Probabilistic Ranking for Relational Databases based on Correlations

Jaehui Park
School of Computer Science and Engineering
Seoul National University
Seoul 151-742, Republic of Korea
+82-2-880-5133
jaehui@europa.snu.ac.kr

Sang-goo Lee
School of Computer Science and Engineering
Seoul National University
Seoul 151-742, Republic of Korea
+82-2-880-5517
sglee@europa.snu.ac.kr

ABSTRACT

This paper proposes a ranking method to exploit statistical correlations among pairs of attribute values in relational databases. For a given query, the correlations of the query are aggregated with each of the attribute values in a tuple to estimate the relevance of that tuple to the query. We extend Bayesian network models to provide a probabilistic ranking function based on a limited assumption of value independence. Experimental results show that our model improves the retrieval effectiveness on real datasets and has a reasonable query processing time compared to related work.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information search and retrieval; H.2.8 **[Database Applications]**: Miscellaneous

General Terms

Algorithms, Experimentation

Keywords

Ranking, Keyword Search over Structured Data, Correlation, Attribute value, Bayesian Networks

1. INTRODUCTION

Recent ranked retrieval approaches in relational databases have proposed many different ranking functions for their purposes. The commonly used ranking schemes (such as, *tf*×*idf* or *BM25*) are based on, no matter how complex, repeated application of *term frequency* over a given tuple and *document frequency* over a collection. We conjecture that simply counting the repeating groups of terms hardly represent the importance of the tuple. The adaptation of *tf*×*idf* schemes in structured data cannot effectively work in many cases, especially for databases with categorical attributes (e.g., e-catalogs). For example, consider a car instance that has a term *Ford* for the attribute *Make* and for the attribute *Seller's name* at the same time. In this case, we have to consider the term *Ford* to have two different semantics, which are a car maker and a person's name. In an explicit feature space, such as in relational databases, we should differentiate between the semantics of the term *Ford* in two different categories of characteristics to interpret the different topics (i.e., information needs).

In this paper, we propose an effective ranking method for relational databases. Although rich dependency structures are explicitly modeled in relational databases, the term frequency-based schemes ignore them for practical reasons. We have to look beyond the query terms because the Boolean query results cannot distinguish the results based on term frequency. Rather than using the frequencies of a given term (as a query) within a tuple, we exploit the *correlations* of the term with other terms in the same tuple to estimate the significance of the term in the tuple. Statistical relationships, such as correlations, provide a useful understanding of the semantics inside data collections. Because real world databases typically exhibit strong correlations between attribute values and reside in a relatively dense and low dimensional feature space compared to document collections, the correlations are easily identified and are effective for ranking relational data. By considering the correlations with other values in the same tuple, we can indirectly estimate the significance of given terms. Hence, we have to model dependencies between terms, which require high computational cost. By assuming a limited independence on term sets, we reduce the computational cost and still maintain the effectiveness of the ranking function. Furthermore, we extend the Bayesian network model to provide a probabilistic ranking function based on the limited assumption of value independence. The Bayesian network model describes the possibilities of correlations between terms by using a probabilistic approach. We evaluated and validated our ranking method through experimentation using real datasets extracted from commercial web databases. We compared our method with the *tf*×*idf* scheme-based approach [1]. The experimental results show that our method yields a higher quality of retrieval.

The rest of the paper is organized as follows: In Section 2, we describe an intuitive example of correlation based weighting scheme. In Section 3, we introduce a probabilistic ranking model based on attribute value correlation with a limited assumption of value independence. In Section 4, we present experimental results for evaluating our ranking method on real datasets. Finally, we present concluding remarks in Section 5.

2. ATTRIBUTE VALUE CORRELATION

In this section, we introduce the concepts of the attribute value as a basis for modeling the correlation in relational databases.

2.1 Attribute Values

We present two types of attribute values: specified attribute values and unspecified attribute values.

Table 1. A part of a table with correlated values

	WD	Brakes	Doors	Color
t1	AWD	Power Brakes	3 Doors	Silver
t2	AWD	Power Brakes	3 Doors	Black
t3	AWD	Power Brakes	3 Doors	Black
t4	AWD	Power Brakes	3 Doors	Black
t5	AWD	Power Brakes	3 Doors	Green
t6	AWD	Power Brakes	2 Doors	Grey

Definition 1. Let $K = \{k_1, k_2, ..., k_n\}$ be the ***terms*** of a collection R of tuples, that is, the set of n unique terms that appear in all tuples in R.

Definition 2. Let $V = \{v_1, v_2, ..., v_n\}$ be the ***attribute values*** of a collection R of tuples, that is, the set of n unique attribute values that appear in all tuples in R. We assume that a term k_i in K is modeled as an attribute value v_i; there exists a one-to-one matching between k_i and v_i in R.

Definition 3. A ***query*** Q, $Q \subseteq V$, is a set of l attribute values. $|Q|$ denotes the number of query terms. A ***tuple*** T, $T \subseteq V$, is a set of m attribute values. $|T|$ denotes the number of attribute. An attribute value av of a tuple t binding to an attribute a is denoted as $av=t[a]$. R is a set of tuples, $R=\{T_Q^1, T_Q^2, ..., T_Q^{|R|}\}$ and $Q \subseteq T_Q^i$, which contains all of the query terms.

Definition 4. Given query Q, let $SV_i = \{sv_{i1}, sv_{i2}, ..., sv_{i|Q|}\}$, $SV_i \subseteq T_Q^i$, be the set of unique attribute values that appear in the query Q and the tuples T_Q^i. We define this set as the ***specified attribute values***. Let $UV_i = \{uv_{i1}, uv_{i2}, ..., uv_{i|T|-|Q|}\}$, $UV_i \subseteq T_Q^i$, be the set of unique attribute values that appear in the tuples T_Q^i but not in the query Q. We define this set as the ***unspecified attribute values***. Two sets, $SV_i \cap UV_i = \emptyset$, are disjoint, and the union of SV_i and UV_i forms a tuple T_Q^i.

2.2 Illustrative Example

To quantify the usefulness of terms in characterizing the semantics of the tuple in which they appear, each attribute value matching the terms in a tuple is assigned a weight based on its significance for the tuple, given a query. We suggest using the *correlations* between the specified attribute values and the unspecified attributes to derive the weight of the attribute values specified by the queries. A key feature of our approach is that we can easily extract the correlations from the statistics of relational databases. The basic intuition is that semantically related terms often occur close to each other.

Example 1. Suppose that we have a set of tuples and wish to determine the weight of an attribute value for a tuple. For example, let us consider Table 1 as the Boolean query results for the query Q={'AWD'}. The value *AWD* is considered as a specified attribute value SV_i={'AWD'} for tuple t_i in Table 2. We estimate the weight of the specified attribute value SV_i based on the correlations with unspecified attribute values UV_i. Table 2 illustrates an extreme case of statistical closeness, such as the correlations between *AWD* and *Power Brakes*$\in UV_i$. These values are heavily correlated and can be exploited to weight the specified attribute value *AWD*: the weight of the attribute value *AWD* is estimated as high for a tuple that contains *Power Brakes*. The weight of a specified attribute value for a given tuple is increased by the correlations of unspecified attribute values. As an easy illustration for computing the correlation, we can naively count the of common tuples t_i to estimate correlations (similar to *support* measure in association rule mining). The correlations of the unspecified attribute values *Power Brakes*$\in UV_{1\sim6}$, *3 Doors*$\in UV_{1\sim5}$ and *2 Doors*$\in UV_6$ are *6*, *5* and *1*, respectively. Based on these correlations, the weight of AWD for t_1~t_5 is estimated higher than that for t_6.

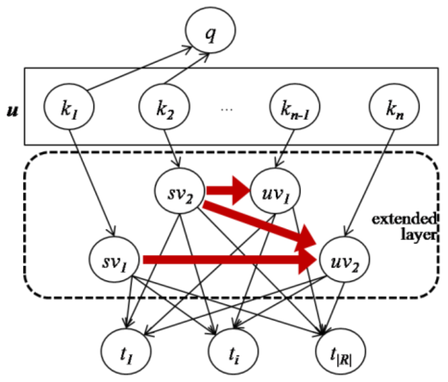

Figure 1. Correlation modeling in the Bayesian network.

2.3 Measures of the Correlations

Originally, correlations represent statistical relationships between random variables or observed data values. We consider relational databases as collections of random variables at a specific point in a search space.

Definition 5. Let V be a discrete random variable on a finite set $V = \{v_1, v_2, ..., v_n\}$, with probability distribution function $p(v_i) = Pr(V=v_i)$. If the observations of $V=v_i$ are not independent of the observations of $V=v_j$, then we say that the two attribute values v_i and v_j, have a correlation. Although correlation does not imply causation, we can establish a causal relationship in the correlation by conditioning the prior event, the observation of the query Q. $Pr(V=v_j|V=v_i,Q)$ describes the extent of the correlation of the unspecified attribute value v_j with the specified attribute value v_i given by the query Q.

There are several correlation coefficients that measure the degree of correlation. Many of them have essential resemblances, so we adopt a representative measure: the Pearson correlation coefficient. Because it has been well studied in the fields of statistics, we can simply interpret the measurements of all possible outcomes pointwise from the expected value of the covariance for categorical values.

3. PROBABILISTIC RANKING MODEL

3.1 Problem Formulation

Consider a database table R with tuples $T = \{t_1, t_2, ..., t_{|R|}\}$ with attributes $A = \{a_1, a_2, ..., a_m\}$ and a query $Q = \{k_1, k_2, ..., k_{|Q|}\}$ over R with a parameterized query of the form "SELECT * FROM R WHERE $a_1 = k_1$ AND $a_2 = k_2$ … AND $a_{|Q|} = k_{|Q|}$", where each a_i is the attribute from A corresponding to the specified attribute value k_i specified by the query terms ($|Q|<m$). The set of attributes $A_s = \{sa_1, sa_2, ..., sa_{|Q|}\} \subseteq A$ is called the set of specified attributes by the query terms. Analogously, the set of attributes $A_u = \{ua_1, ua_2, ..., ua_{m-|Q|}\} \subseteq A$ is called the set of unspecified attributes. The goal of this work is to develop a score $S(Q,t)$ for each tuple t that captures the relevance of the tuple t and the query Q. To compute the score, we mine the attribute value

correlations $C(sv,uv)$ for the given query, which is an specified attribute value $sv=t[sa_i]$ ($uv=t[ua_i]$ denotes an unspecified attribute value). Now a simple scoring function can be defined by the following formula:

$$S(Q,t) = \prod_{\forall sa_i \in A_s, \forall ua_i \in A_u} D_{sa_i} \times C(t[sa_i], t[ua_i]). \quad (4)$$

where D_{sa_i} denotes the weight for a given attribute sa_i.

3.2 Extending the Bayesian Network Model with the Limited Assumption of Value Dependency

Figure 1 illustrates the extended Bayesian network model for our purposes. A database is represented by the set of the tuples t_i. The nodes sv_i and uv_j represent two types of attribute values: the specified attribute values and the unspecified attribute values. Each node k_i represents a term in the concept space U. The universe of discourse U, which we take as our sample space, is a set of elementary concepts u in the space U. The node q represents the query that corresponds to terms in the concept u.

First of all, we define an extended layer for attribute values to allow dependencies between them. The specification of the probability $P(t|q)$ needs to consider the dependency. Traditionally, most practical retrieval models assume the following: given a query Q and a tuple t, dependencies between the terms are not allowed. Although this assumption is unrealistic in many cases, the assumption reduces computational costs. For example, given the sample space U, we can have 2^t concepts (subsets of U). If all concepts are considered to be equally likely a priori, each prior probability $P(u)$ is set to $P(u)=(1/2)^t$, which is a constant. Without this assumption, the dependencies among elementary nodes have to be specified by expanding $P(u)$ to $P(u|u')P(u'|u'')$ … . Our preliminary implementations suggested that such approaches have unacceptable preprocessing and query processing costs to calculate the probabilities. Consequently, we define a limited independence assumption only on the extended layer.

Limited Value Independence Assumption. Given a query q and a tuple t, the specified attribute values are assumed to be mutually independent. Analogously, the unspecified attribute values are assumed to be mutually independent. We allow dependencies between a specified attribute value and an unspecified attribute value. Therefore, $p(sv,uv)$ is not equal to $p(sv)p(uv)$ if there exists a correlation between them.

By allowing dependencies (thick arrows in Figure 1) in only a limited set of nodes, we can avoid the high cost of calculating the prior probabilities. Still, the dependencies approximate the probability of relevance that is proved to be significant for the quality of ranking. Of course, the full utilization of dependencies among unspecified attribute values as well as specified attribute values is complete to derive the relevance of the tuple. However, our assumption is a proper trade off. Based on *Definition 5*, the probability of the relevance between the query Q and the tuple t_i can be derived as follows:

$$P(t_i|q) = \frac{1}{P(q)} \sum_{u \in U} P(t_i|u)P(q|u)P(u) \quad (5)$$

$$= \frac{1}{P(q)} \sum_{u \in U} P(SV_i, UV_i|u)P(q|u)P(u)$$

$$= \frac{1}{P(q)} \sum_{u \in U} P(SV_i|u)P(UV_i|SV_i, u)P(q|u)P(u).$$

Based on *Definition 2*, edges from the term k_i only exist if k_i is observed for a given query and given attribute values. This implies that the term k_i is conditionally independent, given the query q. Therefore, u can be divided into two disjoint concepts u_q and $u_x=u-u_q$ corresponding to the attribute values sv and uv, respectively. The conditionally independent concepts u_x (and u_q) for corresponding sv_i (and uv_i) is removed because the value is reducible. Additionally, $P(q)$ and $P(u)$ can be considered as constants. The final ranking function is as follows:

$$P(t_i|q) \propto \sum_{u \in U} P(SV_i|u_q)P(UV_i|SV_i, u_x)P(q|u_q). \quad (6)$$

Therefore, to infer the probability of a tuple t_i given a query q, we need to specify the following probabilities: $P(SV_i|u_q)$, $P(UV_i|SV_i,u_x)$ and $P(q|u_q)$

The specifications of the probabilities determine the final ranking function for our method.

$$P(SV_i|u_q) = \prod_{\forall sv_{il} \in SV_i} P(sv_{il}|u_q). \quad (7)$$

$$P(UV_i|SV_i, u_x) = \prod_{\forall sv_{il} \in SV_i, \forall uv_{iw} \in UV_i} D \times C(sv_{il}, uv_{iw}). \quad (8)$$

$$P(q|u_q) = \begin{cases} 1 & if\ \forall k_i, \quad I_q(k_i) = I_{u_q}(k_i) \\ 0 & otherwise \end{cases}. \quad (9)$$

$P(SV_i|u_q)$ is specified based on the mutual independence among the specified attribute values. Optionally, we can flexibly adapt the *tf×idf* schemes for cases in which a group of repeating terms is present in a tuple. We may specify the probability $P(SV_i|u_q)$ as a similarity function such as the cosine similarity. In our experiment, we set this probability to unity, which corresponds to Boolean matching. $P(UV_i|SV_i,u_x)$ is defined as the aggregation of the attribute value correlations C. We use $P(q|uq)$ to select the concept u_q that matches the query. The function $I_q(k)$ is 1 if $k \in q$ and 0 otherwise. This equation guarantees that the states where the active values u_q are those of the query q are taken into consideration. By applying Equations 7, 8 and 9 to Equation 6, we obtain a ranking equivalent to the one found with Equation 4. The score for tuple t, given the query Q, is defined by the probability $p(t|q)$. Our probabilistic ranking model is subsumed by the Bayesian network model, which is flexible enough to represent evidential information, such as attribute value correlations.

4. EXPERIMENTAL EVALUATION

4.1 Experimental Setup

In our evaluation, we use a used car database *CarDB* (*Make, Model, Year, Color, Style, Price, Mileage, Location*) containing 158351 tuples extracted from a web database, *Yahoo! Autos* [2]. A MySQL Server 5.0 RDBMS is used on an AMD Athlon 64 processor 3.2 GHz PC with 2GB of RAM for the environment of the experiment. All of the algorithms are implemented in JAVA, connected to the RDBMS through JDBC. To efficiently identify the top-k results with the highest scores, we materialize the correlations between the attribute values in the auxiliary index table at preprocessing time. We proactively identify all of the correlations between attribute values using an SQL query

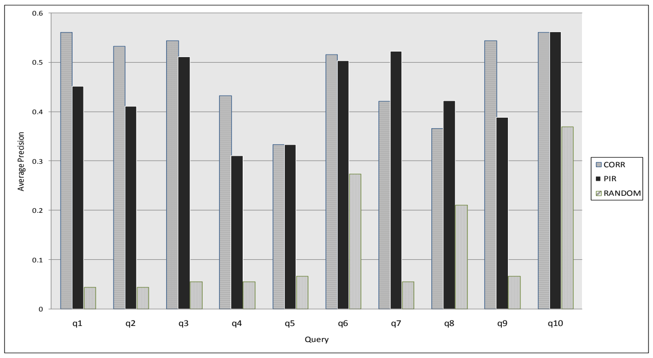

Figure 2. Average precision

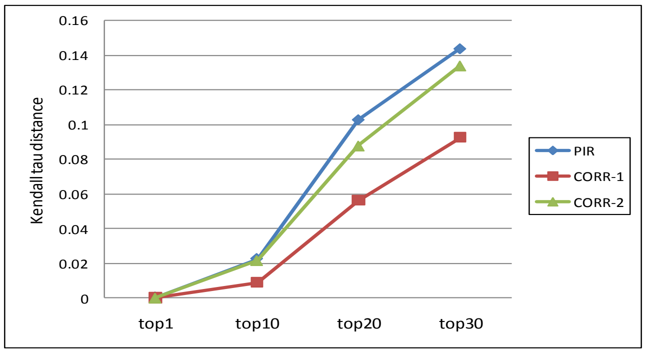

Figure 3. Kendall tau distance (cut-off: 1, 10, 20 and 30)

interface. The query interface computes all pair-wise correlation by single table scan and stores the results in the auxiliary table. We devise an inverted index to maintain every pair of attribute values mapping to tuples in the database with the correlation values.

Two other ranking methods are implemented for comparison with our method. For simple presentation, we refer our method as *CORR* (*CORelation-based Rank*). RANDOM method present the query results in random order. PIR [1] method is one of the state-of-the-art method in the related work. We solicited 10 test queries that represent a heterogeneous mix of different profiles of positional car buyers as we did not wish our queries to be biased. We conducted a comparative study for the top-10 results for average precision. The Kendall tau distance is measured by comparing the counts of swaps required to place the list of one ranking method in the same order as the answer list ordered by the candidates.

4.2 Retrieval Effectiveness

The retrieval effectiveness of CORR is evaluated by two metrics: *average precision* and *Kendall tau distance*. These two metrics are used to evaluate how well the user's intention is captured with the different ranking methods. The precision is the ratio of the number of relevant tuples retrieved to the total number of tuples retrieved. The precision measures the number of tuples that are either relevant or non-relevant. The Kendall tau distance is provided to measure the pairwise disagreements between two top-*k* ranked lists of tuples.

The evaluation results show that CORR generally produces rankings of higher quality compared to the RANDOM method for every query. For most of the queries, there are some overlaps between the results of CORR and that of PIR. The precision of CORR is 0.1 higher than that of the PIR method on average. However, the query results for q7 and q8 have lower in precision compared to the PIR method. The reason for this phenomenon was that some terms that were very commonly surfaced in the query workloads, but these terms were not popular terms for car sales databases. Because our method only focuses on the given database statistics, the vocabulary not appearing in the database cannot be used as evidences. For eight of ten queries, the precision of CORR is 0.15 higher than that of the PIR method on average. Figure 3 shows the Kendall tau distance of the different ranking methods for each top-*k* query. It shows that CORR-1 generates the most relevant tuple ranking for any *k* threshold. We emphasize that even the primitive correlation metrics-based method (e.g., co-occurrences) performs better than the *tf*×*idf* scheme-based method, PIR. CORR-1 and CORR-2 consistently perform better at the top lists. While the results of these user studies appear promising, we caution that it would be premature to interpret the results as conclusive evidence at all cases. From the above results, we find that our basic premise, that the statistical relationships between attribute values are effective for ranking tuples, is reasonable.

5. ACKNOWLEDGMENTS

This research was supported by the MKE(The Ministry of Knowledge Economy), Korea, under the ITRC(Information Technology Research Center) support program supervised by the NIPA(National IT Industry Promotion Agency). (grant number NIPA-2010-C1090-1031-0002)

6. CONCLUSION

In this paper, we introduced a probabilistic method to enable an effective retrieval over relational databases with categorical attributes by analyzing the attribute value correlation. Our ranking function is based on the Bayesian network model, explicitly introducing the limited independence assumption on the set of attribute values to avoid high computational costs. We presented a set of results from experiments conducted on a real dataset. These experiments show that our ranking strategy is effective without a priori knowledge.

In this work, we provided a crude query processing model that could handle simple top-*k* query in a given context. Further research is required to develop the efficient top-*k* computation of the probability that represents the correlation between attribute. Many contents including related work and experimental results are omitted due to the space limit in this paper.

7. REFERENCES

[1] Chaudhuri, S., Das, G., Hristidis, V., and Weikum, G. 2004. Probabilistic ranking of database query results. In *Proceedings of the Thirtieth international Conference on Very Large Data Bases - Volume 30* (Toronto, Canada, August 31 - September 03, 2004). M. A. Nascimento, M. T. Özsu, D. Kossmann, R. J. Miller, J. A. Blakeley, and K. B. Schiefer, Eds. Very Large Data Bases. VLDB Endowment, 888-899.

[2] Yahoo! Autos, http://autos.yahoo.com