Chapter 7

Web-Based Knowledge Acquisition to Solve Inverse Problems Arising in Health-Care Management

V. Rao Vemuri and Na Tang

Automated web-based knowledge acquisition can play a useful role in developing systematic methods for solving inverse problems arising in the context of healthcare management. As inverse problems are ill-posed, they are normally solved by using some regularization procedure - a mathematical strategy that seeks to supply the "missing data." We seek to fill the missing data by an automated knowledge discovery process via mining the WWW. This novel procedure is applied by first restoring missing information via web mining and next learning the structure and parameters of the unknown system from the restored data. We learn the Bayesian network structure by looking at various possible interconnection topologies. The parameters, i.e. the probabilities associated with the causal relationships in the network, are deduced using the knowledge mined from the WWW in conjunction with the data available on hand. Using heart disease data sets from the UC Irvine Machine Learning Repository, this procedure is tested and some preliminary results are presented.

Key words: Inverse problems, Structure learning, Web search, Bayesian nets, Heart disease data

1 Introduction: What are Bayesian Networks?

Bayesian networks (also called belief networks) play an important role in many areas of research. A Bayesian network is represented as a *directed acyclic graph*. Each node represents a variable and each arc specifies the probabilistic cause-effect relation that exists between nodes. This information is often managed via a table, called a probability table. Each node can represent a simple binary true/false or Boolean variable, an array of discrete variables (e.g. cold, warm, hot) or even continuous data (e.g. temperature). For most systems these values are typically discrete.

There is a whole family of Bayesian networks; the more complex the interdependence among the entities representing the nodes, the more complex the network. A simple (or N*aive*) Bayes model assumes (rightly or wrongly) that two or more findings are independent with no statistical bearing on each other. Figure 1 shows a Naive Bayes model of the interdependence between heart disease and a number of factors. A more realistic – and somewhat more complex – topology, shown in Figure 2, is called the Tree Augmented Naive (TAN) Bayes model. Many more topological configurations are possible, depending on the nature of cause-effect relationships in a given problem.

If the Naive Bayes assumption is true, and if all the probabilities are known, then estimating a posterior probability (say, the probability of a heart attack given the prior probabilities of precursor conditions) is straightforward. Clinically, however, to solve most diagnostic tasks, one cannot assume each piece of evidence is unrelated as the Naive Bayes model does. In medicine, as in other domains, this conditionally independent state is usually an exception. But, to represent all the conditional probabilities (so that each possible combination of clinical findings is accurately represented) we would require enormous tables containing each possible conditional probability. The task then becomes much more difficult, if not impossible. For example, if *n* represents each finding or clinical symptom, then we would need a conditional probability table with

2^n entries. If we were to represent heart disease with *13* clinical symptoms, then we would need $2^{13} = 8,192$ entries.

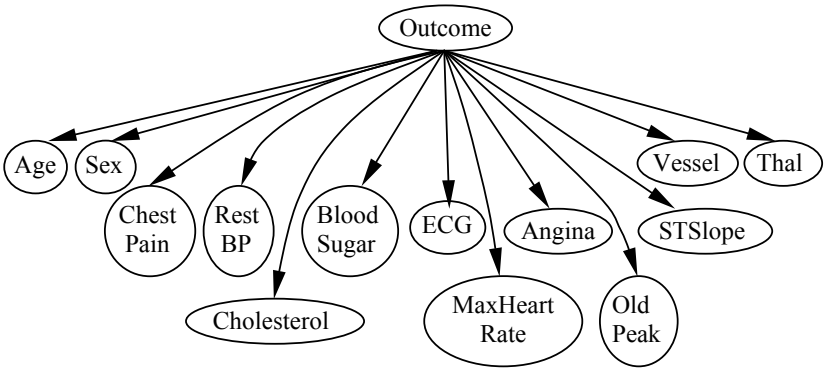

Figure 1. Naive Bayes Model.

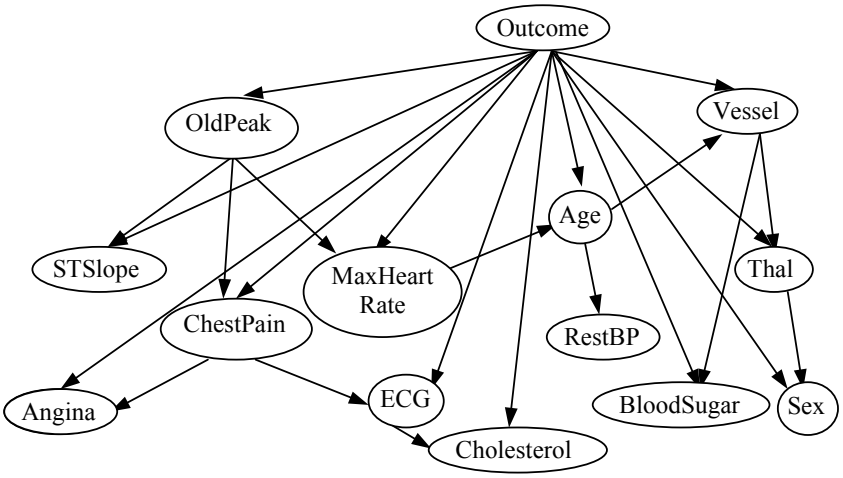

Figure 2. A Tree-augmented Naive (TAN) Bayes Model.

In many medical applications, however, the findings (i. e. clinical evidence) are often dependent upon each other and the independence assumption of Naive Bayes will tend to over-estimate the likelihood of a given hypothesis. Since each finding is treated independently, the presence or absence of a finding (and its inherent

weight or probability) will have no bearing on the other. Although simple Bayes often performs well in the right environment, the development of other *Bayesian networks* help to resolve this interdependency of relationships between findings as well as make management of larger collections of evidence easier and more intuitive. Bayes theorem is a robust and extensible method for reasoning under uncertainty. In medical applications Bayes methods can be implemented, depending on the relative complexity of the problem, from a simple probability approach to more complex Bayesian probabilistic frames each representing a specific disease or diagnostic decision. Further, a disease or process may be represented within a Bayesian network containing many nodes representing a wide variety of causally dependent (or independent) disease conditions. Simple Bayesian probability helps us to understand our belief in a hypothesis given some prior - or *a priori* - evidence that is either explicitly known or estimated to determine the posterior probability given a new finding.

2 Inverse Problems, Bayesian Nets and Knowledge Discovery

The focus of this chapter is on solving *inverse* problems associated with Bayesian inference.

What are inverse problems and where do they arise? Inverse problems arise anywhere data is collected which is related to the unknown quantities by a mathematical model. The *direct* problem consists of predicting the output, given the model and the input. This problem, in general, does not pose any great difficulties. Other problems, in general, are more difficult:

Identification: Given the input and the output, estimate the parameters of the model

Inversion: Given the model and the output, estimate the input

Blind inversion: Given the output, estimate both the input and the parameters of the model

Design: Given a specification, design a system in such a way as to be able to solve optimally the aforementioned problems.

Examples of inverse problems can be found in many walks of life. Diagnosis of a disease from symptoms is a familiar example from the medical field. Here, the system can be visualized as a directed acyclic graph whose "top" layer of nodes can represent diseases and "bottom" layer can represent observed symptoms. The goal is to infer the posterior probability of each disease given the symptoms (which can be present, absent or unknown). In a densely connected graph exact inference is impossible and various approximation methods can be used.

Inverse problems are mathematically *ill-posed* and therefore are hard to solve. An inverse problem does not have a unique solution; indeed many different hypotheses (or models) may result from the same set of observations. This non-uniqueness is often resolved by using *auxiliary* information. In the modeling of dynamical systems (such as those described by differential equations), this *auxiliary* information will be in the form of initial, boundary or interface conditions. In probabilistic formulations (such as those described by Bayesian networks) this information will be in the form of prior and conditional probabilities. A very generic mathematical structure for the incorporation of auxiliary information is the well-known regularization procedure [16].

Depending on the amount of insight one has about the graph, its topology, the various cause-effect pathways within the graph, the parameters associated with the nodes and edges, and so on, several flavors - from purely black box (nothing is known about the system) to a completely white box (everything is known about the system) - can be recognized.

In the *white box*, or analysis, problem one assumes that the network structure is specified in advance and the variables characterizing the structure and the cause-effect relations among the nodes are given. That is, there is sufficient “knowledge” (in the form of labeled training data) from which the structure and relations can be derived (learned). In this case, this problem can be treated as a direct problem and the learning task is to estimate the conditional probabilities at the various nodes of the network. Then standard Bayesian methods can be used to draw necessary inferences in a probabilistic manner. The task here is not much more difficult than

the calculations associated with the well-known Naive Bayes classifier.

If one assumes that the network structure is given (that is, the number of nodes and how they are connected to each other), but *not all* of the variables are observable, then the learning problem is a bit more difficult. Indeed, it is possible to use a method similar to the Backpropagation method of neural nets (i. e., a gradient ascent procedure) and learn the entries of the conditional probability tables (CPT's). The objective function that is maximized during this gradient ascent is $P(D \mid h)$, the probability of observing the data D given the hypothesis h. This, indeed, corresponds to a search for the maximum likelihood hypothesis.

In *parameter estimation* problems complications can arise if direct access to the data necessary to estimate the unknown parameters becomes difficult or impossible. Perhaps some of the data items are missing. The missing data issue typically arises when an outcome is the result of the accumulation of simpler outcomes, or when outcomes are clumped together in a 'binning' or histogram operation. This problem can be further complicated if the number of underlying data points is unknown.

The next level of difficulty arises if the network structure is *not* known in advance, although data is fully observable. There has been some effort in learning network structure from data [5] [7], although this problem is known to be NP-hard [7]. The learning procedure usually searches a space of possible networks to find a best structure fitting the data sets. The strategy is to consider a family of possible network structures and choose one among them. One way to tackle this problem is to use a scoring metric for choosing one of the many alternative networks (or hypotheses, in machine learning parlance). Two cases can be considered here:

Case 1: The unknown network structure is inferred by search. Using heuristics, it is possible to search the space of possible networks. For example, the greedy strategy trades off network complexity for accuracy over the training data. Alternatively, a genetic algorithm can probabilistically search the space of solutions.

Case 2: The unknown network structure is inferred by exploiting the dependence/ independence relations in the observed data. This, in turn, will help in building a Bayesian network.

The most challenging of inverse problems occurs when only a subset of the relevant features is observable, that is, the problem of learning in the presence of unobserved variables. One possible way to address this issue is to use the observed data to infer information about unobserved variables. There is some hope in tackling this problem if one assumes some knowledge about the probability distributions to which the unobserved variables belong. A well-understood method in this special case is the Expectation Maximization (EM) algorithm. Briefly, the EM algorithm searches for the maximum likelihood hypothesis *h'* by seeking that *h'* that maximizes E [$ln\ P(Y \mid h')$] where Y stands for the entire data set, namely $X \cup Z$, in which X is the observed data and Z is the unobserved data.

3 Architecture of a Stochastic Engine

Stochastic Engine is the name given to the framework under which we propose to solve this family of inverse problems. The approach we wish to take is akin to the generate-and-test method of classical AI. To implement this, a generating procedure is required that systematically generates hypotheses and the testing step verifies whether or not a hypothesis generated satisfies certain criteria. The generating mechanism is data-driven in the sense that any generated hypothesis should be consistent with the training data. Portions of these training data sets themselves are created via a *knowledge discovery* process.

Figure 3 shows the architecture of an interactive stochastic engine that facilitates the generate-and-test cycle. The Front-end, dubbed the Graphical Bayesian Network Constructor (GBNC), assumes that all necessary data is already available in a relational format and the Back-end, dubbed the Web Mining Engine (WME) combs the WWW and supplies (missing) data to these tables.

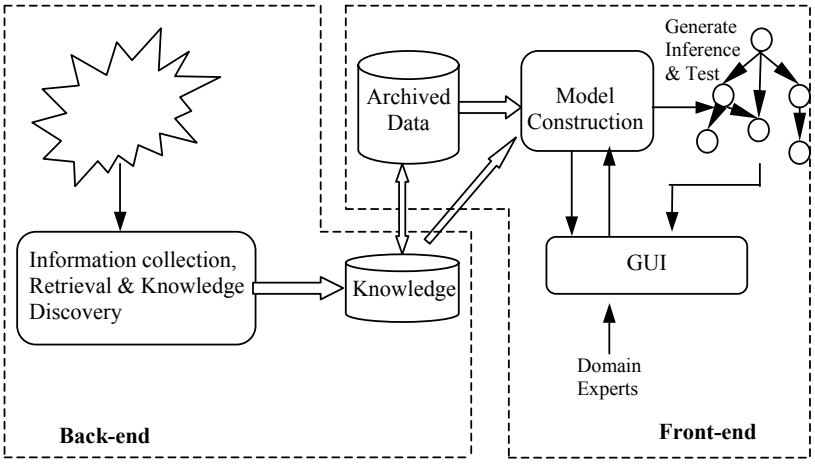

Figure 3. Architecture of the Stochastic Engine. Knowledge acquisition is done in the back-end and the hypotheses space is explored in the front-end.

4 Exploring Hypotheses Space via Families of Bayesian Networks (Front-end)

The "front-end" of the stochastic engine facilitates an exploration of the hypotheses space. In the "automatic mode" this exploration can be conducted using a genetic algorithm that works with populations of Bayesian networks. In the "interactive mode," the front-end allows for the construction of a variety of Bayesian networks. The interactive mode allows either the use of pre-programmed topologies or user-defined, interactively generated, topologies. At this time, two pre-programmed topologies are available in our model library: Naive (or simple) Bayes and Tree Augmented Naive Bayes (TAN). A GUI to accomplish these functions is shown in Figure 4.

Any Bayesian network can be visualized as consisting of two parts: (a) a qualitative part and (b) a quantitative part. The qualitative part captures the network structure in the form of a directed acyclic graph containing nodes and edges. In such a network the nodes

stand for domain variables and the edges stand for the (probabilistic) dependences over these variables. The quantitative part defines the probabilities required, namely, the prior probabilities and the conditional probability tables, if sufficient data is available to perform these calculations. When the required data is not available, sufficient or reliable, then one option is to actively search for the information on the WWW. This knowledge acquisition step is performed by the back-end.

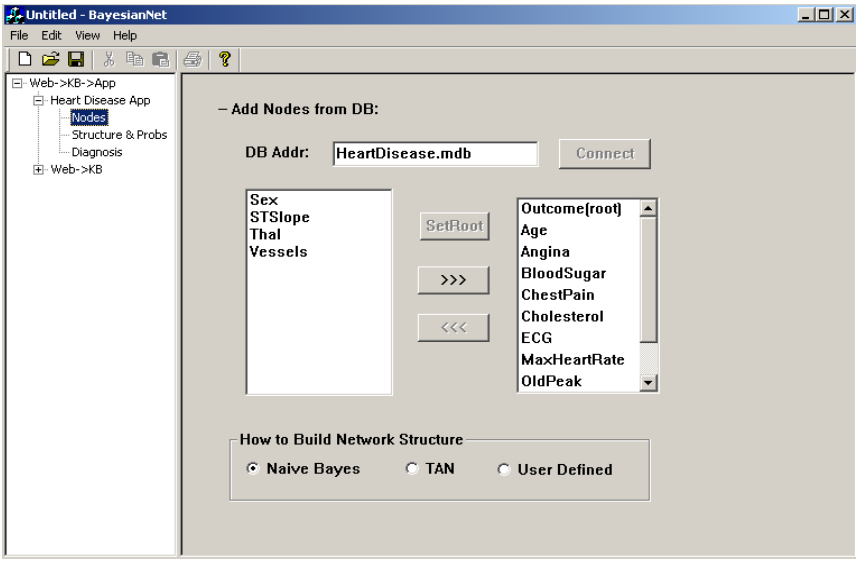

Figure 4. Screenshot of the GUI to help build a family of Bayesian networks.

4.1 The Qualitative Part

At this time the GUI shown in Figure 4 permits the construction of virtually any Bayesian network topologies.

4.1.1 Naive Bayes Model

The Naive Bayes network is the simplest Bayesian network given all the attributes A_i and the class variable C. It assumes C is the root and the parent of each A_i, and there are no dependence between any two attributes A_i and A_j $(i \neq j)$. Although this strong assumption of

independence does not hold in the real world (e.g. the value of the attribute "Age" apparently affects the value of the attribute "MaxHeartRate"), the experimental results still show its good accuracy and it proves to be very useful in many applications. Figure 1 shows the Naive Bayes network for heart disease data sets.

4.1.2 Tree Augmented Naive (TAN) Bayes Model

The TAN, on the other hand, is defined as the set of all possible trees in which the class variable C is the parent of any attribute A_i and each A_i has at most one more parent A_j. This is more general than Naive Bayes, yet operating in a restricted search space [5]. Figure 2 shows a TAN Bayes model for the same heart disease data set. TAN provides a conditional log-likelihood scoring function to evaluate each candidate network in a restricted search space. The restricted search space is a set of all possible trees in which the class variable C is the parent of any attribute A_i and each A_i has at most one more parent A_j. The conditional log-likelihood (CLL) scoring function in TAN is given as follows:

$$CLL(B|D) = \sum_{i=1}^{N} \log P_B(C^i \mid A_1^i, \ldots, A_n^i) \quad \textbf{(1)}$$

Here B is a candidate Bayesian network, D is the dataset, C is the class variable, $A_1 \ldots A_n$ are the attributes, N is the number of classes we need to predict and C^i stands for the ith value of the class variable.

This *CLL* scoring function is a sub item in the minimum description length (MDL) scoring function. TAN uses this function because experiments show that a non-specialized scoring function such as *Bayesian scoring* and MDL may get poor results when the number of attributes becomes large. Thus TAN chose a specialized part (CLL) from MDL scoring function. In addition, TAN searches a restricted space of Bayesian networks, which leads to a much smaller complexity compared to unrestricted learning.

The implementation of TAN adopts a greedy strategy for searching:

1. Compute the *conditional mutual information* between each pair of attributes:

$$I(A_i; A_j \mid C) = \sum_{a_i, a_j, c} P_D(a_i, a_j, c) \log \frac{P_D(a_i, a_j \mid c)}{P_D(a_i \mid c) P_D(a_j \mid c)} \quad (2)$$

2. Build the undirected complete graph by connecting each pair (A_i, A_j); then build a *Maximum Spanning Tree* by using the $I(A_i; A_j \mid C)$ as the weight for the edge (A_i, A_j).

3. Transform this undirected tree into a directed one by choosing an attribute as the root.

4. Add additional edges from the class variable to each attribute of the tree generated from this procedure to get TAN model.

The time complexity for TAN is $O(n^2N)$, where n is the number of nodes and N is the number of classes (i.e. the number of class variables).

4.1.3 User-defined Model

The User-defined option allows for the construction of networks with arbitrary topology, based on user's inputs and preferences. Users are responsible to specify nodes and edges and their relationships. This model is usually used by domain experts who have sufficient domain knowledge for all attributes of the problem. Domain knowledge can be used as a guide for Bayesian network construction (front-end) as well as a source to serve as the knowledge base in the back-end. In this paper, we only focus on discovering knowledge from the web.

4.2 The Quantitative Part

The probabilities associated with the nodes of the network are of two types: (a) the prior probability for the class variable, which is the root, and (b) the conditional probabilities for the other nodes.

The prior probabilities of the class variables are calculated using

$$P(C = c_i) = \frac{N(C = c_i)}{\sum_{c_i} N(C = c_i)} \quad (3)$$

Here the notation *N(Pr)* stands for the number of records where the predicate *Pr* holds.

The conditional probabilities are calculated using

$$P(A_i = a_{ij} \mid Parents(A_i)) = P(A_i = a_{ij} \mid P_{A_i}^1 = p_{k_1}^1, \ldots, P_{A_i}^n = p_{k_n}^n) = \frac{N_a}{N_p} \quad (4)$$

Here N_a is $N(A_i = a_{ij}, P_1 = p_{k_1}^1, \ldots, P_n = p_{k_n}^n)$ and N_p is $N(P_1 = p_{k_1}^1, \ldots, P_n = p_{k_n}^n)$. Here $P_{A_i}^j$ stands for the *j*th parent node of A_i. Standard textbooks should be consulted for details of these methods as well as the smoothing techniques [9]. Once all the above probabilities are calculated, the specification of the Bayesian net is complete.

Smoothing Methods to Avoid Zero Probabilities: It is possible that the conditional probability associated with the attribute value $a_{j,k}$ and some class c_i is 0. Then the value of the $P(C = c_i \mid A_1, \ldots A_j = a_{j,k}, \ldots A_n)$ would be 0 no matter what values of the other attributes are. Thus the log score for c_i would be less than any other class values. Therefore, even if the values of the other attributes strongly indicate the class value c_i, the prediction result would not be c_i because of this bias. Several smoothing methods are available to modify the probability calculation that avoids 0/1 probabilities.

Two smoothing methods are considered: Laplace estimate and *m*-estimate. The Laplace estimate simply modifies every probability in the same direction:

$$P(A_i = a_{ij} \mid Parents(A_i)) = \frac{N_a + 1}{N_p + n} \quad (5)$$

Here n is the number of the attributes. The m-estimate modifies the probability in the direction of the prior probability of the attribute A_i:

$$P(A_i = a_{ij} \mid Parents(A_i)) = \frac{N_a + mP(A_i)}{N_p + m} \quad (6)$$

Here m is the equivalent sample size. Reference [5] uses a similar method for the m-estimate:

$$P(A_i = a_{ij} \mid Parents(A_i)) = \alpha \cdot \frac{N_a}{N_p} + (1-\alpha) \cdot P(A_i), \text{ where } \alpha = \frac{N \cdot P(A_i)}{N \cdot P(A_i) + 5} \quad (7)$$

This smoothing operation also modifies the probability in the direction of prior probability of the attribute A_i. This operation is termed*, prior smooth* in this chapter.

In the experiments conducted and reported here, the smoothing operation seems to improve TAN but not Naive Bayes. A possible explanation is that the probabilities in Naive Bayes network are well balanced and not likely to be 0/1 while the probabilities in TAN have higher chances to be 0/1, but this explanation needs further investigation.

4.3 Experimental Results

The methods NB, TAN, along with some variations in smoothing strategies, were tried on the heart disease data sets and some preliminary results are shown in Table 1. The metrics used, namely, *Accuracy* and *True Negative,* are defined as follows:

$$Accuracy = \frac{\text{Number of correct predctions}}{\text{Total Number of predictions}}$$

$$True\ Negative = \frac{\text{Number of incorrect predctions that predicted no heart disease}}{\text{Total number of predictions}}$$

Table 1. Experimental results for different methods on heart disease data sets

Experiments	NB	Laplace Smoothed NB	Prior Smoothed NB	TAN	Laplace Smoothed TAN	Prior Smoothed TAN
Accuracy	78.2%	79.8%	77.4%	79.0%	79.7%	81.8%
True Neg.	16.2%	15.4%	18.2%	15.9%	16.1%	14.7%

These preliminary results indicate that although the Prior Smoothed TAN performed slightly better than the others, further study is necessary to draw any meaningful conclusions.

5 Restoring Missing Data by Mining the WWW (Back-end)

When data is incomplete, the scoring function for evaluating network structures is not in a closed form, which makes the Bayesian learning very difficult. Several statistical methods are available to deal with missing data for Bayesian network construction:
(1) Filling in missing data values using available data as a guide [14].
(2) Expectation-maximization (EM) algorithm [2] [4]: The EM algorithm estimates the parameters by iteratively finding the expectation of the parameter and then finding the maximum likelihood estimate (MLE) using the parameter from the expectation step. Reference [4] starts an initial structure and passes the structure to the EM algorithm. The MLE returned by EM algorithm is considered as the score for the structure. A new structure is generated by adding, deleting or reversing an edge in the previous structure and the new structure is passed to the EM algorithm again and the score for the new structure is returned and compared to the previous score. The process is repeated until there is no improvement for the score. One of the problems with the EM method is that the deterministic search tends to find the local optima. Multiple restarts are suggested to avoid this problem [8].

(3) Evolution Algorithm (EA) [10]: The EA uses a genetic algorithm to evolve both network structures and missing values in order to find an optimal Bayesian network. It can also avoid local maximization because of the stochastic search. Another probability is to use simulated annealing.

Most of these methods do not work well when a large percentage of data is missing and they do not work well with non-random missing data (e.g. a whole column is missing). Alternatively, we propose a novel approach to deduce the missing information via a knowledge discovery process and content mining of the WWW. Reference [1] shows a methodology of extracting useful information to fill a knowledge base. This idea is modified and explored further in our approach to facilitate Bayesian learning.

There are some other statistical methods - the so-called imputation techniques (mean substitution, regression imputation, hot-deck imputation etc. [12]) - to fill in missing data for classification tasks. These methods can also be used when some attributes are totally missing. While we strive to fill in missing values using the existing data as well as the new knowledge obtained from the web, these imputation techniques accomplish the task by relying on statistical methods applied only on the existing data, while our approach mines the web for new data. The imputation approach, however, is simple - even simpler than the EM algorithm. On the debit side, the imputation methods cannot reach high accuracy. For example, when applied to a specific example, the imputation method achieved an 80% accuracy with a complete dataset, and 74% accuracy when fill-in values. Work reported in this chapter did better than this.

The starting point for this phase of the work is the "heart disease data" available at the UCI Machine Learning Repository [15]. Four distinct data sets on heart disease are available at this location. Each of these data sets has fourteen attributes. Many values of the attribute “Cholesterol” are missing in the "Va data set" (see Table 2). We call the attribute containing missing values *incomplete attribute*.

Table 2. “Va” Data Set (one of the four heart disease data sets) from UCI repository. All “-1”s stand for missing values.

Age	Sex	Chest Pain	Rest BP	Choles-terol	Blood Sugar	ECG	···	Outcome
60	1	3	180	-1	0	1	…	0
60	1	3	120	-1	-1	0	…	1
60	1	2	160	267	1	1	…	1
56	1	2	126	166	0	1	…	0
59	1	4	140	-1	0	1	…	1
62	1	4	110	-1	0	0	…	1
63	1	3	-1	-1	0	2	…	1
63	0	2	-1	-1	0	0	…	0
62	1	4	152	153	0	1	…	1
56	1	2	124	224	1	0		0
…	…	…	…	…	…	…	…	…

5.1 System Framework

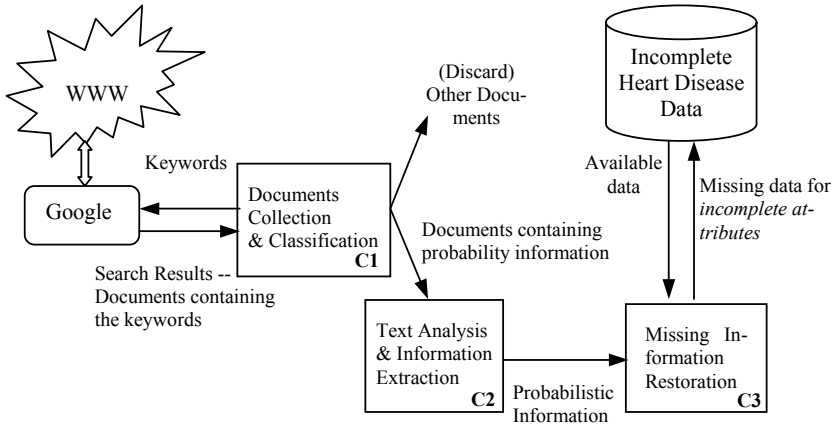

Figure 5. Implementation Architecture for Generating Missing Information

The crux of the idea for retrieving the missing information from the WWW is to look for patterns of *relationship* between the incomplete attribute and some or all of the other attributes of the problem. It is assumed that the sought after *relationship* information appears in a web document either as a natural language sentence or as an item in a table. In either case the relationship indicates how the other attributes influence the *incomplete attribute* or how the *incomplete attribute* influences the other attributes. The implemen-

tation architecture is shown in Figure 5, (which constitutes the back-end of Figure 3).
The three components C1, C2 and C3 are explained in the following subsections. In this content it is assumed that "Cholesterol" is an *incomplete attribute* in the heart disease data set.

5.2 Box C1: Documents Collection and Classification

Document Collection Phase: We collected *N* (= 300 or more, typically) documents from Google by searching on keywords about the *incomplete attribute* and other attributes. In our case, since "Cholesterol" values are missing, we want to get the probability information between "Cholesterol" and other attributes. To extract documents about "Cholesterol" and "Outcome", for example, we used the keyword set: <cholesterol "heart disease">. (Similarly, the keyword set <cholesterol age> was used to get documents containing relationship information between "Cholesterol" and "Age" and so on.) We chose Google because it is a very general, and widely available, search engine; it can deal with all sorts of topics making it a convenient framework applicable to many fields. A specialized search engine and a specialized database, such as Medline, may work better for the healthcare field.
Document Classification Phase: A trained Naive Bayes Classifier (Rainbow [13]) was used to divide the resulting documents into two classes: the *positive class* containing information on heart disease causes *and* probability data and the *negative class* comprising of all other documents. As documents in the *positive class* (positive documents) contain the probability information we want, they are retained for further processing; the *negative class* documents are discarded. Figure 6 shows a segment of a positive document in which the sought after information is shown highlighted with an underline.

PROPERTIES OF CHOLESTEROL

Cholesterol is a fatty, waxy substance made by your liver and found in every living cell in your body. It is necessary for certain essential functions, such as the production of hormones. Many people who eat a diet rich in fatty foods have high cholesterol. Elevated cholesterol levels in the blood are caused by many factors, including being overweight, certain inherited tendencies, smoking and lack of exercise.

Too much cholesterol in your blood can create deposits on the inside of your arteries. Over time, these build-ups may clog the arteries and restrict blood flow - forcing your heart to work harder to keep the blood moving. And if the blood can't bring enough oxygen to the heart muscle, chest pain - and even heart attacks - can result. Experts believe that most people should have a total cholesterol level under 200 mg to reduce these risks absolutely, but a level of 200-240 may not increase risk significantly. In people older than 70, there is no current data that proves that lowering cholesterol helps increase life span or decrease illness or death, but many doctors feel that it is wise to control cholesterol in older people just as in younger people.

By testing your blood, your doctor can measure several substances relating to cholesterol: total cholesterol, triglycerides (another type of blood fat), as well as LDL and HDL (low- and high-

Figure 6. A Segment of a *Positive* Document. The underlined parts express the relation between “Cholesterol” and “Outcome”

Training data: The Naive Bayes Text Classifier needs to be trained before the preceding step can be implemented. This training is performed manually as follows. Using a number of professional web sites that specialize in heart disease [6] 292 documents were collected, manually inspected and hand-labeled as positive and negative (78 for the *positive class* and 214 for the *negative class*). During this stage we were looking for documents that contained relevant probability information. The Naive Bayes Text Classifier was trained using this as a training set.

The parameter setting for the classifier is as follows: 1) *Token option*: The classifier skips HTML tags, i.e. it does not treat HTML tags as words in documents. It also uses stop list, i.e. it skip a list of heavily-used words which do not indicate the content of the documents. These words include "a", "an", "the", "because" etc. It does not apply stemming operation, which is a method to convert words into its root. For example, "continue", "continuous" and "continuation" has the same root "continu". 2) *Event model*: Different event models represent different ways to represent the probabilities. There are two commonly used event models: *word-based* (multi-

nomial) and *document-based* (multivariate, Bernoulli). The former considers the word frequency for a word in a document as its real occurrences in that document, while in the latter model, if a word appears in a document, the word frequency is set to unity no matter how many times this word appears in that document. It is shown that the document-based model works well with small-sized vocabulary while the word-based model works well with large-sized vocabulary [11]. We use word-based event model because the documents we obtain from the web involve a large size of vocabulary. Also, the word-based model gave us the higher classification accuracy. 3) *Smoothing method*: Smoothing methods are used to avoid 0/1 probabilities, i.e. the bias of a probability that the training data induces. We use no smoothing methods in our classification.

5.3 Box C2: Text Analysis and Information Extraction

Because the documents are HTML files coming from the web, the embedded knowledge may reside in free text as well as semi-structured text (e.g. tables). For the latter, some HTML tags are very useful for our extraction. The probability information we want to extract, i.e. the relations between the incomplete attribute and other attributes, is divided into two categories: *Point probabilities* and *Qualitative influences*. Other forms of probability information such as comparison and qualitative synergies would also be useful. Formal definitions of all these items are given in [3] but we only focus on the above two categories. Rules are used to extract these two types of probability information.

(1) Point probabilities: These are probabilities expressed in the mathematical form $P(a_i|a_j) = c$, where c stands for some constant. Consider the relation between "Cholesterol" and "Outcome" as an example. The degree of risk for heart disease for different levels of cholesterol is usually explicitly described in the relevant documents. For example, the probability information indicated by the text in Figure 6 is that *P(Outcome = 1 | Cholesterol < 200 mg/dl)* is low, *P(Outcome = 1 | 200 mg/dl < Cholesterol < 239 mg/dl)* is borderline high and *P(Outcome = 1 | Cholesterol > 240 mg/dl)* is very high.

In the documents we examined, we found that the point probabilities are typically expressed in two ways: *(a)* by the use of tables (semi-structure text, see Example 1) and *(b)* by the use of regular sentences (free text, see Example 2).

Example 1: Figure 7 (a) and (b) illustrate how the relation between "Cholesterol" and "Outcome" could appear in a table. The rule to extract useful information from that table is shown in Figure 7 (c). If the table fits the regular expression in Figure 7 (c), we extract the cholesterol levels and the degree of heart disease risk.

Total Cholesterol Levels	
< 160 mg/dL	optimal for people with a history of hea
< 200 mg/dL	desirable for the general population
200 mg/dL to 239 mg/dL	borderline high blood cholesterol
240 mg/dL or greater	high blood cholesterol

HTML source →

(a)

```
<table border="1" cellpadding="3" cellspacing="0">
    <tr><td colspan="2" bgcolor="#FFFFFF"><p
align="center"><strong>Total Cholesterol Levels</strong></p></td></tr>
    <tr><td>&lt; 160 mg/dL</td>
        <td>optimal for people with a history of heart disease</td></tr>
     <tr><td>&lt;   200 mg/dL</td>
        <td>desirable for the general population</td> </tr>
     <tr><td>200 mg/dL to 239 mg/dL</td>
        <td>borderline high blood cholesterol</td></tr>
     <tr><td>240  mg/dL or greater</td>
        <td>high  blood cholesterol</td>
     </tr>
</table>
```

(b)

```
Table =  TableStartTag * TableEntryTag * level * degree * TableEndTag
          | TableStartTag * TableEntryTag * degree * level * TableEndTag,
Length(table) < 1500 ,
Contains("total cholesterol", Table) = True,

   TableStartTag = "<table*>", TableEndTag = "</table>",
   TableEntryTag = "<tr*>",
   level   = LeftOp Number ("mg/dl" | "")
           | Number ("mg/dl" | "") MidOp Number ("mg/dl" | "")
           | Number ("mg/dl" | "")  RightOp,
   degree= degree1 | degree2 | degree3 | degree4
   degree1 = "optimal" | "ideal" | "ideally" …
   degree2 = "desirable" | "ok" | "healthy"…
   degree3 = "borderline" …
   degree4 =  "high" | "serious" …
   LeftOp  = "less than" | "lower than" | "below" | "under" | "&lt;"
           | "greater than" | "over" | "&gt;"
   MidOp  = "to" | "-"
```

(c)

Figure 7. *Point Probabilities* Expressed in Tables and Extraction Rule. (a) Text that appears in the web browser; (b) Text that read by computers; (c) Extraction Rule. Here "*" stands for any character and "|" for "or".

The output resulting from the processing shown in Figure 7 is ("< 160 mg/dl", "optimal"[degree1]), ("< 200 mg/dl", "desirable"[degree2]), ("200 mg/dL to 239 mg/dl", "borderline high"[degree3]) and ("240 mg/dl or greater", "high"[degree4]). The output (*level*, *degree*) can be interpreted in probability format as $P(Outcome = 1 | Cholesterol \in level) = degree$.

Example 2: An example of probabilities expressed by regular sentences (free text) is given below:

"In general, total cholesterol is considered high when 240 or more, borderline when 200-239, and desirable when 200 or less."

The above sentence is from the document in Figure 6. Figure 8 shows the rules to extract the required information.

If the sentence matches the expression in Figure 8, we extract cholesterol levels and the degree of heart disease risk. The output for the sentence above is ("200 or less", "desirable"[degree2]), ("200-239", "borderline"[degree3]) and ("240 or more",

"high"[degree4]). The output (*level, degree*) can be converted to probability information in the same way with the table extraction.

```
Sentence = * level * degree * | * degree * level *;
Contains("cholesterol", Sentence) = True;

    level   = LeftOp Number ("mg/dl" | "")
              | Number ("mg/dl" | "") MidOp Number ("mg/dl" | "")
              | Number ("mg/dl" | "")  RightOp;
    degree = degree1 | degree2 | degree3 | degree4;
    degree1  = "optimal" | "ideal" | "ideally" …
    degree2  = "desirable" | "ok" | "healthy"…
    degree3  = "borderline" …
    degree4  =  "high" | "serious" …
    LeftOp   = "less than" | "lower than" | "greater than" | "&lt;" | "&gt;";
    MidOp   = "to" | "-";
    RightOp = ("and" | "or") ("lower" | "less" | "below" | "more" | "greater"
                             | "higher"  | "above")

    Output: (level, degree)
```

Figure 8. Extraction Rule for *Point Probabilities* in Regular Sentences. "<" is "<" and ">" is ">" in HTML documents.

(2) Qualitative Influences: This kind of relation describes how one attribute influences another in a qualitative way. A positive qualitative influence from attribute A_i to A_j means choosing a higher value for A_i makes the higher value for A_j more likely. A sentence example to describe the relation between "Age" and "Cholesterol" is "As people get older, their cholesterol levels rise". This describes a positive qualitative influence from "Age" to "Cholesterol".

Example 3: An example of extracting information about qualitative influences is given below.

"As people get older, their cholesterol levels rise."
"Cholesterol levels naturally rise as men and women age."
"Old people have higher cholesterol levels than the youth."
"Women have lower total cholesterol levels than men of the same age."

The above four sentences all match the rule in Figure 9. The output of the first three confirms a positive influence from "Age" to "Cholesterol" and the output of the fourth confirms a positive influence from "Sex" to "Cholesterol" (here, we assume "women" < "men" as the order of the attribute "Sex").

Sentence = * description(A_i) * description(cholesterol) *
| * description(cholesterol) * description(A_i) *
| * description(A_i)$_1$ * description(cholesterol) * "than"
* description(A_i)$_2$,
where IsSameLevel(description(A_i)$_1$, description(A_i)$_2$) = False
Length(sentence) < 150;
Contains("cholesterol", Sentence) = True;

description(cholesterol) = subject(cholesterol) * level(cholesterol)
| level(cholesterol) * subject(cholesterol);
description(age) = subject(age)* level(age);
distance(subject, level) < 20 ;
description(gender) = level(gender);
subject(age) = "people" | "human" | "woman" | "women"
| "man" | "men";
subject(cholesterol) = "cholesterol";
level(A_i) = higher(A_i) | lower(A_i) ;
higher(cholesterol) = "higher" | "high" | "rise" | "increase";
lower(cholesterol) = "low" | "low" | "decrease";
higher(age) = "older"| "old";
lower(age) = "younger"| "young";
higher(gender) = "female"| "woman" | "women";
lower(gender) = "male"| "man"| "men";
…
Output: **If** (IsSameLevel(description(A_i), description(cholesterol)) ||
IsSameLevel(description(A_i)$_1$, description(cholesterol)))
Then Positive influence from A_i to "cholesterol"
Else Negative influence from A_i to "cholesterol"

Figure 9. Extraction Rule for *Qualitative Influence*

Each sentence and each table from the documents retrieved in C1 (i. e. the positive documents) are analyzed, examined to find if they matched any of our rules and collected the outputs of the probability information for the next stage, namely C3.

We assumed that the probability information published in most websites is the most reliable information. For example, if most websites show that it is optimal to have cholesterol less than 200mg/dl while a few websites regard 160mg/dl as the separation line, then we choose to believe the former. In addition, the information we obtained came from the top of the list returned by

mation we obtained came from the top of the list returned by the search engine. We are implicitly making the assumption that the higher-ranked search results are more reliable.

5.4 Box C3: Missing Information Restoration

There are two types of output from Box C2: point probabilities and qualitative influences. The point probabilities for the relation between "Outcome" and "Cholesterol" can be represented by:

$$P(Outcome = 1 \mid Cholesterol = v_1) = v_2 \quad (8)$$

Here v_1 stands for a range of cholesterol levels and v_2 stands for some constant value. The second type of output from Box C2 includes a positive influence from "Age" to "Cholesterol" and a positive influence from "Sex" to "Cholesterol". A positive influence from "Age" to "Cholesterol" means that with lager value of "Age" the risk of getting higher values of "Cholesterol" is greater. This fact can be represented by:

$$P(Cholesterol > v \mid Age = a_1) > P(Cholesterol > v \mid Age = a_2) \text{ given } a_1 > a_2 \quad (9)$$

A positive influence from "Sex" to "Cholesterol" can be interpreted in a similar way. Given all these probability outputs from Box C2 and given the probability constrains such as *P(Age, Sex, Outcome) = v* from the available data, we can elicit the probabilities *P(Cholesterol| Age, Sex, Outcome)* based on the approach described in [3]. This method allows us to convert all the probability information into a linear system of equalities and inequalities, from which bounds on the probabilities of interest are calculated. From these bounds, it is possible to elicit the required probabilities, namely, *P(Cholesterol| Age, Sex, Outcome)*. Now we can fill the missing values in the data set based on these probabilities. For example, we get:

$$\begin{aligned} &P(Cholesterol < 200 \mid Age < 50, Sex = female, Outcome = 1) = v_1, \\ &P(200 < Cholesterol < 240 \mid Age < 50, Sex = female, Outcome = 1) = v_2, \end{aligned} \quad (10)$$

P(Cholesterol > 240|Age < 50, Sex = female, Outcome = 1) = v_3,

If a patient's age is less than 50, is female, and has heart disease, then we set her missing cholesterol number to one of the values from the set {< 200, 200 - 240, > 240} respectively with probabilities $\frac{v_1}{v}$, $\frac{v_1}{v}$ and $\frac{v_1}{v}$, where $v = v_1+v_2+v_3$.

5.5 Experiments

We did our experiments on the Cleveland data set, which is a table with no missing data. We chose 2/3 of the data for training and the remaining 1/3 for testing. Using the complete data set (i.e. no missing values) we obtained 80.2% accuracy with Naive Bayes method and 81.1% with the TAN method. Then we assumed all the cholesterol values in the training data as missing and applied our method to fill in these values. Using the two sets of training data with the filled-in values, we trained our front-end respectively. We then tested the trained network with the testing data set and were able to obtain 79.2% accuracy with Naive Bayes and 83.2% with TAN (see Table 3). It is clear that the filled-in data set performed almost as good with Naive Bayes and even better with TAN. If we randomly fill in the missing values in the training data instead of using our mining approach, we can only obtain 78.2% accuracy with Naive Bayes and 77.8% with TAN, which proves that the filled-in data via our approach outperforms randomly filled-in data.

Table 3. A Comparison of prediction accuracies with complete and filled-in data

Prediction Accuracy	Naive Bayes	TAN
Complete Data	80.2%	81.1%
Incomplete Data with filled-in values via our approach	79.2%	83.2%
Incomplete Data with random filled-in values	77.2%	76.8%

6 Summary

This chapter is only a proof of concept for a new method of solving ill-posed inverse problems. The WWW was used as a source for gathering missing information in relational tables. The filled-in tables are, in turn, used to simulate two Bayesian network models of the unknown system under study. Several interesting questions need to be addressed before this method can be effectively used to solve realistic problems. The most obvious questions pertain to the reliability, scalability and performance of the method. Reliability depends on the confidence one can place on the filled-in numbers generated by this method. Scalability refers to the performance of the method as function of the percentage of missing values in the data set. Performance refers to the computational complexity of the method. Furthermore, as the filled-in values are only estimates, for the purpose of tracking changes, there has to be some cognizance as to which of the data items have been filled in. This will facilitate sensitivity analysis.

References:

1. Craven, M., DiPasquo, D., Freitag, D. McCallum, A. Mitchell, T. Nigam, K. and Slattery, S.: Learning to Construct Knowledge Bases from the World Wide Web. *Artificial Intelligence*, (118):69-113, 2000.

2. Dempster, A., Laird, N., and Rubin, D.: Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society*, B39:1-38, 1997

3. Druzdzel, Marek J., van der Gaag, Linda C.: Elicitation of Probabilities for Belief Networks: Combining Qualitative and Quantitative Information. *Proceedings of the Eleventh Annual Conference on Uncertainty in Artificial Intelligence (UAI-95)*, pp. 141-148, Morgan Kaufmann Publishers, Inc., San Francisco, CA, 1995.

4. Friedman, N.: The Bayesian Structural EM Algorithm. *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, pp. 647- 654, 1998.

5. Friedman, N., Goldzmidt M.: Bayesian Network Classifiers. *Machine Learning*, 29:131–163, 1997.

6. http://www.heartinfo.org/, http://www.americanheart.org, http://www.heartcenteronline.com, http://www.heartsavers.org, http://www.healthandage.com, http://www.nhlbi.nih.gov, http://heartdisease.about.com, http://www.medem.com/medlb/medlib_entry.cfm, http://www.med.umich.edu, http://www.google.com

7. Heckerman, D., Geiger D. and Chickering D. M.: Learning Bayesian Networks: The Combination of Knowledge and Statistical Data. *Machine Learning* 20(3):197-243, 1995.

8. Lauritzen, S. L.: The EM Algorithm for Graphical Association Models with Missing Data. *Computational Statistics & Data Analysis*, pp. 191-201, 1995.

9. Mitchell, Tom M.: *Machine Learning*. McGraw-Hill, 1997.

10. Myers, James W., Laskey, Kathryn B. and DeJong, Kenneth A.: Learning Bayesian Networks from Incomplete Data Using Evolutionary Algorithms. In Wolfgang Banzhaf, Jason Daida, Agoston E. Eiben, Max H. Garzon, Vasant Honavar, Mark Jakiela, and Robert E. Smith, editors, *Proceedings of the Genetic and Evolutionary Computation Conference*, 1:458-465, Orlando, Florida, USA, Morgan Kaufmann, 1999.

11. McCallum, A. and Nigam, K.: A Comparison of Event Models for Naive Bayes Text Classification. *In AAAI-98 Workshop on Learning for Text Categorization*, 1998.

12. Mundfrom, D. J. and Whitcomb, A. J. (1998). Imputing missing values: The effect on the accuracy of classification. *Multiple Linear Regression Viewpoints*, 25(1):13-19

13. Rainbow Library (in C language) http://www-2.cs.cmu.edu/~mccallum/bow/rainbow/

14. Spiegelhalter, D., Lauritzen, S.: Sequential Updating of Conditional Probabilities on Directed Graphical Structures. *Networks*, 20:579-605, 1990.

15. UCI Machine Learning Repository. http://www.ics.uci.edu/~mlearn/MLRepository.html

16. Tenorio, L.: Statistical Regularization of Inverse Problems, *SIAM Review*, 43(2):347-366, 2001.