Probabilistic Models for Query Approximation with Large Sparse Binary Data Sets

Technical Report No. 00–07
Department of Information and Computer Science
University of California, Irvine

Dmitry Pavlov
Information and Computer Science
University of California, Irvine
CA 92697-3425
pavlovd@ics.uci.edu

Heikki Mannila
Nokia Research Center
P.O. Box 407
FIN-00045 Nokia Group, Finland
Heikki.Mannila@nokia.com

Padhraic Smyth
Information and Computer Science
University of California, Irvine
CA 92697-3425
smyth@ics.uci.edu

February 2000

Abstract

Large sparse sets of binary transaction data with millions of records and thousands of attributes occur in various domains: customers purchasing products, users visiting web pages, and documents containing words are just three typical examples. Real-time query selectivity estimation (the problem of estimating the number of rows in the data satisfying a given predicate) is an important task for such databases.

We investigate the application of probabilistic models to this problem. In particular, we investigate probabilistic models based on frequent sets and maximum entropy, and compare such models to the independence model and the Chow-Liu tree model. We find that the maximum entropy model provides substantially more accurate probability estimates than the other methods but is more expensive from a computational and memory viewpoint. To alleviate the computational requirements we show how one can apply bucket elimination and clique tree approaches to take advantage of structure in the models and in the queries. We provide experimental results on two large real-world transaction datasets.

1 Introduction

Massive datasets containing huge numbers of records have recently become an object of increasing interest among both the businesses who routinely collect such data and data miners who try to find regularities in them. One class of such datasets is binary transaction data. This class is typically characterized by high *sparseness*, i.e. there might be hundreds and thousands of binary attributes but a particular record would only have a few of them set to 1. Examples include retail transaction data sets and Web log data sets, where each row is a transaction or session and each column represents a product or Web page.

Owners may have a lot of questions about their data. For instance, it may be of interest to know how often the pages W_1 and W_2 but not W_3 were requested together. Such types of questions about the data can be formalized as Boolean, typically conjunctive, queries on arbitrary subsets of attributes. The problem then is to find the frequency of rows in the dataset that satisfy query Q, or, equivalently the probability of the query $P(Q)$ with respect to the empirical probability distribution defined by the data.

Any Boolean query can be answered using a single scan through the dataset. While this approach has linear complexity and works well for the small datasets, it becomes infeasible for real time queries on huge datasets which do not reside in main memory. We would thus like to have an approximate algorithm that would allow us to trade accuracy in the estimate of $P(Q)$ with the time taken to calculate it.

This paper studies different probabilistic models for approximating $P(Q)$. There exists a substantial body of work on this problem in the database literature, largely from a non-probabilistic viewpoint; in this paper we focus exclusively on probabilistic approaches. Our hypothesis is that the probabilistic modeling approach is a useful general framework for approximate query answering. A simple model-based approach is to calculate the marginal frequencies of all attributes and use an independence model for $P(Q)$ (often the method of choice in commercial relational database systems). The independence model is easy to learn, has low time and space requirements but as we shall see below is fairly inaccurate. Mannila et. al. [MPS99, MS] introduced the idea of using frequent itemsets and the maximum entropy approach (we use abbreviation *maxent* in what follows) for query approximation. The motivation comes from the fact that there exist many efficient data mining algorithms for extracting frequent itemsets from massive data sets; and maximum entropy can be used to combine these itemsets to form a coherent probabilistic model (details are discussed below).

In this paper we improve the standard iterative scaling algorithm for learning parameters of the maxent models by showing how one can apply bucket elimination and clique tree approaches to take advantage of structure in the models and in the queries. We show that depending on the number of itemsets used as an input to the maxent models their prediction accuracy averages within 0.1-1% of the true count.

We also investigate a number of approaches which fill in the gap between the computationally inexpensive (but not very accurate) independence model and the relatively expensive maxent solution. These methods include tree structured belief networks that can be learnt from the data [CL68] and an ad hoc method based on the pairwise marginals between the attributes.

We provide extensive experimental results on the performance of all the methods on the real data. We compare models in terms of the accuracy of the approximation, and the time and amount of information the model requires.

The rest of this paper is organized as follows. In Section 2 we introduce notation and

give the formal statement of the estimation problem. Section 3 gives precise definitions of the methods that we apply to the query selectivity estimation problem and the methodology we use to compare them. Section 4 presents the empirical results and in Section 5 we draw conclusions and present some extensions.

2 Statement of the Problem and Notation

Let $R = \{A_1, \ldots, A_k\}$ be a relation schema (table header) with k 0/1 valued attributes (variables) and r be a table of n rows over this schema. We assume that the number of attributes is substantially smaller than the number of records, and that the data is sparse, i.e. the average number of 1's per row is substantially smaller than the number of attributes. A row of the table r satisfies a conjunctive query Q iff the values of the corresponding attributes in the query and in the row are equal. We are interested in finding the number of rows in the table r satisfying a given query Q defined on a subset of its attributes. We can view this *query selectivity estimation* problem in a probabilistic light and pose the problem as estimating the true frequency of Q in the table r using an approximate (and presumably much smaller and more efficient) probability model P_M.

The time involved in using a probability model P_M is divided into the *offline cost* T_P, i.e., the time needed for building the model, and the *online cost* $t_p(Q)$ needed to give the approximate answer to query Q using the model P_M. We use S_P to denote the amount of space (memory) needed to store the model P_M.

For a given class of queries, let $\pi(Q)$ denote the probability of the query Q. We assume that this distribution is known, but in principle we could learn $\pi(Q)$ for a population or individuals. By $e_P(Q)$ we denote the absolute error in answering the query Q, i.e., the difference between the true count $C_t(Q)$ and the count estimated from the model P_M. We are interested in the expectation of the relative error with respect to the underlying query distribution $E_\pi[e_P(Q)/C_t(Q)]$. We use the empirical relative error defined as

$$\hat{E} = \frac{1}{N_{Q's}} \sum_{j=1}^{N_{Queries}} \frac{|e_P(Q_j)|}{C_t(Q_j)}, \tag{1}$$

where $N_{Q's}$ is the number of random query drawings from $\pi(Q)$ and $C_t(Q_j)$ is the true count of the query Q_j. Equation 1 is just an empirical estimate of the expectation of the relative error.

3 Models

3.1 Full data and Independence model

There are a wide variety of options in choosing the model P_M. One extreme is to store the entire dataset so that for each record we will only keep a list of columns that have 1's in them. We will have 100% accurate estimates, but for most of the real-life datasets this approach will incur inordinately large memory requirements, namely $S_P = O(c \sum_{i=1}^{n} N_{1's}(i))$, where c is the prespecified number of bits required to store a number to some fixed precision and $N_{1's}(i)$ is the number of positively initialized attributes in the record i.

The other extreme is also easy to describe—*the independence model.* Since the data is binary-valued, we only have to store one count per attribute. The probability of a

conjunctive query is approximated by the product of the probabilities of the single attribute-value combinations occurring in the query. Obviously, S_P is small in this method, namely $O(kc)$ bits. The preprocessing can be done by a single scan through the data, and the online cost consists of n_Q multiplications, where n_Q is the number of conjuncts in the query Q. However, as we shall see, the quality of the approximations produced by the independence method can be relatively poor.

3.2 Model Based on the Multivariate Tree Distribution

This model, first introduced by Chow and Liu [CL68], assumes that there are only pairwise dependencies between the variables and that the dependency graph on the attributes is a tree. To fit a distribution with a tree it is sufficient to know the pairwise marginals of all the variables. The algorithm consists of three steps, namely, computing the pairwise marginals of the attributes (complexity is $O(k^2 n)$), then the mutual information between the attributes (complexity is $O(k^2)$) and, finally, applying Kruskal's algorithm [CLR90] to compute the minimum spanning tree of the full graph whose nodes are the attributes and the weights on the edges are the mutual informations (complexity is $O(k^2 \log k)$). The dominating term in the overall offline time complexity will be $O(k^2 n)$ due to the computation of the marginals. The memory requirements for the algorithm is $O(k^2 c)$. Once the tree is learned, we can use the standard belief propagation algorithm to get the answer to a particular conjunctive query Q in time linear in n_Q.

3.3 Ad Hoc Model Based on the Pairwise Marginals

An ad hoc algorithm based on the pairwise marginals will compute P_M based on the chain rule and some specfied ordering of the attributes:

$$P_M(q_1, \ldots, q_{n_Q}) = P(q_1) \prod_{i=2}^{n_Q} P(q_i | q_{i-1}, \ldots, q_1) \tag{2}$$

and substitute the higher order conditionals with first order ones:

$$P_M(q_1, \ldots, q_{n_Q}) \approx P(q_1) \prod_{i=2}^{n_Q} P(q_i | q_{i-1}) \tag{3}$$

Since the ad hoc algorithm only involves computing the marginals, its memory requirements will be the same as for the tree algorithm while the offline time cost will be less.

3.4 Maximum Entropy Model

Let us first introduce a definition of an *itemset* that will be extensively used in the rest of the paper. An itemset associated with the binary table r with the schema R is by definition either a single positively initialized attribute or a conjunction of the mutually exclusive positively initialized attributes from R. We will call an itemset *T-frequent* if its count in the table r is at least T where T is some predefined non-negative threshold.

There exist efficient algorithms to compute all the itemsets from large binary tables (see, e.g., [MTV94, AS94]). In practice the running time of these algorithms is linear in both the size of the table and the number of frequent itemsets provided that the data is sparse. Thus, by computing itemsets we won't typically incur a high preprocessing cost.

The maximum entropy approach makes use of the T-frequent itemsets and the associated frequency counts treating them as constraints on the query distribution. Indeed, each pair of (a) an itemset and (b) its associated frequency count can be viewed as a value of the marginal distribution on the query variables when they all are positively initialized. Consider an arbitrary conjunctive query Q on variables $x_Q = \{q_1, \ldots, q_{n_Q}\}$. Forcing the estimate P_M to be consistent with the T-frequent itemsets for some $T > 0$ restricts P_M to a constrained set $\mathcal{P}$ of probability distributions within the general n_Q-dimensional simplex containing all possible distributions defined on n_Q variables.

Information about frequencies of the T-frequent itemsets for some $T > 0$ in general under-constrains the target distribution and we will need an additional criterion to pick a unique estimate $P_M(Q)$ from the set $\mathcal{P}$ of all plausible ones. The maximum entropy principle provides such a criterion. It essentially instructs one to select a distribution that is as uninformed as possible, i.e. makes the fewest possible commitments about anything the constraints do not specify. Given maximum entropy as a preference criterion, we face a constrained optimization problem of finding $P_M(x_Q) = \arg\max_{P \in \mathcal{P}} H(P)$, where $H(P)$ is the entropy of the distribution P . If the constraints are consistent (which is clearly the case with itemset-based constraints) one can show that the target distribution will exist, be unique [BPP96, Ros96, PPL97, MPS99] and can be found in an iterative fashion using an algorithm known as iterative scaling [DR72, CT84]. Note that we are estimating the whole distribution on the variables x_Q, not only the particular cell corresponding to a specific initialization of variables in the given query Q.

In order to get a probability estimate $P_M(x_Q)$ we will only retain itemsets whose variables are subsets of x_Q. Enforcing the j-th constraint c_j can be performed by just summing out from $P_M(x_Q)$ all the variables not participating in the j-th itemset (the variables in the constraint c_j should be kept fixed to 1), and requiring that the result of this sum equals the true count f_j of the j-th itemset in the table. Constraint c_j will thus look like:

$$\sum_{x_Q \in \{0,1\}^{n_Q}} P_M(x_Q) G(A_1^j = 1, ..., A_{n_j}^j = 1) = f_j \tag{4}$$

where $G(.)$ is the indicator function.

Whenever we say that initialized query variables x_Q satisfy a given constraint c_j, we shall mean that variables x_Q agree in their values with all the variables c_j. It can be shown (see, e.g., [Jel98]) that the maxent distribution will have a special product form

$$P_M(x_Q) = \mu_0 \prod_{j=1}^{N} \mu_j^{G(x_Q \ satisfies \ c_j)} \tag{5}$$

Once the constants μ_j are estimated, Equation 5 can be used to evaluate any query on the variables in x_Q, and Q in particular. The product form of the target distribution will contain exactly one factor corresponding to each of the constraints. Factor μ_0 is a normalization constant whose value is found from the condition

$$\sum_{x_Q \in \{0,1\}^{n_Q}} P_M(x_Q) = 1 \tag{6}$$

The general problem is thus reduced to the problem of finding a set of numbers μ_j from Equations 4 and 6. The iterative scaling algorithm is well known in the statistical literature as an iterative technique which converges to the maxent solution for problems of this general form (see, e.g., [Jel98]). A high-level outline of the most computationally efficient version of the algorithm [Jel98] is as follows.

1. Choose an initial approximation to $P_M(x_Q)$
2. While (Not all Constraints are Satisfied)
   For (j varying over all constraints)
   Update μ_0;
   Update μ_j;
   End;
   EndWhile;
3. Output the constants μ_j

The update rules for parameter μ_j^t corresponding to the constraint c_j at iteration t are:

$$\mu_0^{t+1} = \mu_0^t \frac{1-f_j}{1-S_j^t} \tag{7}$$

$$\mu_j^{t+1} = \mu_j^t \frac{f_j(1-S_j^t)}{S_j^t(1-f_j)}, \tag{8}$$

where S_j^t is defined as:

$$S_j^t = \sum_{x_Q \ satisfying \ c_j} {P_M}^t(x_Q) \tag{9}$$

Equation 9 essentially calculates the constraints as in Equation 4 but using the current estimate ${P_M}^t$ which in turn is defined by the current estimates of the μ_j^t's via Equation 5. Since the estimate ${P_M}^t$ may not necessarily meet the c_j-th constraint, equations 7 and 8 update the terms μ_0 and μ_j to enforce it. The algorithm proceeds in a round-robin fashion to the next constraints, thus at each iteration getting closer to satisfying *all* of them.

Convergence of the algorithm can be determined by various means. In the case of the query selectivity estimation problem, we are interested only in one cell of the distribution on the query variables corresponding to a particular query Q. We monitor this particular cell and terminate the algorithm when

$$|{P_M}^t(Q) - {P_M}^{t-1}(Q)| < \varepsilon {P_M}^{t-1}(Q). \tag{10}$$

ε is one of the free parameters of the algorithm, in the experiments we looked at $\varepsilon = 10^{-4}$. It usually takes 10-15 iterations for the algorithm to meet such convergence criterion.

Preprocessing for the maxent model consists of finding the itemsets for the entire dataset given a specified threshold T. Suppose that we have selected N itemsets on the sets of variables $I_j = \{A_1^j, ..., A_{n_j}^j\}$ and we know their frequencies f_j in the table r. Then the memory cost is $O(c(\sum_{k=1}^N n_k + N))$. The first term in this estimate corresponds to storing the attributes that are set to 1 in each of the itemsets, and the second term - to storing the counts of the itemsets in the table r.

The main computation in the iterative scaling algorithm falls on summing out the distribution ${P_M}^t(x_Q)$ according to Equation 9. The total number of summands in Equation 9 is $2^{n_Q - n_j}$. Each summand will have a product form and will contain at most N factors. Thus, the overall time complexity of performing summation in Equation 9 once for all the factors μ is $\sum_{j=1}^N \alpha_j 2^{\alpha_j}$, where $\alpha_j = n_Q - n_j$. The last estimate is obviously upper-bounded by $O(N n_Q 2^{n_Q})$. Note that this estimate is independent of the size of the original dataset. Although the exponential time complexity in the size of the query makes the method prohibitive for large query sizes, it is still feasible to use it for queries of length 8 or

so in practice. The iterative scaling algorithm in its formulation above has linear memory complexity in n_Q since the summation in Equation 9 can be performed using backtracking.

In the next subsection we discuss how one can speed-up the iterative scaling algorithm by trading memory for time based on the structure of the itemsets used to constrain the distribution.

3.5 Trading Memory for Time in Iterative Scaling

We propose several strategies for reducing the computational complexity of the iterative scaling algorithm.

3.5.1 Bucket Elimination

One possible approach to reducing the high computational complexity of iterative scaling is to apply a technique called *bucket elimination* [Dec99] which is essentially a smart way to do bookkeeping as one goes along the updates of factors in the representation of the maxent distribution. The idea here is to use the distributive law (see, e.g., [MA99]) in Equation 9.

Bucket elimination has both time and memory complexity exponential in the *induced width* of the constraint graph. The notion of induced width is closely related to the size of cliques of the graph and can be thought of being equal to the size of the largest clique in the constraint graph after it is triangulated. Note that any itemset on n_I variables will correspond to a clique of size n_I in the constraint graph.

Consider a simple example. We issue a query on six binary attributes $A_1, \ldots, A_6$ and there are frequent itemsets corresponding to each single attribute and the following frequent itemsets of size 2: $\{A_2 = 1, A_3 = 1\}$, $\{A_3 = 1, A_4 = 1\}$, $\{A_4 = 1, A_6 = 1\}$, $\{A_3 = 1, A_5 = 1\}$ and $\{A_5 = 1, A_6 = 1\}$. Constraint graph H in the figure 1 shows the interactions between the attributes in this example.

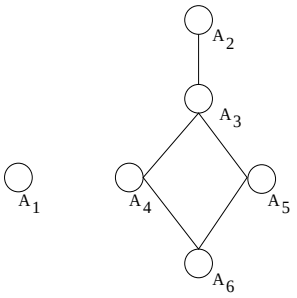

Figure 1: Constraint graph for an example problem

The maxent distribution according to Equation 5 will have the following form

$$P = \mu_0 \prod_{i=1}^{6} \mu_i^{G(A_i=1)} \prod_{i,j} \mu_{ij}^{G(A_i=A_j=1)} G(\exists edge(i,j)\ in\ H)$$

Suppose, that on the current iteration we are updating μ_{56} corresponding to the itemset $\{A_5 = 1, A_6 = 1\}$. According to our update rule in Equation 9, we need to fix attributes A_5 and A_6 to 1 and sum out the rest of the attributes in $P^t(A_1, \ldots, A_6)$. We will get

$$\sum_{A_1,\ldots,A_4,A_5=1,A_6=1} P(A_1, \ldots, A_6) = \mu_0\mu_5\mu_6\mu_{56}.$$

$$\cdot \sum_{A_1,\ldots,A_4,A_5=1,A_6=1} \prod_{i=1}^{4} \mu_i^{G(A_i=1)} \prod_{i,j:\ \exists edge(i,j)\ in\ H} \mu_{ij}^{G(A_i=A_j=1)}$$

It is easy to see that brute force summation over all the values of $A_1, \ldots, A_4$ will involve computing 16 terms, each having a product form. The bucket elimination algorithm will produce exactly the same result but will do it more efficiently—the number of terms to evaluate is reduced by a factor of 2 compared to the brute force method:

$$\sum_{A_1,\ldots,A_4} P(A_1,\ldots,A_4,A_5=1,A_6=1) = \mu_0\mu_5\mu_6\mu_{56}\cdot$$
$$\cdot \sum_{A_4} \mu_4^{G(A_4=1)} \mu_{46}^{G(A_4=1)} \sum_{A_3} \mu_3^{G(A_3=1)} \mu_{34}^{G(A_3=A_4=1)} \mu_{35}^{G(A_3=1)}$$
$$\cdot \sum_{A_2} \mu_2^{G(A_2=1)} \mu_{23}^{G(A_3=A_4=1)} \sum_{A_1} \mu_1^{G(A_2=1)}$$

The distributive law thus allows for a more time-efficient implementation of the iterative scaling procedure.

3.5.2 Clique Tree Ideas

Another way to speed-up the iterative scaling algorithm is based on the decomposability of the probability distribution with respect to the graph of the model. The detailed treatment of such ideas can be found in several recent papers [JP93, Mal92] and in the earlier book by Pearl [Pea88]. We provide a brief overview here.

We first create a *chordal* graph H' from H. To enforce chordality we use a graph triangulation algorithm [Pea88].

For the chordal graph, the joint probability distribution on the variables corresponding to its vertices can be decomposed into the product of the probability distributions on the maximal cliques of the graph divided over the product of the probability distributions on the clique intersections. The maximal cliques of the graph are placed into a *join* tree that shows how cliques are interacting with one another. The join or clique tree for our problem is given in the figure 2.

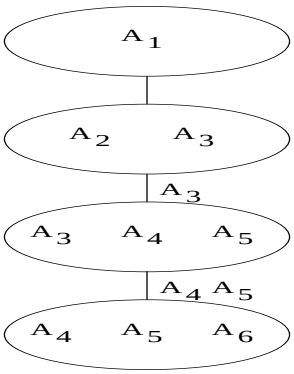

Figure 2: Clique tree corresponding to the problem in example. Cliques and their intersections are shown.

Thus, the original problem is decomposed into the smaller problems corresponding to the cliques of the triangulated graph H'. Each smaller problem can be solved using iterative

scaling, while distributions corresponding to the intersections can be found by summing out the corresponding distributions on the cliques. As we noted above, the time complexity of the iterative scaling algorithm grows exponentially with the size of the query. Thus, the algorithm that solves a number of smaller problems instead of solving a single big one may be considerably more efficient.

Finally, it is possible to combine the join-tree approach with the bucket elimination by first decomposing the original problem into smaller ones and then processing each of the cliques by iterative scaling, which in turn can use bucket elimination.

Table 1: Characteristics of the Data Sets. k is the number of attributes, n is the number of records, $N_{1's}$ is the number of 1's in the data, $E(N_{1's}) = N_{1's}/n$, $Std(N_{1's})$ is the standard deviation of the number of 1's in the record, $Max(N_{1's})$ is the maximum number of 1's in the record.

	k	n	$N_{1's}$	$E(N_{1's})$	$Std(N_{1's})$	$Max(N_{1's})$
MS Web Data Set	294	32711	98654	3	2.5	35
Retail	52	54887	224580	4.09	3.98	44

4 Empirical Results

4.1 Conjunctive Queries

We ran experiments on the two datasets: "The Microsoft Anonymous Web" dataset (publicly available at the UCI KDD archive) and a large proprietary dataset of consumer retail transactions. Both datasets contained binary transaction data. Before learning the models we analyzed the structure of the data and the itemsets that can be derived from it. Parameters of the datasets are given in the table 1.The retail dataset appears to be much more dense than the Microsoft Web Data. For more dense data sets, the larger itemsets will be more frequent, and the resulting constraint graph will also be more dense.

We empirically evaluated the following models for conjunctive query selectivity estimation: (1) the independence model, (2) the ad hoc model based on the pairwise marginals, (3) the Chow Liu tree model, and (4) the maxent model that used either of brute force, bucket elimination and clique tree methods. All experiments were performed on the Pentium III, 450 MHz machine with 128 Mb of memory. We generated 500 random queries for query sizes of 4, 6, and 8, and evaluated different models with respect to the average memory, online time and error per Equation 1.

To select a query we first fixed the number of its variables $n_Q = 4$, 6 or 8. Then we picked n_Q attributes according to the probability of the attribute taking a value of "1" and generated a value for each selected attribute according to its univariate probability distribution. Note, that negative values for the attributes are more likely in the sparse data than positive ones. Thus, because of the query generation algorithm, generated queries typically had at most one positively initialized attribute.

The plots in Figure 3 show the dependence of the average relative error on the memory requirements for the model (or the model complexity) for the Microsoft Web data. The independence model that uses the least memory is the most inaccurate one. The ad hoc and the Chow-Liu tree models use the same amount of memory, however the tree model is more accurate as expected. Note that all the maxent models, i.e. brute force, bucket

Table 2: Comparison of the Models on Arbitrary Boolean and Purely Conjunctive Queries.

	Conjunctive				*Arbitrary*			
n_Q	t_P	C_t	e_P		t_P	C_t	e_P	
			Indep.	Maxent			Indep.	Maxent
4	0.052	14000	0.163	0.0021	0.059	25000	0.0066	$8.2 \cdot 10^{-5}$
6	0.248	9200	0.304	0.0067	0.258	28000	0.0135	$2.8 \cdot 10^{-4}$
8	2.036	6700	0.342	0.0112	2.525	29000	0.0319	$6 \cdot 10^{-3}$

elimination or clique tree, will have the same average error for a fixed query length since they essentially estimate the same product form of the distribution.

That is why on this figure we only report results for a single "maximum entropy model" (since all 4 produce the same estimates, but using different computational methods). Circles that show results for the maxent model correspond to various values of the threshold T that was used to define itemsets. We varied T as $15, 30, 50, 60, 100$ and 200. The higher the value of T, the less information is supplied to the model and the less accurate the results are. Thus, the leftmost circle corresponds to $T = 200$ and the rightmost to $T = 15$.

The maxent model outperforms in terms of accuracy both the tree and the ad hoc models even when the amount of memory for the maxent model is less than for the rivals. This assertion holds true for all the query sizes. Another important observation is that for all the models the error increases with increasing query size.

We also measured the average online time taken by various models to generate the estimated query count. Figure 4 illustrates how the error depends on the online time for all the models and query sizes $4, 6$, and 8. Among the various models the independence model is the fastest but the least accurate of all. The ad hoc model is slightly slower and a bit more accurate.

The Chow-Liu model fills in the large gap between the cluster of the independence and the ad hoc models and the cluster of the maxent models. On all three plots the diamond corresponding to the Chow-Liu model has nearly the same x-coordinate, showing a very slow increase in processing time as query size grows. The quality of the tree model is not as good as for the maxent models. Recall that the amount of information supplied to the Chow-Liu model is comparable to, and for some of the threshold values is even greater than, that for the maxent models.

The brute force, clique tree and bucket elimination maxent models have smaller errors than the other models but it takes them longer to produce the estimates. The error for all three types of maxent models is the same, and so is the y-coordinate for all points corresponding to the same threshold T; the only difference will be in the online time. The brute force version is the fastest on the queries of size 4: this is not surprising, as both the clique tree and the bucket elimination methods have an overhead that dominates for the queries of small sizes. The clique tree method becomes the fastest for the query sizes 6 and 8.

As the threshold T decreases (the corresponding points on the plots go from left to right, top to bottom) and the number of itemsets (or memory size) increases, we see that the difference in the online time between the maxent algorithms that employ the graph structure and the brute force maxent decreases. We attribute this to the fact that when the number of itemsets increases, so does the average density of the constraint graph and the average induced width.

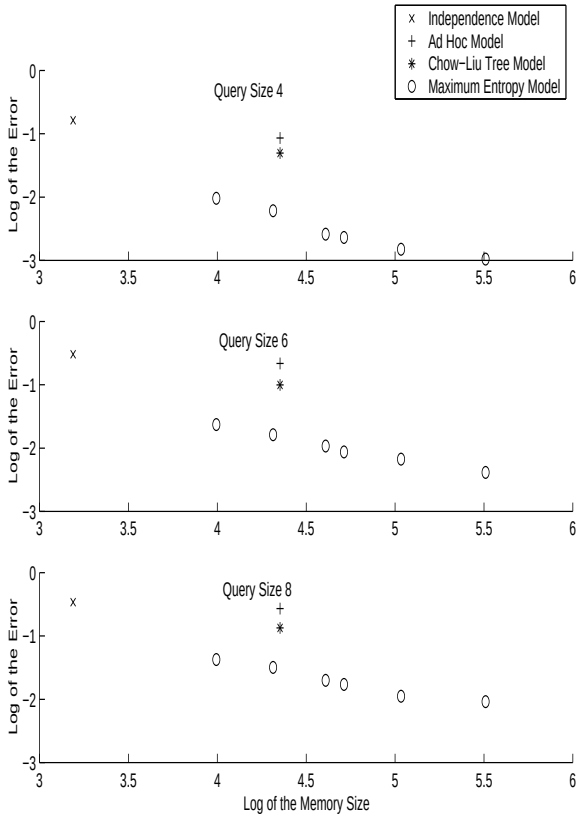

Figure 3: Average relative error on the 500 random queries as a function of model complexity for the Microsoft Web data.

The performance of various models relative to one another on the *retail* data was qualititatively the same as for the Web data with the maxent model being the most accurate but most computationally expensive. However, due to the fact that the retail data is much more dense, the memory requirements and the online running times are generally much higher than for the Web data.

4.2 Arbitrary Boolean Queries

It is straightforward to generalize the maxent approach to handle arbitrary Boolean queries. For a given arbitrary (not necessarily conjunctive) query we first estimate the maxent distribution on the query variables, then transform the query to a disjunctive normal form and evaluate the distribution on the disjuncts. This approach will be worst case exponential in the query size but so is iterative scaling.

We have run experiments on arbitrary Boolean queries that we generated according to the algorithm described above for conjunctive queries. The only difference is that the connective between two attributes was selected as either a disjunction or a conjunction by flipping a fair coin. Table 2 compares results on arbitrary and purely conjunctive queries (n_Q is the query length, t_P, C_t and e_P are the average online time, query count and error across 200 runs of the algorithms). The maxent models again enjoy a distinct advantage in accuracy over the independence models.

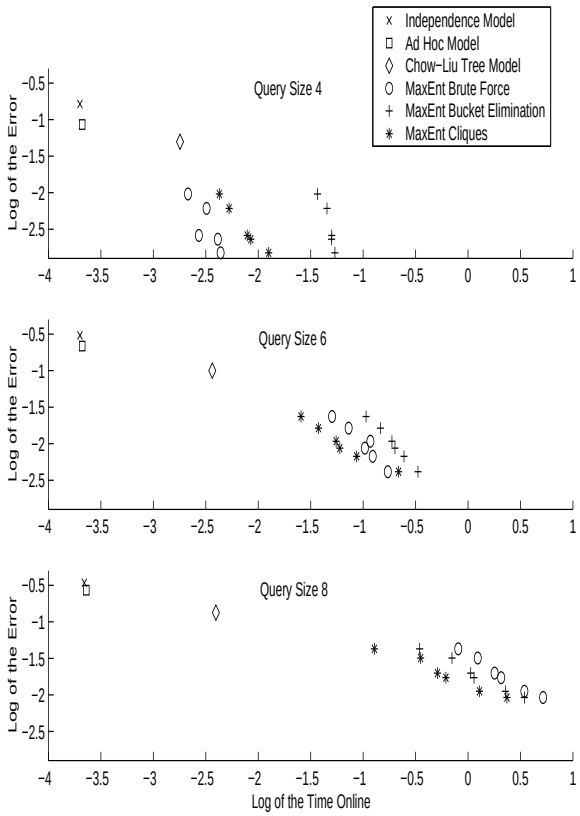

Figure 4: Average relative error on the 500 random queries as a function of the online time for the Microsoft Web data.

5 Conclusions and Extensions

We have shown that (a) probabilistic models in general, and (b) the maximum entropy approach in particular, provide a useful general framework for approximate query answering on large sparse binary datasets. We have analyzed the relative performance of various probabilistic models for this problem and showed that given sufficient information about the data in the form of itemsets maxent approach will be the most accurate of all the models. We also showed how bucket elimination and clique tree ideas can be employed for speeding up the learning of the maxent models.

The work described in this paper allows for several possible extensions. For arbitrary Boolean queries one can in principle incorporate query structure directly into the iterative scaling algorithm or into bucket elimination (see [DS] for an example of such an approach).

In addition, there are several important open questions involving modeling of the query distribution: how should the query model be chosen? can it be learned from online user data? if it is known a priori, can it be profitably used in generating the approximate probability model, e.g., can one spend more resources on modeling parts of the data which have high probability of being queried?

Acknowledgements

The research described in this paper was supported in part by NSF CAREER award IRI-9703120.

References

[AS94] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules in large databases. In *Proceedings of the Twentieth International Conference on Very Large Data Bases (VLDB'94)*, pages 487 – 499, 1994.

[BPP96] A.L. Berger, S.A. Della Pietra, and V.J. Della Pietra. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–72, March 1996.

[CL68] C. K. Chow and C. N. Liu. Approximating discrete probability distributions with dependence trees. *IEEE transactions on Information Theory*, IT-14(3):462–467, 1968.

[CLR90] T.H. Cormen, C.E. Leiserson, and R.R. Rivest. *Introduction to Algorithms*. MIT Press, 1990.

[CT84] I. Csiszár and G. Tusnády. Information geometry and alternating minimization procedures. *Statistics & Decisions, Supplement Issue*, (1):205–237, 1984.

[Dec99] R. Dechter. Bucket elimination: A unifying framework for structure-driven inference. *AI*, 1999.

[DR72] J. N. Darroch and D. Ratcliff. Generalized iterative scaling for log-linear models. *Annals of Mathematical Statistics*, 43:1470–1480, 1972.

[DS] R. Dechter and P. Smyth. Processing boolean queries over belief networks. *Submitted to UAI-2000.*

[Jel98] F. Jelinek. *Statistical Methods for Speech Recognition*. MIT Press, 1998.

[JP93] R. Jirousek and S. Preucil. On the effective implementation of the ipf procedure. *Computational statistics and data analysis*, 1993.

[MA99] Robert J. McEliece and S. M. Aji. The generalized distributive law. *IEEE Trans. Inform. Theory*, 1999.

[Mal92] F.M. Malvestuto. A unique formal system for binary decompositions of database relations, probability distributions and graphs. *Information Sciences*, 59:21–52, 1992.

[MPS99] H. Mannila, D. Pavlov, and P. Smyth. Predictions with local patterns using cross-entropy. *KDD 1999*, 1999.

[MS] H. Mannila and P. Smyth. Approximate query answering with frequent sets and maximum entropy. *To appear, ICDE 1999.*

[MTV94] H. Mannila, H. Toivonen, and A. I. Verkamo. Efficient algorithms for discovering association rules. In *Knowledge Discovery in Databases, Papers from the 1994 AAAI Workshop (KDD'94)*, pages 181 – 192. AAAI Press, 1994.

[Pea88] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers Inc., 1988.

[PPL97] S. Della Pietra, V. Della Pietra, and J. Lafferty. Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(4):380–393, April 1997.

[Ros96] R. Rosenfeld. A maximum entropy approach to adaptive statistical language modelling. *Computer Speech and Language*, 10(3):187–228, July 1996.