Computer Science Department Technical Report
University of California
Los Angeles, CA 90024-1596

APPROXIMATE ANSWERS THROUGH BAYESIAN BELIEF NETWORKS

B. Ribeiro
R. Muntz

July 1993
CSD-930025

Approximate Answers through Bayesian Belief Networks[*]

Berthier A.N. Ribeiro   Richard Muntz

Computer Science Department

University of California, Los Angeles

Los Angeles, California 90024

Abstract

Conventional database systems deal with exact queries which have to be met precisely. However, there are many applications that simply cannot perform well without some notion of *approximation* implemented by the database system. Examples include decision support systems, engineering applications, decision making systems, and image databases. Different approaches for providing approximate answers have been proposed in the literature [1, 2, 4, 7, 9]. We propose a new approach based on *Bayesian belief networks* to generate approximate answers. Belief networks provide a flexible framework that can be successfully used to merge approximation spaces, complex queries, and a ranking strategy in natural fashion. We compare our system with Motro's Vague system [7] and, through examples, show that our approximate answers agree better with human intuition.

1 Introduction

Conventional database systems deal with exact queries. An exact query specifies constraints that have to be met precisely. For instance, if a user asks for a film directed by Hitchcock that is being shown in Beverly Hills and there is no such film in the database, the user receives a null answer. However, it might be that there is a Hitchcock film in a theater located at Westwood which is only 5 miles

[*]Work supported in part by Organization of American States grant F29180 and NASA grant NAG 5-2225.

away. Further, the user has no mechanism to specify an approximate constraint such as *film-category similar-to suspense* in which case an adventure film might be satisfactory.

Since conventional database systems cannot handle approximate queries directly, these have to be emulated through a sequence of exact queries. But then, the user has to direct the system through a possibly long interaction to satisfy his needs. If the user is not aware of close alternatives, then even this solution is infeasible [7].

There are many applications that simply cannot perform well without some notion of approximation implemented by the database system. Cuppens and Demolombe [2] mention decision support systems and advice giving systems. Fuhr [4] mentions engineering applications, materials data systems, and business decision making systems. Rabitti and Savino [9] point to image databases.

Different approaches for providing approximate answers have been proposed in the literature. Cuppens and Demolombe [2] suggest the use of a knowledge based system implemented on top of a Prolog system extended by a meta level. Chu,Chen, and Lee [1] use a type abstraction hierarchy to form clusters of neighbor instances that can then be used to provide approximation. While the first has the resolution engine of Prolog to help with deducing inferences, the second provides a more structured approach that scales up better for large systems. However, none of them provide a ranking that reflects the *goodness* (i.e., proximity) of their answers with regard to the user query. Fuhr [4] enhances a database system with a probabilistic ranking formula borrowed from information retrieval. Rabitti and Savino [9] adopt another information retrieval strategy, the vector model, and apply it to image databases. While both approaches are able to provide a measure for the goodness of the approximate answers, the use of essentially empirical formulas raises questions. For example, by borrowing directly from information retrieval they fail to investigate under which conditions their adaptation is valid and/or useful. In addition, not having a more structured underpinning makes it difficult to extend and generalize these ideas.

Our approach also draws from information retrieval. However, we look to establish basic criteria desirable in a strategy for ranking approximate answers in a database. From fundamental assumptions and axioms, we aim at a new ranking formula.

Combining a ranking formula with an expert provided notion of proximity requires sound justification. Motro [7] proposes to solve this problem by normalizing the measures with a neighborhood radius and then combining them in a weighted sum of squares. While the approach does provide a solution, it lacks intuition. It also lacks a formal foundation. Yet, to the best of our knowledge, it is the work closest to ours.

Our approach relies on Bayesian belief networks [8]. Besides a solid formal basis, belief networks provide the epistemological foundation for combining different dependency relationships. They allow us to model the database, the expert quantification of proximity, neighborhood spaces for the instances, and complex queries; all under the same unifying framework. As discussed in section 7, our approach presents important advantages over the Vague query answering system [7].

The paper is organized as follows. Section 2 briefly introduces Bayesian belief networks. In section 3 we present our network model for databases. Section 4 discusses the derivation of a formula for ranking responses to queries from basic principles. This formula is used to complete the specification of our network model. Section 5 presents the incorporation of queries into the belief network. The concept of vague query conditions and its inclusion in the network is discussed in section 6 and section 7 illustrates our approach with an example. A comparision with the Vague approximate query answering system is given in section 8, followed by our conclusions.

2 Bayesian Belief Networks

In this section we briefly introduce *Bayesian belief networks* and discuss its fundamental properties. The exposition is entirely based on the presentation by Pearl [8].

Bayesian networks are DAGs in which the nodes represent random variables, the arcs portray causal relationships between the linked variables, and the strenghts of these influences are expressed by conditional probabilities. We proceed with a simple example.

Consider an experiment involving two dice and a bell. The dice are fair and independent. The bell

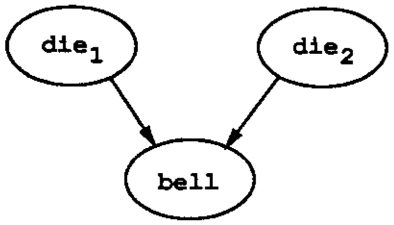

Figure 1: Bayesian belief network for a pair of dice and a bell.

rings when the outcomes of the dice coincide. The event that the *bell rings* is said to be *caused* by the event that the outcomes of the dice coincide. This behavior is properly modelled by the DAG in figure 1. If nothing is known about the outcome of the bell, the outcomes of the dice are *independent.* Suppose now that a spy observed an experiment and told us that the bell rang. In the face of this new piece of information (i.e., evidence), we can tell something about the outcomes of the dice even if the outcome is not completely determined. For instance, we immediately rule out events in which $outcome_1 \neq outcome_2$. Thus, the outcomes of the dice are no longer independent.

This behavior is referred to as an *induced dependency.* Induced dependencies are an important property present in many practical problems. Belief networks are able to distinguish induced dependencies from transitive dependencies (e.g., those established through an undirected path in the graph). For example, there is a path between the two dice in figure 1, but it is not a directed path. A path is said to be *blocked* if the *sink* nodes in it (e.g., those that participate in the path with two incoming links) are not instantiated. Instantiation of these sink nodes induces the dependency. In the language of belief networks, this is called the *d-separation criteria.* The main weakness of Markov networks, which use undirected graphs, is their inability to represent induced dependencies.

The DAG in figure 1 is an incomplete specification of the belief network for the dice-bell experiment. To complete the model it is necessary to quantify the strength of the dependencies between the two dice and the bell. This is done by specifying the probabilities $P(bell|die_1, die_2)$. For this simple case,

$$P(bell\ rings|die_1, die_2) = \begin{cases} 1 & \text{if } die_1 = die_2 \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

This completes the specification of the belief network for this example.

The *parents* of a node are those judged to be direct *causes* for it. The *roots* of the network are the nodes without parents. The set of parents of a node isolates the node from the rest of the network. This feature allows changing the strenghts of the link dependencies in a rather local fashion. One has to concentrate only on the relationship between a node and its parents.

When incoming evidence becomes available (e.g., the spy informing us of the outcome of the bell) the corresponding variables in the network are *instantiated.* This generates a belief in the state of the evidential node (i.e., the state of the bell node). In this case, the state of the bell node becomes *rang* with probability 1 (i.e., there is no doubt associated with the evidence). From the instantiated variables the new information is emanated to the rest of the network by a mechanism called *Bayesian belief propagation.* Section 7 illustrates the mechanism with an example.

Bayesian belief networks are a powerful and flexible modelling tool. Further, they have a solid formal foundation.

3 Belief Network Model for the Database

In this section, we discuss a model for representing a database as a belief network. Our interpretation is motivated by an analogy between a database and an information retrieval system.

In an information retrieval system, each document is described by a set of representative keywords called *index terms.* This set of index terms is usually converted into a vector as follows. If an index term i occurs in a document d and is representative of the document content (as decided by an expert or by some heuristic algorithm), then the *ith* position in the vector is set to 1, otherwise, it is set to 0. The vector for d provides an abstract and succinct description of the document. The information retrieval system uses these document vector descriptions for its retrieval and ranking operations. A user requests information by specifying a set of relevant index terms. These terms also form a vector – the query vector. The information retrieval system answers the user by searching for documents whose vectors *approximate* the query vector. Further, the system ranks the documents according to some measure of similarity between the document vectors and the query vector.

Ranking of the answers is better supported by assigning weights to the index terms [12, 13]. The weight of a term expresses its relative importance inside the document as well as its importance to the document in comparison with other documents. The resultant weighted document vectors are then compared for ranking purposes. A well accepted measure of proximity between weighted vectors is the cosine of the angle between them [12].

Turtle and Croft [15] propose a network model for an information retrieval system in which documents and index terms are modeled by binary random variables. As defined in their work, *a document variable corresponds to the event that a document has been observed.* Also, *an index term variable corresponds to the event that an index term has been assigned to a document.* An arc from a document node to an index term node indicates that the index term occurs in the document and has been recognized as a relevant index term (e.g., the document node is a *cause* for the index term node). This model presents, in our view, the following problems: (a) the event of observing a document is *not* well defined, (b) the event of assigning index terms to documents has no probabilistic meaning, and (c) most importantly, the *causal* relationship between document and index term variables is, at best, hard to grasp. We present an alternative below for databases that circumvents these problems.

The analogy that we establish with a database is as follows. A tuple in a database corresponds to a document vector in an information retrieval system. The attribute values in the tuple correspond to the index terms in the document vector. While the document vector summarizes a book in a collection, the tuple summarizes features of an object in the world. In an information retrieval system, however, there are no constraints on the index terms. In a database system an attribute can typically have only values from the attribute domain (i.e., a database is a *typed* system). The consequence is that when referring to a value we also have to refer to its type. This requires the introduction of the concept of an *attribute-value pair.*

An attribute-value pair (AVP) is a pair [<attribute>,<value>] stating that the attribute is equal to the value. For now, let a user query be a *conjunction* of AVPs. Thus, we can think of the experiment of a user selecting AVPs for a query. In this experiment, we assume that each AVP is equally likely to be selected by the user. Similarly to Turtle and Croft, we model each AVP (i.e., index term) by a binary random variable. However, the associated event is well characterized in our interpretation.

Once the user has specified the query, the system has to retrieve the *relevant* answers. In traditional relational databases, this means that tuples that satisfy all AVPs are retrieved. In this paper, *relevant* refers to the tuples that have some match of AVPs in common with the query. Thus, besides the traditional answers, the system also retrieves tuples that approximate the query [1, 2, 4, 7]. A tuple that matches the query exactly is *relevant*. However, a tuple that *nearly* matches the query is likely to be relevant. We want to formalize these notions in a database system which finds the candidate tuples and determines to which extent they are relevant. We model this relevance judgement by a binary random variable (relevant or non-relevant) assigned to each tuple. In section 4, we discuss how to estimate, using Bayesian belief networks, the probability that a tuple is relevant to a query, i.e. the probability that the associated binary variable is true. This probability is a measure of the strength of belief that the tuple is relevant and is the basis for ranking possible responses (for example, a user might ask for the *most relevant* answers).

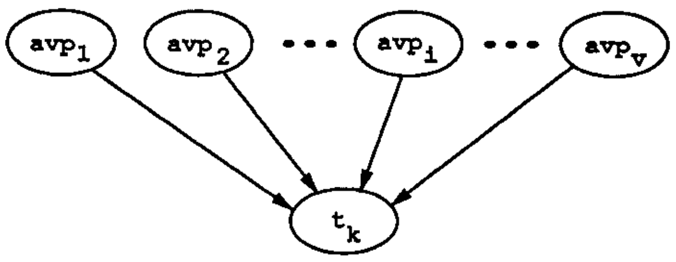

Figure 2: Bayesian belief network relative to tuple t_k only.

In our model, a relevance judgement is issued only after the user chooses a set of AVPs (e.g., submits a query). We interpret this temporal ordering to mean that the submission of a query *causes* the system to emit a relevance judgement for every tuple in the database[1]. Thus, our network model considers that the pattern of AVPs chosen by the user is a cause for the emission of relevance judgements by the

[1]Temporal ordering is the reason most often used to infer causation [5].

system. Notice that the causal directionality here is the *opposite* of that proposed by Turtle and Croft. Their work considers that documents (tuples) are the reason (i.e., the cause) for the existence of index terms (AVPs). While this relationship is true, it is not the focus of the querying process. The user is not worried about *how* and *why* index terms were associated to documents. This is the concern of the information system designer who, given a book, has to decide which are the representative index terms. The user reasons from the opposite viewpoint: given a set of index terms what books could possibly be relevant. Thus, specification of index terms (AVPs) is the reason (i.e., the cause) for deciding the relevance of a document (tuple). Our proposed *causality* directionality reflects the user querying process and is, in our view, more natural for the task of ranking relevant answers.

Figure 2 illustrates the corresponding fragment of a Bayesian belief network for a tuple t_k in the database. The set of parent nodes for t_k is the set $\{avp_1, avp_2, \ldots, avp_v\}$ of all v AVPs in the database. The reason is that the *relevance* to a query may depend on any AVP mentioned in the query (and not only on those in the tuple).

To complete our network model, we have to quantify the influence of the set of parent nodes on each tuple node. This requires the specification of the probability dependencies between a tuple node and its parents. Let,

$\vec{Q}$ $\rightarrow$ vector describing the state (*true,false*) of all the AVP nodes in the belief network according to query Q. For now, we assume that an AVP node is *true* if it was specified in the query.

$P(+t_k|\vec{Q})$ $\rightarrow$ probability that tuple t_k will be found relevant to query Q.

$P(-t_k|\vec{Q})$ $\rightarrow$ $1 - P(+t_k|\vec{Q})$

In the following section, we discuss the desired properties associated with these probabilities.

4 A Probability Ranking Formula for our Network Model

In this section we quantify the probability $P(+t_k|\vec{Q})$ that a tuple t_k is relevant to a query Q. We take this probability as the ranking of tuple t_k with regard to query Q. Our choice of a particular ranking formula is motivated by basic principles and empirical evidence borrowed from the field of information retrieval.

Given a query Q, for each tuple t_k we distinguish four vectors over the state space of all AVPs in the database: $\vec{x}_{Q\wedge t_k}$, $\vec{x}_{Q\wedge\overline{t_k}}$, $\vec{x}_{\overline{Q}\wedge t_k}$, and $\vec{x}_{\overline{Q}\wedge\overline{t_k}}$. Let g_i be a function that takes any of these vectors and returns the state of its *ith* AVP (avp_i). Then,

$g_i(\vec{x}_{Q\wedge t_k})$ is 1 if avp_i *true in* Q $\wedge$ avp_i *true in* t_k, otherwise it is 0;

$g_i(\vec{x}_{Q\wedge\overline{t_k}})$ is 1 if avp_i *true in* Q $\wedge$ avp_i *false in* t_k, otherwise it is 0;

$g_i(\vec{x}_{\overline{Q}\wedge t_k})$ is 1 if avp_i *false in* Q $\wedge$ avp_i *true in* t_k, otherwise it is 0;

$g_i(\vec{x}_{\overline{Q}\wedge\overline{t_k}})$ is 1 if avp_i *false in* Q $\wedge$ avp_i *false in* t_k, otherwise it is 0.

Further, let $\vec{x}_{t_k}$ and $\vec{x}_Q$ be vectors such that

$g_i(\vec{x}_{t_k})$ is 1 if avp_i *true in* t_k, otherwise it is 0.

$g_i(\vec{Q})$ is 1 if avp_i *true in* Q, otherwise it is 0.

The most accepted ranking strategy in information retrieval ranks documents (e.g., tuples in a database) by the cosine of the angle between the vectors $\vec{x}_{t_k}$ and $\vec{x}_Q$ [12]. The numerator in this cosine similarity formula (i.e., the *inner product* of the two vectors) is a function of $\vec{x}_{Q\wedge t_k}$, while the denominator is a function of $\vec{x}_{t_k}$ and $\vec{x}_Q$. However, $\vec{x}_Q$ is the same for all documents and can be regarded as a constant. Thus, the cosine similarity formula yields a ranking that is a function of $\vec{x}_{Q\wedge t_k}$ and $\vec{x}_{\overline{Q}\wedge t_k}$ only ($\vec{x}_{t_k} = \vec{x}_{Q\wedge t_k} + \vec{x}_{\overline{Q}\wedge t_k}$). The impact of $\vec{x}_{\overline{Q}\wedge t_k}$ on the ranking is to provide a normalization over the document space. This is important because different documents might be represented by index term vectors of rather disparate lengths.

In the world of databases all tuples have the same length and thus, $\vec{x}_{\overline{Q}\wedge t_k}$ does not have the same impact on the quality of the ranks. Further, database queries usually request information about a subset of the attributes in a tuple and not about the whole tuple. Thus, in general the database user worries about specific AVPs (those in the query) and not about the tuple as a whole. This discussion is summarized in the following:

Assumption 1 *Proper ranking of approximate answers in a database context must take into account $\vec{x}_{Q \wedge t_k}$ and $\vec{x}_{Q \wedge \overline{t_k}}$. Consideration of $\vec{x}_{\overline{Q} \wedge t_k}$ is not crucial. Contribution of $\vec{x}_{\overline{Q} \wedge \overline{t_k}}$ is irrelevant.*

We proceed from basic principles.

Axiom 1 *The ranking of a tuple t_k with regard to a query Q is an increasing monotonic function of the AVPs common to t_k and Q.*

$$R_{t_k|Q} \propto f_1(\vec{x}_{Q \wedge t_k}) \tag{2}$$

$$\vec{x}_{Q \wedge t_1} \subseteq \vec{x}_{Q \wedge t_2} \Rightarrow f_1(\vec{x}_{Q \wedge t_1}) \leq f_1(\vec{x}_{Q \wedge t_2}) \tag{3}$$

The intuition behind this axiom is that the ranking of a tuple should increase with the number of AVPs it has in common with the query increases.

Axiom 2 *The ranking of a tuple t_k with regard to a query Q is a decreasing monotonic function of the AVPs that appear in the query but do not appear in the tuple.*

$$R_{t_k|Q} \propto \frac{1}{f_2(\vec{x}_{Q \wedge \overline{t_k}})} \tag{4}$$

$$\vec{x}_{Q \wedge \overline{t_1}} \subseteq \vec{x}_{Q \wedge \overline{t_2}} \Rightarrow f_2(\vec{x}_{Q \wedge \overline{t_1}}) \leq f_2(\vec{x}_{Q \wedge \overline{t_2}}) \tag{5}$$

The intuition here is that a tuple that fails to match one or more AVPs in the query should be penalized. The penalty should increase with the number of missing AVPs.

Research in information retrieval shows that ranking formulas based on the frequency count of terms inside documents perform well [11, 12, 13]. Models based on a probabilistic interpretation of these frequencies of occurrence also perform well [4, 6, 16]. This empirical evidence motivates our next assumption.

Assumption 2 *Probability functions based on the frequency of occurrence of AVPs in the database are reasonable selections for f_1 and f_2. Thus,*

$$R_{t_k|Q} \propto \frac{f_1(P(\vec{x}_{Q\wedge t_k}))}{f_2(P(\vec{x}_{Q\wedge \overline{t_k}}))} \tag{6}$$

where $P(\vec{x}_{Q\wedge t_k})$ is the probability that vector $\vec{x}_{Q\wedge t_k}$ occurs among the tuples in the database. $P(\vec{x}_{Q\wedge \overline{t_k}})$ is defined analogously.

Equation 6 is too generic and allows an infinite number of ranking formulas. To narrow the possibilities, we search for additional intuition.

The best index term weighting schemas in information retrieval emphasize the selectivity of the index term [12]. Index terms that appear in many documents are not very discriminatory and therefore receive smaller weights. This idea concurs with the concept, introduced by Shannon [14], of *amount of information* generated by an event. Ross calls it *surprise* [10]. An index term that occurs infrequently *carries* more information (i.e., surprise) and should contribute with higher weight in the ranking computation. The concept of amount of information has interesting properties that agree with the behavior expected of a ranking formula. Our approach is to establish this relationship and to adapt it to our network model of the database.

Let f be a function variable for either f_1 or f_2. Let r be a variable for probabilities such as $P(\vec{x}_{Q\wedge t_k})$ and $P(\vec{x}_{Q\wedge \overline{t_k}})$. We associate f to the amount of information provided by events such as $\vec{x}_{Q\wedge t_k}$ and $\vec{x}_{Q\wedge \overline{t_k}}$.

Axiom 3

$$f(1) = 0 \tag{7}$$

The intuition is that an AVP that occurs in every tuple (e.g., with probability 1) provides no information.

Axiom 4 *$f(r)$ is a strictly decreasing function of r. Thus,*

$$r < s \Longrightarrow f(r) > f(s). \tag{8}$$

This axiom states that AVPs that are more discriminatory boost the value of f. This is a standard assumption in information retrieval systems.

Axiom 5 *$f(r)$ is a continuous function of r.*

This axiom states that small variations on r cause small changes on $f(r)$ – a desired property.

Axiom 6

$$f(r \times s) = f(r) + f(s) \tag{9}$$

This axiom states an independence assumption. Let $r = P(avp_i)$ be the probability that avp_i occurs in the database. The associated value for f is $f(r)$. Consider avp_j, another AVP distinct of avp_i, and let $s = P(avp_j)$. Further, assume that the event of avp_i occurring in the database is *independent* of the event of avp_j occurring in the database. Clearly, the value of f for the event of avp_i and avp_j occurring simultaneously is $f(r \times s)$. Thus, consideration of avp_j increases the value of f by $f(r \times s) - f(r)$. Since the occurrence of avp_j is assumed independent of the occurrence of avp_i, it has to be that $f(s) = f(r \times s) - f(r)$.

Theorem 1 *If a function f satisfies axioms 1 through 4, then*

$$f(r) = -C \log_2 r \tag{10}$$

where C is a positive integer.

Proof Refer to [10].

Years of research in the field of information retrieval solidified the use of the *inverse document frequency* as a good weighting factor for index terms [3, 12, 16]. The empirical formulation for the inverse document frequency factor has the same form as equation 10. Our interpretation is that the desirable properties of axioms 1 through 4 were intuitevely incorporated in the specification of index term weights, culminating in the final form displayed by (10). Our contribution here is to offer a more detailed , basic derivation of this result which exposes the assumptions made in the ranking formula. In addition, this formalization provides the basis for generalization to a formal treatment of approximate answers for database systems.

By fixing the proportionality constants to 1 and substituting (10) into (6), we obtain

$$R_{t_k|Q} = \frac{-\log_2 P(\vec{x}_{Q \wedge t_k})}{-\log_2 P(\vec{x}_{Q \wedge \overline{t_k}})} \tag{11}$$

Using the independence assumption of axiom 6, we are able to write

$$R_{t_k|Q} = \frac{\sum_{avp_i \ true \ in \ \vec{x}_{Q \wedge t_k}} -\log_2 P(avp_i)}{\sum_{avp_i \ true \ in \ \vec{x}_{Q \wedge \overline{t_k}}} -\log_2 P(avp_i)} \tag{12}$$

Let the number of tuples in which avp_i occurs be n_{avp_i} and the total number of tuples in the database be N. This yields

$$P(avp_i) = \frac{n_{avp_i}}{N} \tag{13}$$

Substituting into 12, we finally obtain

$$R_{t_k|Q} = \frac{\sum_{avp_i \ true \ in \ \vec{x}_{Q \wedge t_k}} -\log_2 \frac{n_{avp_i}}{N}}{\sum_{avp_i \ true \ in \ \vec{x}_{Q \wedge \overline{t_k}}} -\log_2 \frac{n_{avp_i}}{N}} \tag{14}$$

Now that we have derived the ranking formula, we return to the problem of quantifying $P(+t_k|\vec{Q})$, and thus $P(-t_k|\vec{Q}) = 1 - P(+t_k|\vec{Q})$, for our network model. Let $\vec{Q}_{set}$ be the set of all conjunctive[2] queries that the user might specify. A generic conjunctive query $\vec{Q}_g$ in this set is represented by the vector

$$\vec{Q}_g = (x_{avp_1}, x_{avp_2}, \ldots, x_{avp_v}) \tag{15}$$

$$\forall i, x_{avp_i} \in \{0, 1\} \tag{16}$$

where v is the total number of AVPs in the database. Thus, there are 2^v possibilities for $\vec{Q}_g$. To quantify the strength of the influence of $\vec{Q}_g$ on t_k, one can use any set of 2^v functions $F_i(t_k, \vec{Q}_g)$ satisfying

$$\sum_{t_k \ in \ \{true, false\}} F_i(t_k, \vec{Q}_g) = 1 \tag{17}$$

[2]In section 5, we relax this restriction and consider complex queries.

where $0 \leq F(t_k, \vec{Q}_g) \leq 1$. This specification is consistent and complete because the product form

$$P_r(t_k, x_{avp_1}, x_{avp_2}, \ldots, x_{avp_v}) = \prod_{\forall t_k, x_{avp_i}} F_i(t_k, \vec{Q}_g) \tag{18}$$

constitutes a joint probability distribution that supports the assessed quantities. For a thorough discussion of this matter refer to Pearl [8].

Since we take $P(+t_k|\vec{Q})$ to be the rank of tuple t_k in the network model, we make it proportional to $R_{t_k|Q}$ given by equation 14. The only required step is to normalize $R_{t_k|Q}$ so that it can be interpreted as a conditional probability and used in the Bayesian network. Thus,

$$\begin{aligned} P(+t_k|\vec{Q}) &= \frac{(\sum_{avp_i \ true \ in \ \vec{x}_{Q \wedge t_k}} - \log_2 \frac{n_{avp_i}}{N}) \div (\sum_{avp_i \ true \ in \ \vec{x}_{Q \wedge \overline{t_k}}} - \log_2 \frac{n_{avp_i}}{N})}{\sum_{avp_i \ true \ in \ \vec{x}_Q} - \log_2 \frac{n_{avp_i}}{N}} && (19) \\ P(-t_k|\vec{Q}) &= 1 - P(+t_k|\vec{Q}) && (20) \end{aligned}$$

This set of equations completes the specification of our network model for the database. $P(+t_k|\vec{Q})$ is such that it reflects all the desirable properties of a ranking formula as discussed in this section. In fact,

Theorem 2 *The expression for $P(+t_k|\vec{Q})$ in equation 19 satisfies all the axioms stated in this section. Further, all assumptions stated here apply to it.*

Proof Equation 19 differs from equation 14 by the sum over $\vec{x}_Q$ only. For a given query, this sum is the same for all tuples in the database. □

5 Belief Network Model for Queries

Until now, we have assumed that a query is simply a conjunction of AVPs. In this section, we extend a query to conjunctions and disjunctions of AVPs. As we are about to see, belief networks naturally extend themselves to handle complex queries.

As before, an AVP is modeled by a binary random variable avp_i. The associated experiment is the selection of AVPs by the user for inclusion in a query. Without prior information, we assume that the user is equally likely to select any of the AVPs.

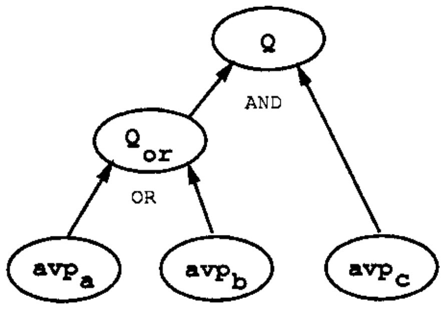

Figure 3: Bayesian belief network for query $Q = (avp_a \vee avp_b) \wedge avp_c$.

A query is a conjunction or disjunction of AVPs. It imposes a restriction on the set of AVPs that should be considered for retrieval (and ranking). A purely conjunctive query restricts the system to consider a unique combination of AVPs: those in the query are selected ($avp_i = true$), while those not mentioned are not selected ($avp_i = false$). A query that includes disjunctions of AVPs allows the system to consider more than one combination of AVPs. In this case, each combination of AVPs satisfying the query contributes to the final rank of a tuple t_k[3].

This behavior can be modeled in a belief network by representing a query by a binary random variable Q. $Q = true$ means that the query is satisfied, $Q = false$ means that the query is *not* satisfied. As suggested by Pearl [8], the links are directed from the AVP nodes to the query node.

Figure 3 illustrates the Bayesian belief network for the query $Q = (p_a \vee p_b) \wedge p_c$. The disjunction is modeled by the introduction of an auxiliary query node Q_{or}. Having the query node as sink establishes an *induced dependency* among variables p_a, p_b, and p_c. If nothing is known about node Q, the network states that AVPs p_a, p_b, and p_c are independent of each other (our system does not prefer one to another). If now the user specifies the query as input ($Q = true$), a dependency is induced among the three AVPs. For instance, our system will rule out choosing $(\neg p_a, \neg p_b, p_c)$ because it contradicts the new evidence (forcing p_a and p_b to false does not satisfy Q). Induced dependencies naturally model the action of a user inputting a query as new factual evidence (i.e., a new constraint). The impact of this

[3]Through Bayesian belief conditionalization and propagation, the network model combines all the evidence supplied by the query into a final rank for each tuple. The calculation is illustrated in section 7.

new evidence can then be propagated throughout the network resulting in new relevance judgements (e.g., rankings). Section 7 illustrates the approach with an example.

To complete this network model for queries we need to specify the link dependencies between a query node and its parent nodes. These dependencies are described by conventional logic since the conditions are deterministic. Let,

$P(+Q|\vec{x})$ $\rightarrow$ probability that query node Q is true given a state description of all the AVP nodes.

For the network in figure 3 this translates to

$$P(+Q_{or}|\vec{x}) = \begin{cases} 1 & \text{if } p_a \vee p_b \\ 0 & \text{otherwise} \end{cases} \tag{21}$$

$$P(+Q|\vec{x}) = \begin{cases} 1 & \text{if } Q_{or} \wedge p_c \\ 0 & \text{otherwise} \end{cases} \tag{22}$$

Notice that a user no longer selects the AVP nodes directly. Instead, a query is specified as new factual evidence. This evidence deterministically constraints combinations of AVPs that satisfy the query.

6 Belief Network Model for Vague Patterns

In the previous section, we have designed a ranking strategy from basic principles, discussed its properties, and shown how to model this strategy using a belief network. However, we have yet to address the key issue of this work – approximate answers. In this section we show how approximate or *vague* patterns can be naturally incorporated in a belief network.

Our definitions below are based on the work by Motro [7]. To be able to specify proximity or nearness, the user needs a *vague* operator. Motro calls this operator *similar-to*. We refer to it by either *similar-to* or $\sim$. Let a *vague* AVP be the triple [<attribute>,<value>,$\sim$]. This *vague* AVP is a reference to all AVPs that are similar-to the AVP [<attribute>,<value>]. The system searches for tuples containing these similar-to AVPs and provides a rank for these tuples. This requires quantifying the similarity between neighbor AVPs.

Measuring similarity requires the design of *data metrics.* A data metric is a function $M : D \times D \rightarrow R$ that specifies a *semantic* distance between any two AVPs in the same attribute domain. AVPs in different attribute domains are considered unrelated (i.e., separated by an infinite distance). In this paper, we do not concern ourselves with the design of data metrics for a database[4]. We simply assume that they have been specified by an expert.

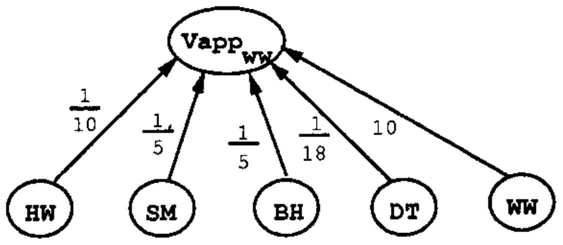

Figure 4: Belief network for the vague AVP $[location, Westwood, \sim]$.

A vague AVP establishes a dependency among similar AVPs only when it is mentioned in a user query. Before the specification of a vague AVP, no approximation can be used (all AVPs are considered independent). In the language of belief networks, we say that specification of a vague AVP *induces* a dependency among its neighbor AVPs. Thus, as done for queries, vague AVPs can be modeled by binary random variables representing factual evidence. We model the vague AVP corresponding to avp_i by the binary random variable $Vavp_i$. The AVPs that bear influence on $Vavp_i$ are those that are neighbors of avp_i according to the data metric.

Figure 4 illustrates the belief network corresponding to the vague AVP $[location, Westwood, \sim]$. $Vavp_{ww}$ is the associated random variable. Westwood (WW) is a region in Los Angeles. Hollywood (HW), Santa Monica (SM), Beverly Hills (BH), and Downtown (DT) are regions in the neighborhood of Westwood. The degree of influence of each region upon $Vavp_{ww}$ is taken here to be inversely proportional to the distance (from Westwood) to that region. The influence of Westwood upon $Vavp_{ww}$ tends

[4]Motro [7] discusses this issue in detail.

FILM					
Id	**Title**	**Director**	**Category**	**Theater**	**Location**
t_1	Four_Feathers	Korda	Adventure	Music_Hall	Beverly_Hills
t_2	Modern_Times	Chaplin	Comedy	Rialto	Downtown
t_3	Psycho	Hitchcock	Suspense	Chinese	Hollywood
t_4	Rear_Window	Hitchcock	Suspense	Egyptian	Westwood
t_5	Robbery	Yates	Suspense	Odeon	Sta_Monica
t_6	Star_Wars	Lucas	Adventure	Rialto	Downtown
t_7	Surf_Party	Dexter	Drama	Village	Westwood

Figure 5: A database for films.

to infinite. We model this fact by a weight that is an order of magnitude greater than any other.

The link matrix between $Vavp_{ww}$ and its parent nodes is specified by the normalized sum of the respective weights. For instance,

$$\begin{aligned} P(+Vavp_{ww}| + hw, +sm, -bh, -dt, -ww) &= \frac{\frac{1}{10}+\frac{1}{5}}{\frac{1}{10}+\frac{1}{5}+\frac{1}{5}+\frac{1}{18}+10} \\ P(-Vavp_{ww}| + hw, +sm, -bh, -dt, -ww) &= 1 - P(+Vavp_{ww}| + hw, +sm, -bh, -dt, -ww) \end{aligned}$$

All the other conditional link probabilities are calculated similarly.

7 Ranking Approximate Answers: an Example

In this section we show how belief networks can be used as an unifying framework for ranking approximate answers relative to a vague query. We use a simple conjunctive query here because it better suits our desire to expose clearly the advantages of our approach. However, there is no essential difference in the application of our ranking procedure to complex queries. The final ranking enjoys the properties established in section 4.

Figure 5 illustrates a database adapted from [7]. A film t_k has a title, a director, a category, a theater where it is in exhibition, and a location for that theater. The attribute t_i is not part of the original database. It is introduced here to facilitate reference to the tuples. Our example is based on the following vague query (also from [7]).

Q: **select** title,theater,category,location

from film

where category $\sim$ Adventure

and location $\sim$ Westwood

which asks for films whose categories are similar to *adventure* and that are been exhibited near to Westwood. Thus, this query involves two vague AVPs.

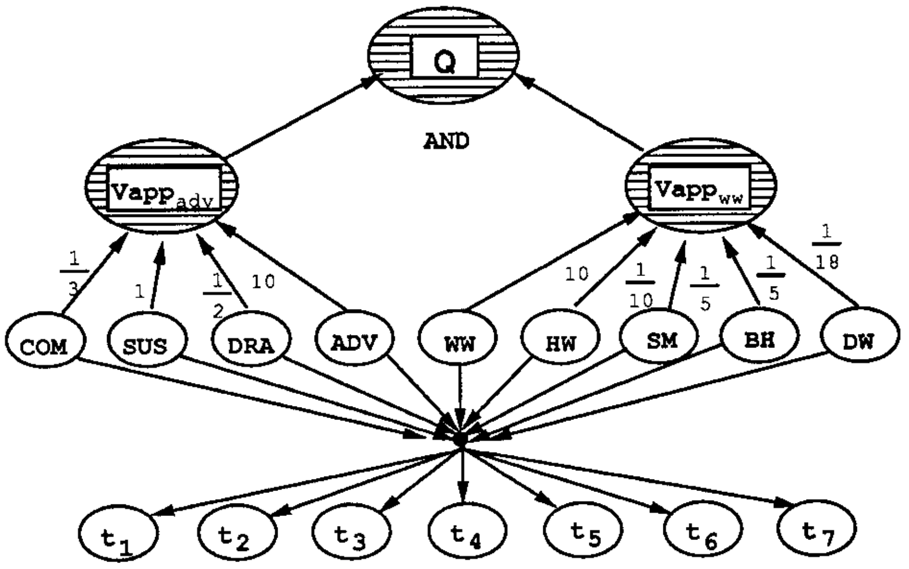

Figure 6: Bayesian belief network for our example query.

Figure 6 illustrates the complete network for our database and our example query. The acronyms stand for: *com* = comedy, *adv* = adventure, *sus* = suspense, and *dra* = drama. The others are as before. Only the links that can be possibly activated by Q (i.e., influence the ranking computation for any tuple) are shown[5]. The black node in the figure is a *switch* that connects any incoming links to all the outcome ones. It was introduced to simplify the drawing. Thus, the set of parent nodes of any

[5]The full network is a complete bipartite graph on the database side

tuple node t_k includes all the AVP nodes. The weights for the parent nodes of AVP $Vavp_{ww}$ are as specified in section 6. The weights for the parent nodes of AVP $Vavp_{adv}$ are taken as the inverse of the *semantic* distance between each parent node and $Vavp_{adv}$. These semantic distances are taken from the data metric given by Motro [7].

The AVP nodes (e.g., $\{com, sus, dra, \ldots, bh, dt\}$) are the *root* nodes in the network. They bear influence on other nodes but are not directly influenced by them (e.g., they have no incoming links). Before the user provides evidential information through a query, the AVPs are considered independent of each other. The boundary condition for a root node in a belief network is given by the prior probability of the root variable [8]. As before, we assume that a user has no preference to any AVP prior to specifying a query. This translates to a prior probability that is uniform over the space of AVPs.

Assuming that the prior probability distribution for AVPs is uniform does not mean that the user is unable to state preference. It means that prior to the manifestation of any preference, there is not much that can be said about the likelihood of an AVP being used in a query. The user manifestates preference when it specifies a query. The network acknowledges this preference by considering, for ranking purposes, only the combinations of AVPs that satisfy the query.

Input of query Q activates the vague AVP nodes $Vavp_{adv}$ and $Vavp_{ww}$. Given all the query evidence, what is the steady state probability (belief) forced upon each tuple node in the network? These beliefs are the corresponding rankings.

For our network model, the propagation of beliefs can be obtained by simple Bayesian conditioning. We proceed as follows.

$$P(+t_k| + Q) \quad = \quad P(+t_k| + Vavp_{ww}, +Vavp_{adv})$$

Let $\vec{x}$ be a vector describing the state of all AVP nodes (e.g., root nodes) in our network. Then,

$$\begin{aligned} P(+t_k| + Q) &= \sum_{\forall\, \vec{x}} P(+t_k| + Vavp_{ww}, +Vavp_{adv}, \vec{x}) \times P(\vec{x}| + Vavp_{ww}, +Vavp_{adv}) \\ &= \sum_{\forall\, \vec{x}} P(+t_k|\vec{x}) \times P(\vec{x}| + Vavp_{ww}, +Vavp_{adv}) \end{aligned}$$

because, given $\vec{x}$, the state of a tuple variable is independent of the other variables in the network. We again apply Bayes rule to obtain,

$$\begin{aligned}
P(+t_k|+Q) &= \sum_{\forall\,\vec{x}} P(+t_k|\vec{x}) \times \alpha \times P(+Vavp_{ww}, +Vavp_{adv}|\vec{x}) \times P(\vec{x}) \\
&= \sum_{\forall\,\vec{x}} P(+t_k|\vec{x}) \times \alpha \times P(+Vavp_{ww}|\vec{x}) \times P(+Vavp_{adv}|\vec{x}) \times P(\vec{x})
\end{aligned}$$

where α is simply a normalization constant and $P(\vec{x})$ is the *prior* probability (before any evidence is collected) for $\vec{x}$ [8].

The computation of $P(+t_k|+Q)$ is carried out as follows. $P(+Vavp_{ww}|\vec{x})$ and $P(+Vavp_{adv}|\vec{x})$ are calculated as exemplified in section 6. $P(\vec{x})$ is the prior probability for $\vec{x}$. Before any evidence is collected, all AVPs are independent of each other and uniformly distributed (they are the roots in the belief network). Thus, $P(\vec{x})$ can be trivially computed. $P(+t_k|\vec{x})$ is calculated using our ranking formula provided by equation 19. The computation has to cover the state space of $\vec{x}$ (2^9 in this example). Further details can be found in [8].

8 Comparision with Vague

In this section, we make a qualitative comparision between the ranking provided by our belief network model and the ranking provided by Vague [7]. The comparision is based on the example discussed in section 7. We can argue comparatively because the database, the data metrics used, and the query are the same for both systems.

Computation of the beliefs $P(+t_k|+Q)$ generates the following relevance ordering:

$$t_7 > t_1 > t_6 > t_4 > t_2 > t_5 > t_3 \tag{23}$$

We first observe that no tuple matches both query conditions (Westwood and Adventure). t_7, t_1, t_6, and t_4 are the tuples that match either one of these conditions. t_7 is considered most relevant because its approximate AVP match $[category, drama]$ is highly specific. There is only 1 drama movie in the database and that is provided by t_7. Thus, if the user is willing to accept a drama movie instead of an adventure he has no options other than to go with t_7. The network highlights this fact by boosting

t_7 in the ranking. Humans tend to do the same when confronted with many alternative choices that is, they concentrate on the more specific ones first. The approximate AVP match for t_1 is also very specific $[location, Beverly\ Hills]$, but t_1 is considered less relevant because the relative semantic distance between Beverly Hills (t_1) and Westwood is greater than the semantic distance between Drama (t_7) and Adventure. Tuple t_6 is preferred over t_4 because Downtown (t_6) is more specific than Suspense (t_4).

Intuitively, the network is ranking tuples as follows. It first detects the exact AVP matches and emphasizes those. For the approximate AVP matches, it first takes into account the amount of information provided by each pattern. Secondly, it considers the influence of the respective data metric. This balance between amount of information and semantic distance can be easily tilted to favor either one by proper manipulation of the data metric (i.e., preserving relative distance among neighbor AVPs). Thus, the network model provides a unifying framework for different ranking strategies that one might think of.

Vague ranking strategy generates the following relevance ordering:

$$t_1 = t_4 > t_5 > t_7 > t_3 > t_6 > t_2 \tag{24}$$

We first notice that t_7 is in the middle of the rank. This is so because Vague does not consider the associated amount of information. For the same reason, t_1 cannot be distinguished from t_4. Most surprising is to find t_5, a tuple without exact AVP matches, well positioned in the ranking. This happens because, without further information from the user (e.g., such as weights for the attributes), Vague heuristically combine distances between unrelated AVPs (category of the movie and location of the theater) by a square sum. Notice also that t_6, a tuple with an exact AVP match, ended up almost at the bottom of the ranking.

We find the network ranking more consistent and more in agreement with human intuition. It also provides a generic and flexible framework that is completely missing in Vague. Further, belief networks have a solid formal foundation.

9 Conclusions

Bayesian belief networks are a powerful modelling tool. Their flexibility and formal background make them suitable for application in many areas. To the best of our knowledge, Turtle and Croft were the first to use it in the field of information retrieval. We have investigated the use of belief networks for approximate answers in databases.

We have pointed out our disagreements with the Turtle and Croft model and have suggested a viable alternative that, we understand, is more promising. Their model reflects the frame of mind of the system designer. Ours models the user querying process and intentions. The latter is the important process for a system that seeks to produce *relevant* answers to approximate queries.

We based our ranking strategy on empirical results borrowed from the field of information retrieval. The idea was to establish an analogy between an information retrieval system and a database. We avoided using information retrieval ranking strategies directly for two reasons: (a) they have an excessively empirical nature and (b) index vectors for documents might have rather disparate lengths while tuples have vectors of fixed length. Our analogy led to the desired, but yet general, form displayed in equation 6. To further constrain our choice for a ranking formula, we drew from information theory. By axiomatization of desired properties, we were able to constrain the form of our ranking formula what led to the result in equation 14. Further, we provided a more detailed, basic derivation for the *inverse document frequency*, a common metric in the field of information retrieval.

We discussed how to use *induced dependencies* in a Bayesian belief network to model vague AVPs and generate approximate answers. The flexibility of the belief network allowed us to combine Motro's data metrics with our ranking formula in a consistent fashion. Further, we could easily extend the approach to deal with complex queries, an issue that is not dealt with properly by Motro's Vague system[6].

We compared the results of our strategy with those generated by the Vague system through a simple example extracted from Motro's paper [7]. Even this simple example was enough to highlight

[6]We did not explore this in the paper.

the advantages of our approach. While Vague merges the contributions of distinct approximate query conditions through a simple weighted sum of squares, our system uses Bayesian conditionalization what preserves the independencies stated by the network. Further, our formula takes into account the amount of information provided by each vague condition what improved the quality of the rank considerably.

References

[1] Wesley W. Chu, Qiming Chen, and Rei chi Lee. Cooperative query answering via type abstraction hierarchy. In *S.M. Deen, editor, Cooperating Knowledge Based Systems*, North Holland, 1991. Elsevier Science Publishing Co.

[2] F. Cuppens and R. Demolombe. Cooperative answering: a methodology to provide intelligent access to databases. In *Second International Conference on Expert Database Systems*, Virginia, U.S.A., 1988.

[3] W.B. Frakes and R. Baeza-Yates, editors. *Information Retrieval: Data Structures & Algorithms.* Prentice Hall, 1992.

[4] Norbert Fuhr. A probabilistic framework for vague queries and imprecise information in databases. In *Proceedings of 16th VLDB Conference*, Brisbane, Australia, 1990.

[5] Dan Geiger, Azaria Paz, and Judea Pearl. Learning simple causal structures. *International Journal of Intelligent Systems*, 8:231–247, 1993.

[6] K.L. Kwock. Experiments with a component theory of probabilistic information retrieval based on single terms as document components. *ACM Transactions on Information Systems*, 8(4):363–386, October 1990.

[7] Amihai Motro. Vague: A user interface to relational databases that permits vague queries. *ACM Transactions on Office Information Systems*, 6(3):187–214, July 1988.

[8] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann Publishers, Inc., 1988.

[9] F. Rabitti and P. Savino. An information retrieval approach for image databases. In *Proceedings of 18th VLDB Conference*, Vancouver, British Columbia, Canada, 1992.

[10] S. Ross. *A First Course In Probability.* Macmillan Publishing Company, 3rd edition, 1988. Chapter 9.

[11] G. Salton. *The Smart Retrieval System – Experiments in Automatic Document Retrieval.* Prentice Hall Inc., Englewood Cliffs, NJ, 1971.

[12] G. Salton and C. Buckley. Term-weighting approaches in automatic retrieval. *Information Processing & Management*, 24(5):513–523, 1988.

[13] G. Salton and M.J. McGill. *Introduction to Modern Information Retrieval.* McGraw-Hill Book Co., New York, 1983.

[14] C.E. Shannon. *The Mathematical Theory of Communication.* The University of Illinois Press, Urbana, U.S.A., 1949.

[15] H. Turtle and W. B. Croft. Evaluation of an inference network-based retrieval model. *ACM Transactions on Information Systems*, 9(3):187–222, July 1991.

[16] C.J. van Rijsbergen. *Information Retrieval.* Butterwords, 1979.