\Delta \sum_{l=1}^{k-1} p_l) + \Delta p_k \\
&= \alpha_m \; + \; j\Delta + (\Delta \sum_{l=1}^{k} p_l) \qquad \square
\end{aligned}$$

ACKNOWLEDGMENT

The authors would like to express their gratitude to Leah Larkey whose original experimentation with *pnorm* operators in the context of the INQUERY inference network was the starting point for the experimentation reported here.