

7-2007

Discovering and Exploiting Causal Dependencies for Robust Mobile Context-Aware Recommenders

Ghim-Eng YAP

Nanyang Technological University

Ah-Hwee TAN


Nanyang Technological University

Hwee Hwa PANG

Singapore Management University, hhpang@smu.edu.sg

DOI: <https://doi.org/10.1109/TKDE.2007.1065>

Follow this and additional works at: https://ink.library.smu.edu.sg/sis_research

 Part of the [Databases and Information Systems Commons](#), and the [Numerical Analysis and Scientific Computing Commons](#)

Citation

YAP, Ghim-Eng; TAN, Ah-Hwee; and PANG, Hwee Hwa. Discovering and Exploiting Causal Dependencies for Robust Mobile Context-Aware Recommenders. (2007). *IEEE Transactions on Knowledge and Data Engineering*. 19, (7), 977-992. Research Collection School Of Information Systems.

Available at: https://ink.library.smu.edu.sg/sis_research/1210

This Journal Article is brought to you for free and open access by the School of Information Systems at Institutional Knowledge at Singapore Management University. It has been accepted for inclusion in Research Collection School Of Information Systems by an authorized administrator of Institutional Knowledge at Singapore Management University. For more information, please email libIR@smu.edu.sg.

Discovering and Exploiting Causal Dependencies for Robust Mobile Context-Aware Recommenders

Ghim-Eng Yap, Ah-Hwee Tan, *Senior Member, IEEE*, and Hwee-Hwa Pang

Abstract—Acquisition of context poses unique challenges to mobile context-aware recommender systems. The limited resources in these systems make minimizing their context acquisition a practical need, and the uncertainty in the mobile environment makes missing and erroneous context inputs a major concern. In this paper, we propose an approach based on Bayesian networks (BNs) for building recommender systems that minimize context acquisition. Our learning approach iteratively trims the BN-based context model until it contains only the minimal set of context parameters that are important to a user. In addition, we show that a two-tiered context model can effectively capture the causal dependencies among context parameters, enabling a recommender system to compensate for missing and erroneous context inputs. We have validated our proposed techniques on a restaurant recommendation data set and a Web page recommendation data set. In both benchmark problems, the minimal sets of context can be reliably discovered for the specific users. Furthermore, the learned Bayesian network consistently outperforms the J4.8 decision tree in overcoming both missing and erroneous context inputs to generate significantly more accurate predictions.

Index Terms—Recommender systems, context-awareness, Bayesian networks.



1 INTRODUCTION

A recommender [1] is an application that ranks a set of available choices with respect to certain criteria. For instance, in information retrieval, search criteria are submitted as queries and the most relevant documents are recommended. With the proliferation of e-services, recommender systems are actively researched due to their obvious commercial values (for example, [2], [3]). In practice, recommender systems have contributed greatly to e-commerce successes like Amazon.com.

The two most popular recommendation techniques today are the *collaborative filtering* [4] and the *content-based* approach [5], although a number of other techniques exist [6]. A collaborative filtering recommender compares its users for similarity in rating profiles and returns items that similar users have rated highly. On the other hand, a content-based recommender learns from the properties of the items and returns the items that are similar to those that the user has rated highly in the past. A shortcoming of these techniques is that they do not take into account situational information, which seriously reduces their effectiveness. For instance, suppose that a user requests to “recommend me a restaurant with Japanese food.” A restaurant recommender selects a place with Japanese food but does not realize that the user would have preferred a nearer place as it is raining. Clearly, the

recommender is disadvantaged because it is unaware of its *context* of use, that is, the relevant situational information that characterizes the user-application interaction [7].

Recent years see a growing interest in *context-aware recommender systems* (for example, [8], [9], [10]). In particular, context-aware recommender systems promise far and viable business applications for mobile e-commerce, or *m-commerce* (for example, [2], [10]). However, current mobile context-aware recommenders face a number of unique challenges that have to be resolved, particularly, with respect to the acquisition of context in the volatile and highly constrained mobile environment.

1.1 Current Challenges and Our Proposed Solution

The first challenge is to identify the *minimal* set of important context parameters for a user. Mobile context-aware recommenders face serious resource limitations, so they need to acquire just the important context in order to conserve resources. A common solution to this problem is *feature selection* (for example, [9]), where statistical tests are used to retain the *individually significant* parameters. Other works like that by van Setten et al. [10] suggest allowing the users to explicate their own preferences via rules like “recommend me a café based on its location, but not on prices,” and Yap et al. [11] automates the personalization by using *Support Vector Machines* to identify the important context parameters for a user. All these prior approaches recognize and exploit the fact that, to individual users, some context parameters may be more important than others. However, they assume that the context parameters affect users’ decisions *independently* and fail to model their causal dependencies.

The second challenge is the presence of missing and erroneous, or *noisy*, context inputs. Within the volatile

• G.-E. Yap and A.-H. Tan are with the School of Computer Engineering, Nanyang Technological University, Nanyang Avenue, Singapore 639798. E-mail: {yapg0001, asahtan}@ntu.edu.sg.

• H.-H. Pang is with the School of Information Systems, Singapore Management University, 80 Stamford Rd., Singapore, 178902. E-mail: hhpang@smu.edu.sg.

Manuscript received 6 Feb. 2006; revised 5 Oct. 2006; accepted 30 Jan. 2007; published online 6 Feb. 2007.

For information on obtaining reprints of this article, please send e-mail to tkde@computer.org, and reference IEEECS Log Number TKDE-0057-0206.

Digital Object Identifier no. 10.1109/TKDE.2007.1065.

mobile environment, lapses in context acquisition frequently occur due to reasons such as failure to negotiate access rights to protected information, faulty sensors, and also intermittently broken or unstable communication links. Similarly, sources of context (for example, hardware and software sensors) are prone to errors. Mobile context-aware recommenders have to be equipped to maintain accurate predictions even under these imperfect conditions. To the best of our knowledge, no prior work has considered exploiting the causal dependencies among context parameters so as to address the problem of missing and erroneous context inputs. In particular, learning methods like *decision tree* (DT) and *support vector machine* do not encode causal dependencies and, thus, cannot compensate readily for missing and erroneous context inputs.

1.2 Our Contributions and Paper Organization

This paper has two primary research contributions. First, we propose a Bayesian network (BN)-based recommender that learns the *minimal* set of important parameters for a user to minimize context acquisition. Our learning procedure iteratively discards parameters that are not connected to the user rating variable in the learned BN. The remaining parameters constitute the minimal context for that user. In applications that reason with defined markers, that is, higher level concepts derived from the primitive context parameters, we apply the procedure on the markers. Those context parameters feeding into the minimal marker set would then constitute the minimal context. Since there are typically far fewer markers than context parameters, learning on markers allows the corresponding minimal context for a user to be identified from fewer examples.

Our second contribution is that we harness the causal dependencies among context parameters to make our recommender resilient against missing and erroneous context inputs. We show how causal dependencies can be captured effectively using a two-tiered context model. The BN learned on this context model is able to compensate for those context inputs that are left unspecified during the prediction. We also introduce a procedure for overcoming erroneous context inputs. The procedure uses the learned BN to estimate the error rates of individual context parameters from the training set. It then discards those inconsistent or outlier examples and learns a new network from the cleansed data. During the prediction, the erroneous context inputs are entered as likelihood estimates in order to account for their uncertainty.

We have evaluated the proposed techniques on a restaurant recommendation problem involving defined markers and strong causal dependencies among context parameters. We have also experimented with a Web page recommendation problem where there are only weak dependencies between word attributes. The results for both problems show that our iterative learning procedure reliably identifies the minimal set of important context parameters for the users. In addition, the learned BN consistently outperforms the J4.8 DT in overcoming missing and erroneous context inputs to produce consistently more accurate predictions on user ratings.

The rest of this paper is organized as follows: In Section 2, we describe the BN, the Causal discovery via Minimum Message Length (*CaMML*) program [12] for learning BN, and the prior works. We define *context-aware content-based*

recommender systems in Section 3, and we present our two-tiered context model in Section 4. In Section 5, we introduce our proposed techniques to overcome missing and erroneous context inputs. Section 6 presents our restaurant recommender application, and we evaluate our techniques on this application in Section 7. Section 8 presents a further validation on a benchmark Web page recommendation problem. Section 9 concludes this paper.

2 BACKGROUND

2.1 The Bayesian Network

Causal dependencies among the variables in a domain can be effectively modeled within the directed acyclic graphical model of a BN. These causal dependencies are captured qualitatively by the network structure, that is, the arcs linking the network variables (*nodes*), and quantitatively by the table of conditional probabilities that is associated with each node.

Heckerman [13] states that the local distribution functions in a BN are actually classification models. Past works [14], [15] have argued that, given complete data for prediction, neither DT nor BN would significantly outperform each other. However, unlike the DT, BN encodes the causal dependencies among parameters. The results we present in this paper clearly demonstrate that this makes the BN more suitable than the J4.8 DT for compensating missing and erroneous context inputs.

Most prior works rely on BNs that are either manually crafted or translated from existing ontologies [16]. However, because both the causal dependencies and the relative importance of the context parameters are user specific, we need techniques to automatically learn the network structure and to parameterize the learned network based on the data. In this work, we adopt the *CaMML* program [12] for automatic BN learning from the data.

2.2 Learning BN from the Data with CaMML

CaMML [12] stochastically searches over the entire space of causal models to find the best model h that maximizes a Minimum Message Length (MML) posterior metric [17]. The MML structural posterior metric that is used in *CaMML* is defined as $P_{MML}(h) = e^{-I_{MML}(h)}$, for $I_{MML}(h) = \log N! - \sum_i \log p_i - \sum_j \log(1 - p_j)$, where N represents the number of nodes in model h , p_i reflects the prior probability for a directed arc i present in h , and p_j reflects the prior probability for a possible arc j absent from h .

CaMML performs a *Metropolis search* in the space of all possible models. For each real model it visits, it computes a representative model and counts only on these representatives to overlook the trivial variations. This grouping of real models is necessary in practice because the space of real models is exponential in the number of nodes, and each model would otherwise get only a very small number of visits even for the sizable data sets [17]. The MML posterior of each representative is computed as the total MML posterior of its members. This total posterior approximates the probability that the true model lies within the MML equivalence class of the representative. The best model is hence the representative model with the highest MML posterior. For more details, please refer to [12] and [17]. It

suffices for us to note that, since the number of possible models is factorial in the number of nodes N , the theoretical worst-case complexity of the *CaMML* learning process is $O(N!)$, although, in practice, the average complexity of the stochastic Metropolis sampling should at most be exponential in the number of nodes.

2.3 Related Works

Context-awareness enhances both collaborative filtering and content-based recommendation. Collaborative filtering now considers also the *context* in which the user ratings are given. Recent works including [9] and [18] have recorded the context of each rating, such that if user A rates a restaurant in good weather, this rating would not be considered when it is raining. Effectively, this matches the users based on context in addition to their rating profiles. In [9], a context parameter is identified as consequential if it produces statistically different average user ratings across its values, for example, weather is consequential if user ratings are generally lower when it rains. However, such a classic application of *feature selection* weighs the context parameters separately and does not capture the causal dependencies among the consequential context parameters.

Collaborative filtering sidesteps the need to analyze and understand *exactly* what are the factors that go into making a certain user prefer a certain item. However, this also means that they cannot make recommendations on unrated items, cannot personalize their recommendations to a unique user, and cannot explain their recommendations easily. In contrast, content-based techniques use attributes of the items to make recommendations. This enables them to recommend unrated items to users with unique interests and also to provide explanations for all their recommendations [19]. A *context-aware content-based recommender* can further explain its recommendations in terms of the captured causal dependencies among context parameters specific to individual users. As such, we use this as our platform to address the challenges highlighted in Section 1.1. A formal definition of *context-aware content-based recommenders* is provided in Section 3.

Among the prior works that have used BN in context-aware computing, Gu et al. [16] add dependency and probability marks to the World Wide Web Consortium (W3C)-endorsed Web Ontology Language. Their *manually defined* context ontology can then be mapped to a BN. These prior works are motivated by the efficient probabilistic reasoning capability of a BN and its graphical superiority in representing causal dependencies among context parameters. Indeed, BN has many meaningful applications in context-aware systems, for example, the context-aware robotic aid for the elderly blind [20]. Unfortunately, all of these prior works do not discover the *user-specific* causal dependencies among context parameters by learning from the data.

Similarly, BN is not new to recommender systems. Breese et al. [21] propose a probabilistic collaborative filtering approach that learns BN from transactional data, with each node representing an item on sale or a *commodity*. Likewise, Ji et al. [3], [22] propose to learn BN from a customer’s shopping history, such that each network node represents one commodity. Commodities are then recommended based on probabilistic inference according to the last known shopping actions. Effectively, all these prior

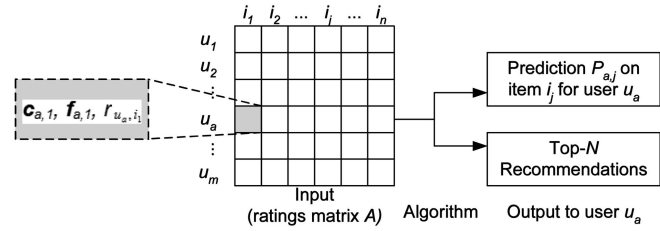


Fig. 1. The context-aware content-based recommendation process.

works learn BN to capture purchase patterns among commodities. A drawback is that this approach suffers from the *sparsity* problem, where data points are often far fewer than the commodities. More importantly, just the purchase patterns among specific commodities are learned from past transactions without exploiting those potentially useful causal dependencies among the context parameters.

In all of the above prior works, either the causal dependencies are manually crafted and, hence, not user specific or context is not exploited at all in the learned BN. Thus, they cannot identify the minimal context nor can they overcome missing and erroneous context inputs.

3 CONTEXT-AWARE CONTENT-BASED RECOMMENDER SYSTEMS

The goal of a context-aware content-based recommendation system is to predict the utility of an unrated item, for a user $u_a \in U$, based on her rated items I_{u_a} and the context in which those ratings, or *scores*, are recorded. The advices are typically presented to the user in two forms:

- **Prediction.** A numerical value $P_{a,j}$ that expresses the predicted utility of item $i_j \notin I_{u_a}$ for the user u_a . $P_{a,j}$ would be within the same scale (for example, from 1 to 10) as ratings given by u_a .
- **Recommendation.** A list of top- N unrated items that the model predicts u_a to like the most.

Fig. 1 shows a schematic diagram of the context-aware content-based recommendation process. The ratings matrix A records past ratings for items by users. A record of the rating by a user u_a on an item i_j is a triplet $\langle \mathbf{c}_{a,j}, \mathbf{f}_{a,j}, r_{u_a,i_j} \rangle$. This triplet comprises the vector of user context $\mathbf{c}_{a,j}$ and the vector of item attributes $\mathbf{f}_{a,j}$ for that interaction, as well as the given rating value r_{u_a,i_j} . For example, in restaurant recommendation, item attributes might include whether vegetarian food is available and also whether an eating place is crowded. User context might include whether the user is a vegetarian and whether he or she minds a crowded restaurant at the time of interaction.

Although collaborative filtering considers all m rows of user ratings to recommend for user u_a , here, we only consider the items that are rated before by user u_a . Each of these records provides an example for learning u_a ’s profile, comprising of the vector of user context inputs, the vector of item attributes inputs, and the recorded rating, or *score* value (the class variable).

From the data, the context-aware content-based algorithm computes for an unrated item i_j , $P_{a,j}(S_j = v | \mathbf{c}_{a,j}, \mathbf{f}_{a,j})$ for $v \in R^S$, where R^S are values that the score S can take (for example, v is one of values “1” to “5” for a five-point

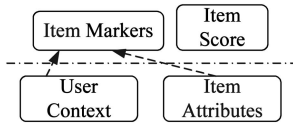


Fig. 2. A two-tiered context model.

rating scale). This is the posterior probability that a user u_a gives a score of v to an item i_j with context vector $\mathbf{c}_{a,j}$ and attribute vector $\mathbf{f}_{a,j}$. The algorithm computes this for each item, and it recommends items with the highest predicted ratings.

Based on this formalism of context-aware content-based recommenders, the objective of our work presented in this paper is to automatically discover, for user u_a , a *minimal set* that contains only context parameters important to u_a . In addition, we aim to harness the causal dependencies among context parameters to effectively overcome missing and erroneous context inputs.

4 A CONTEXT MODEL FOR EFFECTIVE LEARNING

As discussed earlier in Section 2.2, the time complexity of the BN learning algorithm is dependent on the number of network parameters. This is a concern as there are often far too many context parameters that are eligible for consideration in a recommender. Also, many context inputs are acquired from sources that can become intermittently unavailable and require alternatives to be used, especially in service-oriented frameworks (for example, [23], [24]). To achieve a compact and more manageable set of learning parameters, we propose the use of *domain-specific markers*, popular among existing clinical (especially cancer) research works.

In the context of recommender systems, a *marker* is a domain-specific factor that is considered by the system designer as suitable for evaluating an item. For instance, a marker for a restaurant recommender could be “*Is the restaurant open during the visit?*” because this condition would be a relevant concern for the *typical* users of that system. A marker can therefore be regarded as a Boolean function that takes in several specific user context and item attributes as parameters to gauge whether a certain condition of concern is satisfied by a particular item. Since the defined markers are typically far fewer than the relevant context parameters, learning on markers allows us to discover the minimal marker set from fewer examples. We can then identify only those context parameters that feed into this minimal marker set as being important to that user.

With reference to our context model in Fig. 2, an *item* refers to one of the many choices that are to be rated and possibly recommended, for example, a restaurant. A lower tier handles the *user context* and *item attributes*. The *user context* includes higher level context like the user’s preference for cleanliness, as well as more primitive situational context like the weather. An upper tier handles the *item markers* and the *item score*. An *item marker’s* value is computed using designer-specified heuristics based on the values of its associated user context and item attributes. The *item score* for an item is a rating value that is provided by the user to reflect the suitability of that item.

Our proposed two-tiered context model can take advantage of markers to reduce the number of learning parameters

when looking for the minimal context. In principle, the heuristics used to compute the marker values can be generated automatically. For instance, a similar user-adaptation problem has been addressed by bootstrapping the recommender with user profiles sampled from a probabilistic model built from prior knowledge [25]. In applications without defined markers, our two-tiered model applies directly on the context parameters. In such cases, we look for the minimal set of parameters and include the parameters connected to this minimal set as our second tier. Doing so effectively captures their causal dependencies, as we would be demonstrating via the Web page recommendation problem presented in Section 8. For the rest of this section, we shall describe our techniques within the context that markers are available for the recommender system, but the same proposed techniques are equally useful in cases where there are no markers.

4.1 Identifying and Learning on the Optimal Set of Predictors

We develop an iterative procedure for learning a BN with only the minimal set of context parameters. First, we learn with just the markers and the score to find the minimal set of important markers. Learning on all the markers initially, we retain those markers that connect to the user rating, or *score* variable, for a second learning. In this way, the procedure iteratively trims away markers until the remaining markers are all connected to the score variable. We can now include those context parameters and item attributes that feed into the minimal marker set as part of our learning parameters. The resulting BN learned using this procedure effectively captures the causal dependencies among the context parameters important to that user.

Our proposed context model eliminates the need to learn on the large set of context parameters in applications where a far smaller set of markers are available. Also, unlike in prior works that model specific commodities as network nodes, we learn a compact and explanatory BN to capture the user-specific causal dependencies among context parameters. As such, we can predict ratings even for new items, without having to add new nodes or relearn the network.

4.2 Performing Prediction Using Learned BN

Using the *Netica-J* Application Programmer Interfaces (API) [26], we predict the *score* with the learned network by feeding in the values of the user context and item attributes therefore allowing the beliefs of the marker values and, in turn, those of the item score to be updated. We then recommend the highest scoring items to the user. Invoking *Netica-J*’s belief updating mechanism for prediction is straightforward once *CaMML* has recorded the learned model in the *Netica* format. It involves the following steps:

- **Step 1.** For an unrated item i_j , it presents its current context vector $\mathbf{c}_{a,j}$ and attribute vector $\mathbf{f}_{a,j}$ for a user u_a to the learned BN.
- **Step 2.** Let the BN update all $P_{a,j}(S_j = v | \mathbf{c}_{a,j}, \mathbf{f}_{a,j})$, where v represents each of the values that the item score variable can take.
- **Step 3.** Predict, for the unrated item i_j , the score value v with the highest $P_{a,j}(S_j = v | \mathbf{c}_{a,j}, \mathbf{f}_{a,j})$.

Netica uses the fastest known algorithm for the exact general probabilistic inference in a compiled BN, known as

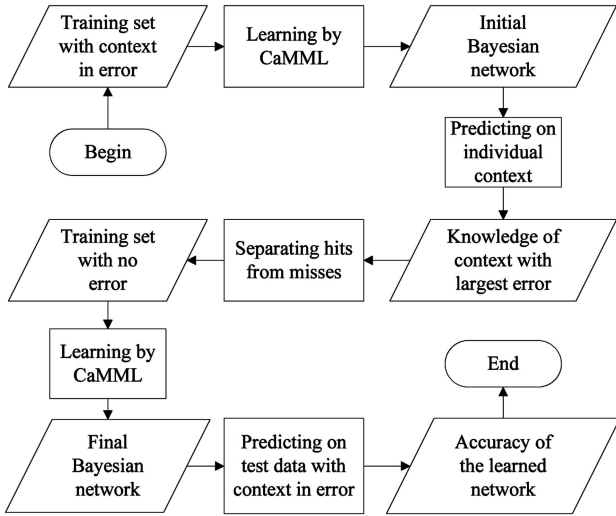


Fig. 3. Procedure for handling erroneous context inputs.

message passing in a join tree (“junction tree”) of cliques. For details of this algorithm, please refer to the software’s manual [26] and the more technical discussions in [27]. As belief updating is done once for each item in each round of recommendation, time complexity of each recommendation is $O(n)$, where n is the number of items. Empirically, we observe that the BN inference mechanism is very efficient. In the experiments, hundreds of predictions are completed within seconds on a PC.

5 HANDLING MISSING AND ERRONEOUS CONTEXT INPUTS FOR ACCURATE PREDICTION

5.1 Handling Missing Context Inputs

In most supervised learning techniques, including DT and the support vector machine, missing values in the test examples are handled by filling in the blanks. We can build a prediction model for each missing attribute to estimate its missing values. Otherwise, we can fill in missing values with aggregate values (for example, mean and mode) that are determined from the training set. Besides these, there are other sophisticated techniques in the statistical literature. One popular approach is *Multiple Imputation* [28], which involves the processes of imputation, analysis, and pooling. A good reference on this and other statistical techniques for handling missing data is [29].

Although all of the above techniques require manipulation of data to prepare them for prediction, in a BN, partial evidence can be entered during prediction, and the network would compensate for the unspecified inputs using its captured causal dependencies.

5.2 Handling a Single Error Context

Our procedure to handle erroneous context inputs is shown in Fig. 3. The training set comprises the parameters identified through our iterative learning procedure in Section 4.1. The error-handling procedure learns an initial BN from this training set with context in error. Predicting on the training set with this network, the procedure iterates over the individual context to identify the erroneous context, such that the percentage of misclassified training

examples for each context is its estimated error rate. The procedure retains the correctly classified examples, or the *hits*, and learns a more accurate BN on these examples. This final BN then takes into account the estimated error rate when predicting on unseen examples.

The subprocedure for automatically identifying a single erroneous context and cleansing the training set is summarized into Algorithm 1. First, a process identifies the context that is in error and estimates its error rate; second, the training set is “cleansed” by discarding training examples that are misclassified for that context. The time added by steps 1 and 2 is linear in the size of the training set, K , due to repeated predictions on the examples. Letting the number of context parameters be p , the complexity of these steps is $O(pK)$. This is efficient in practice as K is usually small (the typical user would seldom have seen too many examples). With reference to Section 2.2, *CaMML* learning has the worst-case time complexity of $O(N!)$, where N is the number of nodes in the network. The overall complexity of Algorithm 1 is therefore $O(pK + N!)$, with the bulk of the time taken up by the *CaMML* learning from the cleansed data.

Algorithm 1. Handling a Single Unidentified Erroneous Context with an Unknown Error Rate

Input: Initial BN ($BnIn$) that is learned on the original training set with errors ($dataIn$).

Output: Final BN ($BnOut$) learned on cleansed data ($dataOut$), the error context ($errCtx$) and its estimated error rate ($errEst$).

Step 1: Identify $errCtx$ and find $errEst$

Set the first context, C_1 , as $errCtx$.

Treat C_1 as missing and predict its values in all of $dataIn$ using $BnIn$.

Set $errEst$ to the proportion of $dataIn$ for which C_1 is misclassified ($P_{err}(C_1)$).

for each remaining context C_i do

Treat C_i as missing and predict its values in all of $dataIn$ using $BnIn$.

if $P_{err}(C_i) > errEst$ then

Set $errCtx$ to C_i .

Set $errEst$ to $P_{err}(C_i)$.

Step 2: Cleanse $dataIn$ of suspicious examples

for each example in $dataIn$ do

Treat $errCtx$ as missing and predict its value using $BnIn$.

if the predicted value of $errCtx$ differs from its labeled value then

Remove this example from $dataIn$.

{If resources permit, **Step 2** simply recalls and removes examples misclassified earlier from **Step 1**.}

Step 3: Complete the error context identification and data-cleansing procedure

Learn $BnOut$ on cleansed training set $dataOut$ from

Step 2.

return $BnOut$, $dataOut$, $errCtx$, and $errEst$.

Training BN on “cleansed” data makes sense as we expect better models when the training set contains fewer error examples. This intuition is an important motivation

behind the traditional works on outlier detection methods [30], as well as recent works that have extended the notion of “noise” to include examples that are irrelevant or weakly relevant [31].

5.2.1 Identifying Error Context and Estimating Error Rate

In the data mining field, dealing with noisy or erroneous data usually involves identifying training examples with inconsistent values as *outliers* and removing these from the training set so that we learn an improved model [30]. In general, the outlier detection methods (for example, distance-based, density-based, and clustering-based) assign an outlier score to each example and remove those examples above some threshold [31]. For instance, Brodley and Friedl [32] employ majority voting to discard the mislabeled training examples.

We take an inspiration from the outlier detection methods—we build a prediction model using the entire training set to make predictions on each context in turn, so that we can identify those examples where the predictions differ from their observed values as the outliers. For a start, we assume that only one of the context important to the user is in error, but we do not know which one it is or what its rate of error is. All we are initially presented with is the set of training examples that are noisy. Hence, we begin by using *CaMML* to learn an initial BN, which we use to predict in turn on each context to estimate their individual error rates.

5.2.2 Cleansing the Training Set

Having estimated the individual error rates, we identify the context with the largest error as the one that is erroneous. We now have its training error rate (proportion of training examples for which the initial learned BN prediction model misclassifies the erroneous context) as our “knowledge of context with largest error” (Fig. 3). Each correctly classified training example is termed as a *hit*, and we separate these *hits* from the *misses* to get a reduced set of training examples. We cannot ascertain in practice if this reduced set of training examples is in fact error free. We are trying to remove as many of those examples with possibly erroneous context inputs, so that our BN learned on the “cleansed” training set can better reflect the actual causal dependencies among context parameters.

5.2.3 Predicting on Score Using Likelihood Findings to Account for Uncertainty in Context

After obtaining the cleansed set of training examples (the *hits* from the initial BN, that is, those correct predictions on the identified error context), we learn a final BN for predicting the user ratings, or the *scores*, of unrated items. We can verify the final BN’s accuracy by predicting the user ratings for an unseen set of test examples. This test set should be noisy too, with similar noise characteristics as the original training set.

When predicting the user rating, BN can take into account the estimated error rate, or the *uncertainty*, in the erroneous context. This is an important capability not available in other classifiers like DTs. In the normal

inference scenario, where we are not aware of possible errors in the context inputs, we enter each input from the test example as *specific evidence*. For example, suppose that the input values for the context parameter “weather” can be “hot,” “rainy,” or “fine.” Now, if we enter an input of “fine” as *specific evidence*, a likelihood probability of 1.0 would be used for “fine,” whereas 0.0 would be used for the other two values. However, when there are errors in the inputs of a context, entering these erroneous inputs as specific evidence is likely to result in wrong inferences. Fortunately, a BN allows us to specify such potentially erroneous evidence as *likelihood findings* instead of specific evidence.

Let the observation on the “weather” context be O . The *likelihood finding* for O is given by

$$\begin{aligned} &\{\text{prob}(O \mid \{\text{weather} = \text{"hot"}\}), \\ &\text{prob}(O \mid \{\text{weather} = \text{"rainy"}\}), \\ &\text{prob}(O \mid \{\text{weather} = \text{"fine"}\})\}. \end{aligned}$$

For the example where we observe “weather” as “fine,” suppose we are also aware that this observation carries an estimated error probability of e . This means that each of the inputs for “weather” might be wrong $e * 100$ percent of the time. This error probability for the erroneous context can be estimated using the training error rate discovered by our error-handling procedure. The likelihood finding for $O: \{\text{weather} = \text{"fine"}\}$ would then be given by $\{e * 0.5, e * 0.5, 1.0 - e\}$. This is because O may be wrong with a probability of e , and “weather” has only three states, so the probability of inputting “weather” as “fine” when it is actually “hot” or “rainy” is $e * 0.5$. Depicting S as the number of states of the erroneous context, we can generalize this formulation of the likelihood finding from the estimated error probability e as follows:

$$\begin{aligned} &\text{Likelihood for the observed state} = 1.0 - e, \text{ and} \\ &\text{Likelihood for any other state} = e * [1.0 / (S - 1)]. \end{aligned}$$

As the effect of entering an error rate of zero for each context is equivalent to entering a specific evidence, the above likelihood formulation is appropriate even when the evidence is known to be certain. As explained in [17], we should consider the prior probabilities within the training set when computing the likelihoods. However, as our restaurant recommendation data set covers all states of each context without bias, the input priors are roughly uniform. Results in Section 7.3 show that, by taking account of the uncertainty in the erroneous context inputs, BN overcomes erroneous context inputs to predict accurately on unseen examples.

5.3 Handling Multiple Erroneous Context

So far, we have assumed the knowledge that a single context is in error. In practice, we do not know how many context might be erroneous, so we need to make sure that our procedure is not restricted by this assumption. Our proposed procedure factors out the negative effects of the highest error context one at a time until the estimated error in the remaining context is below some defined threshold. This error handling procedure has been summarized into

Algorithm 2. Since steps 1 and 2 iterate over the set of p context parameters, the worst-case time complexity of this procedure is $O(p^2K + (p + 1)N!)$, where K is the number of training examples, and N is the number of nodes in the network. Most of this complexity ($O((p + 1)N!)$) is actually due to the *CaMML* learning. As such, our iterative learning procedure (Section 4.1), which finds the minimal set of important parameters, can significantly improve the efficiency of error handling.

Algorithm 2. Handling Multiple Unidentified Erroneous Context with Unknown Error Rates

Input: Initial BN ($BnIn$) learned on the original training set ($dataIn$), and the minimum threshold for an error context ($errMin$).

Output: Final BN ($BnOut$) learned on cleansed data ($dataOut$), the error context ($errCtxs$) and their estimated error rates ($errEsts$).

Step 1: Identify $errCtxs$ and find $errEsts$

Identify the top most erroneous context C_{top} as per Algorithm 1.

Add C_{top} to $errCtxs$ and set $errEsts$ to error rate of C_{top} .

while minimum $errEsts \geq errMin$ **do**

Estimate likelihoods using $errEsts$.

Entering the values of $errCtxs$ as likelihoods, identify the next most erroneous context C_{next} .

Add C_{next} to $errCtxs$ and set $errEsts$ to current error rates of $errCtxs$.

Step 2: Cleanse $dataIn$ of suspicious examples

while mean error on $errCtxs$ is significant under a statistical t-test **do**

Remove the training examples that misclassify the context in $errCtxs$ with the largest error.

Learn on this partially cleansed data for the next iteration.

Step 3: Complete the error context identification and data-cleansing procedure

Learn $BnOut$ on cleansed training set $dataOut$ from

Step 2.

return $BnOut$, $dataOut$, $errCtxs$, and $errEsts$.

5.3.1 Identifying Error Context and Estimating Error Rates

Our error-handling procedure described in Section 5.2 is extended as follows: First, we iteratively predict on the important context with all inputs entered as *specific evidence*, and we take note of the error rate for the single most erroneous context. We again iterate over all the context, but now we specify all inputs for that most erroneous context as *likelihood findings* computed based on its noted rate of error. The next most erroneous context is identified, and its error rate, as well as that of the most erroneous context, are noted. Then, we iterate over all the context using the likelihood findings for these two most erroneous context. This process is repeated until the estimated error rate of the next most erroneous context falls below a specified minimum error threshold. In this way, the procedure identifies the erroneous contexts in sequence based on their estimated error rates. Although

contexts with the largest error rates may not be necessarily erroneous, this is a reasonable basis for identifying the contexts in order of how likely they are to be in error.

This iterative error discovery process is analogous to the *Principle Component Analysis* (PCA). We first identify the single most erroneous context, remove its dismal effects from the training set by inputting it with uncertainty in the next round, then we look for the second most erroneous context, and so on. Analogous to Henry Kaiser’s recommendation for retaining only principle components with an eigenvalue exceeding 1.0, our procedure stops looking for erroneous context when the estimated error in the next most erroneous context drops below some threshold. As users generally do exhibit some reasonably small degree of natural randomness in their behaviors, this minimum error threshold allows our procedure to filter out such background noises.

5.3.2 Cleansing Training Set by Removing Examples with Errors in One or More Context

Now, we have to decide upon the amount of “data cleansing” before we commence the final learning. Under the previous assumption of a single error context, we simply remove wrong predictions on that context from the training set. Now, with multiple context in error, we have the alternatives of either to remove misses for all the error context identified or remove just misses on the most erroneous context, learn a network on the hits, and then evaluate the model on an unseen validation set to decide whether to proceed with removing misses on the second most erroneous context, and so on. We note that removing all the training misses for all the error context presents the danger of *overfitting* as the size of our training set dwindles, so the second approach provides us with a means to cross validate on the given examples in order to avoid overfitting the data.

We implement the second approach above with certain modifications, so as to overcome the two limitations of having to set aside a validation set and having to predefine a fixed threshold for deciding when to stop cleansing. In the first iteration of data cleansing, we perform a statistical t-test [33] to check if the mean estimated context error is significant at some desired level (for example, 0.01 or 0.05). If so, we remove those training examples that have misclassified the single context with the largest error, learn on this partially cleansed training set for the next iteration, and so on. We stop cleansing when the mean estimated error is not significantly different from 0.0 in any iteration. The experimental results for this procedure are presented in Section 7.5.

6 A RESTAURANT RECOMMENDER APPLICATION

We demonstrate how our proposed approach works in a restaurant recommender application named “*RR.II*” (our second restaurant recommender). This application learns automatically from the data the underlying causal dependencies among parameters as a BN for a specific user and then predicts the scores for candidate restaurants using this learned network. To make the application realistic, a rich set of real-world context parameters, restaurant attributes, and

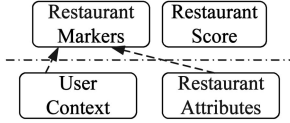


Fig. 4. Context model of our restaurant recommender application, *RR-II*.

user considerations is used to generate data examples that represent multiple users. During the evaluation, only these data examples are presented. This enables us to evaluate whether our proposed approach is indeed useful for reliably discovering from the data the unique personal characteristics of different users and exploiting these to maintain robust predictions.

Fig. 4 shows the context model of *RR-II*, which implements the model in Fig. 2. We define a total of 26 user context parameters, 30 restaurant attributes, and 21 markers. In a typical mobile recommender, the number of contexts is much greater than the number of markers, so our advantage of not learning on all the context is even more significant in practice. User context includes relevant aspects of the user’s current *situation* (for example, “*weather*”), as well as the relevant aspects of the user’s *preferences* (for example, “*cleanliness*”). Restaurant attributes include “*category*” and “*is clean*.” The markers are defined to represent typical user concerns like whether a restaurant is “*open during visit*.” In all, 15 restaurants have been modeled for selection.

Each marker is defined as a boolean variable. For example, a restaurant is either “open” or “not open” at the time of visit. We develop a software simulator program that generates examples spanning across all of the possible values of each parameter while adhering to a user-specific causal model. The corresponding causal models are defined on user context and not on system markers, so as to accurately reflect how a typical and logical user would consistently behave.

For example, our first model (Table 1) represents a user who knows how to dress for different occasions. A set of causal rules can be used to model how such a user would dress up for different eating places. For example, she dresses consciously when visiting a club; otherwise, she prefers to stay casual when eating in a café or canteen. In addition, this user prefers to eat in open-air canteens for informal occasions because fresh air improves her appetite. The corresponding causal model that represents this user, User 1, is illustrated below.

Model 1. Causal Dependencies among Context for User 1

```

if UP.category == "café" then
  US.attire = "casual"; UP.ventilation = "aircon";
else if UP.category == "club" then
  US.attire = "formal"; UP.ventilation = "aircon";
else if UP.category == "canteen" then
  US.attire = "casual"; UP.ventilation = "nonAircon";
else if UP.category == "dunCare" then
  US.attire = "dunCare"; UP.ventilation = "dunCare";
  
```

For each recommendation cycle, the user context values and restaurant attribute values are compared via heuristics

TABLE 1
User Preference Rules and Models

User	Preference Rule	Causal Model
1	M3, M11, M12	Model 1
2	M5, M10, M12, M15	Model 2
3	M10, M12, M14	Model 3
4	M5, M10, M12, M21	Model 4
5	M3, M5, M12	Model 5

to determine whether a marker should take on a value of 1 (“*satisfied*”) or 0 (“*not satisfied*”). Examples of heuristics for deriving three of the defined markers, M3, M11, and M12, are given below. The two prefixes of “*US.*” and “*UP.*” refer to a *user situation* and a *user preference* context, respectively, while the prefix “*RA.*” refers to a *restaurant attribute*.

Heuristic M3 *userAttireIsAppropriate* (M3)

```

if RA.attireRequirement == "formal" then
  if US.attire == "formal" then valueof(M3) = "yes";
  else if US.attire == "casual" then valueof(M3) = "no";
else if RA.attireRequirement == "none" then
  valueof(M3) = "yes";
  
```

Heuristic M11 *matchesIsAirConditionedPref* (M11)

```

if UP.ventilation == "aircon" then
  if RA.isAirConditioned == "yes" then
    valueof(M11) = "yes";
  else valueof(M11) = "no";
else if UP.ventilation == "nonAircon" then
  if RA.isAirConditioned == "yes" then valueof(M11) = "no";
  else valueof(M11) = "yes";
else if UP.ventilation == "dunCare" then
  valueof(M11) = "yes";
  
```

Heuristic M12 *isOfDesiredCategory* (M12)

```

if UP.category == RA.category then
  valueof(M12) = "yes";
else valueof(M12) = "no";
  
```

Fig. 5 shows the defined causal dependencies corresponding to each of these three markers of M3, M11, and M12. Based on the corresponding derived marker values, we simulate the user rating given for a restaurant, or the restaurant’s *score* value, using the formula below:

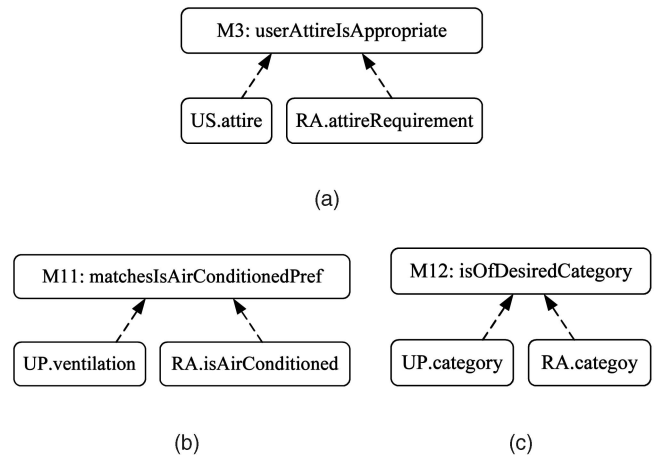


Fig. 5. Causal dependencies for markers (a) M3, (b) M11, and (c) M12.

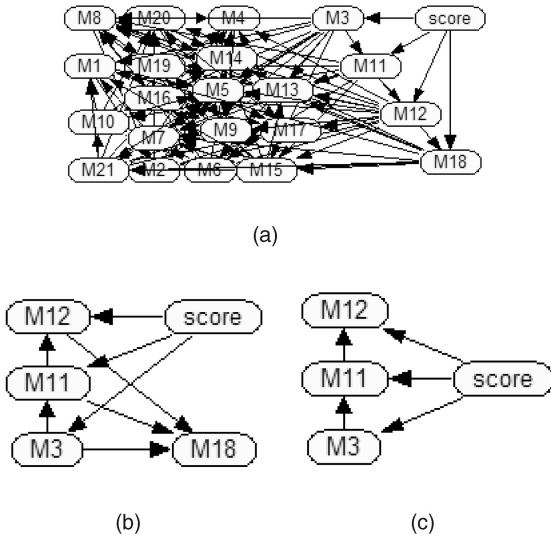


Fig. 6. Discovered BN in each round of learning. (a) In round 1. (b) In round 2. (c) In round 3.

$$score = \frac{\text{No. of Important Markers with value "1"}}{\text{No. of Important Markers}}$$

This formula reflects the behavior of a logical user who rates each of the presented restaurants according to their overall performance on a minimal set of markers that are important to him or her. The equal weighing of important markers is just an example of the many possible scoring functions for a user, which of course can be much more complex and flexible for an idiosyncratic user. In fact, in the actual application scenario, there would not be any explicit scoring function, but the users would instead have stated their preferences among items based on some given scales. As in the actual scenario, just the data examples are presented for analysis in our evaluations, and we have to learn the scoring model for each user based only on these past records. Each example presented in our experiments thus comprises the generated user context and attribute values, the marker values derived using the heuristics, and the scores for the restaurants.

7 EXPERIMENTAL VALIDATION

7.1 Learning the Minimal Set of Important Parameters

The first set of experiments aims to verify that the minimal set of markers that is truly important to a certain user could be effectively recovered in the BN learned from observation data. To generate observation data, we define a set of *preference rules* to represent different user logics in considering markers when scoring restaurants, and a set of *causal models* to state the causal interdependencies among the corresponding important context. The preference rules and their corresponding causal models are listed in Table 1. For example, according to the preference rule for *User 1*, the three markers “*userAttireIsAppropriate*,” “*matchesIsAirConditionedPref*,” and “*isOfDesiredCategory*,” labeled as “*M3*,” “*M11*,” and “*M12*,” respectively, are equally important.

For each rule, we learn using *CaMML* from 1,000 observations and retain only those markers that are directly

connected to the *score* node in the resulting network. We then learn again on the same set of examples, but with all other markers removed. This process of automatic learning and pruning is repeated until a learned model has all of its markers directly linked to the score.

From our first round of learning on the training set corresponding to User 1, we obtain the BN that is shown in Fig. 6a. On the right-hand side of this figure, we see that only the four markers of M3, M11, M12, and M18 are connected to score. To verify if all these four markers are truly important for scoring, we retain just these four markers and the score variable to perform a second round of learning. From the resulting network as shown in Fig. 6b, only M3, M11, and M12 (the three nodes on the left) are directly connected to the score variable. We therefore retain only the score and these three nodes for our third round of confirmation learning. The resulting network in Fig. 6c shows that *all* the three markers that are supposed to be important to the first user (as stated in Table 1) are still directly connected to the score.

To evaluate the system’s performance in discovering the correct minimal set of markers, we have adapted the *F-measure* from the information retrieval field using the following definitions:

$$\begin{aligned} \text{Recall} &= \frac{\text{No. of Important Markers Connected to Score}}{\text{Number of Important Markers}} \\ \text{Precision} &= \frac{\text{No. of Important Markers Connected to Score}}{\text{No. of Markers Connected to Score}} \\ \text{F-measure} &= \frac{2(\text{Recall})(\text{Precision})}{(\text{Recall}) + (\text{Precision})} \end{aligned}$$

These modified definitions accurately reflect our context of analysis where a positive hit refers to an important marker being directly connected to the score node in the learned model. The *recall* value represents the proportion of truly important markers that are identified, whereas *precision* measures the proportion of those markers marked as important that are truly important. The *F-measure* is then the weighted average, or the harmonic mean, of *recall* and *precision*. Therefore, together, they measure how well the set of important markers matches those connected to the *score*.

The results of learning from the data sets are given in Table 2. The learned BNs consistently yield an F-measure value of 100 percent by the second round for users 1, 2, 3, and 5, whereas a 100 percent measure is obtained after just one round of learning for user 4. These results show that our proposed approach reliably identifies the minimal set of parameters important to the user.

7.2 Prediction with Missing Context Values

In our second set of experiments, we employ the minimal sets of important markers identified in our earlier experiments to verify that the prediction accuracy on the *score* remains high despite missing context values. For each user, we prepare the same earlier set of 1,000 observations to perform five rounds of two-fold cross validation. In the first fold of each round, 500 examples are randomly chosen for learning the dependencies among context, and the remaining examples are used to test the prediction accuracy on the score. In the second fold, we now train on the held-out set

TABLE 2
Minimal Set Learning Performance

User	Round	Recall	Precision	F-measure
1	1	1.00	0.75	0.86
	2	1.00	1.00	1.00
2	1	1.00	0.67	0.80
	2	1.00	1.00	1.00
3	1	1.00	0.50	0.67
	2	1.00	1.00	1.00
4	1	1.00	1.00	1.00
5	1	1.00	0.60	0.75
	2	1.00	1.00	1.00

and use the previous training set for testing. In each fold, we learn a BN on the training set consisting of just the score variable and the minimal set of markers together with their corresponding user context and restaurant attributes. Predictions on the test set are then performed using the Netica-J API as it has been described previously in Section 4.2.

In each validation fold, prediction accuracies are recorded for the baseline where all of the context are available and under scenarios where a context value (and, hence, its dependent marker value) is missing in the 500 test examples. We compare the results to the corresponding prediction accuracies that are achieved with the *J4.8 DT* classifier as implemented in the *Weka* knowledge analysis tool [34], using the default options given in *Weka's Explorer* GUI.

We first investigate if the learned network comprising just the minimal set of markers and the score (the *upper tier* in our model of Fig. 2) is sufficient to handle missing context values. Over five rounds of two-fold cross validation, we observe the prediction accuracy on score when all the contexts are available and when the contexts (and, hence, their dependent markers) go missing one at a time. The results are shown in Table 3a. Both the BN and DT suffer a greater than 20 percent drop in accuracy with missing context values, suggesting that the important context and their user-specific dependencies have to be captured directly in our learning process.

Table 3b summarizes the accuracies when the important context and restaurant attribute values from the same examples are incorporated for learning. Clearly, the BN and the *J4.8 DT* perform equally well when the data is complete, that is, when no important context values is missing, but we observe that the BN significantly outperforms the DT to maintain a 100 percent prediction accuracy under the imperfect scenarios of missing context inputs. The results show that the causal dependencies among the various context elements are indeed the key to the effective compensations by the learned BN. The fewer markers enable the minimal set of important parameters to be quickly and reliably identified before these relevant causal dependencies are accurately captured through automatic learning from the data.

7.3 Prediction with a Single Erroneous Context

In all of our previous experiments, *noise* or erroneous context inputs are absent from both our training and our test sets.

TABLE 3
Average Prediction Accuracy with Missing Context Values (Percentages)

User	Bayesian Network (%)		Decision Tree (%)	
	Complete	Missing	Complete	Missing
1	100	76.8	100	76.8
2	100	80.5	100	77.5
3	100	70.5	100	66.3
4	100	86.2	100	68.6
5	100	79.8	100	75.9
Avg.	100	78.8	100	73.0

(a)

User	Bayesian Network (%)		Decision Tree (%)	
	Complete	Missing	Complete	Missing
1	100	100	100	78.4
2	100	100	100	81.7
3	100	100	100	64.5
4	100	100	100	77.8
5	100	100	100	77.5
Avg.	100	100	100	76.0

(b)

Complete: data is complete; Missing: with one context missing. (a) Learning and predicting using only the upper tier. (b) Learning and predicting using a cross-tier approach.

Here, in our third set of experiments, we investigate whether BN learning can indeed reliably identify the same minimal sets of important markers even in the presence of noisy context. Furthermore, we evaluate if the prediction accuracy on the *score* can still remain high despite the errors in important context values. For each of the users modeled in Table 1, we have prepared two sets of 1,000 examples for n rounds of two-fold cross validation, where n is the number of contexts that is important for each user. In each round, we simulate the scenario where one of these n contexts significant to the user is erroneous, but the system does not know which is the erroneous context or what is its rate of error. One thousand observations embedded with errors are used for learning, and another set of 1,000 unseen test cases carrying similar errors are used to validate the resulting prediction accuracy on the score.

Within each fold, we first learn a BN on the training set consisting of just the score variable and the set of markers. Through the same stepwise elimination process as described previously in Section 7.1, we identify the minimal set of markers that are important for each user. The optimal set of learning parameters then consists of the score variable, together with these important markers and their corresponding user context and restaurant attributes. With this information in hand, we can now evaluate our proposed procedure (Algorithm 1) in Section 5.2 for handling a single unidentified erroneous context that has an unknown rate of error.

With reference to the same user models that are described in Table 1, our observations for User 1 are summarized in Table 4. As before, we compare the average prediction accuracies for our BN-based approach to the results achieved with the *J4.8 DT*.

TABLE 4
Average Prediction Accuracy for User 1 (Percentages)

Fold	Err-Ctx	Without data cleansing		With data cleansing	
		DT	BN	DT	BN
1	UP1	86.03	91.74	86.03	100.0
	UP3	79.55	89.25	79.55	100.0
	US4	97.87	98.45	100.0	100.0
	Average	87.82	93.15	88.53	100.0
2	UP1	85.06	94.01	85.06	100.0
	UP3	78.05	88.76	78.05	100.0
	US4	98.09	99.61	100.0	100.0
	Average	87.07	94.13	87.70	100.0

Error rate of 30 percent is used in all experiments; Err-Ctx: context in error; DT: J4.8 DT; BN: BN, using likelihood estimates.

We observe that the DT and BN exhibit comparable prediction accuracies when we treat the erroneous context values as specific evidence. However, Table 4 shows that prediction using the BN on the same test set after taking into account the error rates in context consistently yields significantly better results than the DT. This confirms that our emphasis on discovering error rates for prediction is indeed sound. Furthermore, results obtained when we repeat the learning on cleansed data show that BNs trained on cleansed data perform better than their counterparts trained on uncleaned data sets on all occasions.

In Fig. 7, we compare the average prediction accuracies of BN against DT over all of the modeled users when one context is in error at a time. We observe that, even at the high error rate of 45 percent, the BN maintains its accurate predictions unlike the DT. This is because our procedure has reliably identified the erroneous inputs and has presented their values to the BN as likelihood estimates during inferences, such that these could then be compensated by the other presented input values. The strength of our approach thus stems from its ability to reliably capture the uncertainty in the erroneous context values in its estimation of their likelihoods. A baseline of DT induction from the original uncleaned data is added for comparison. The results show that both the BN and the J4.8 DT benefit from cleansing, and that, by using the discovered training errors on a context as an estimate for its rate of error, our procedure effectively derives its likelihoods to achieve higher accuracy on erroneous data.

7.4 Prediction Performance under a Small Number of Training Examples

We now evaluate the performance of our proposed error-handling procedure from Algorithm 1 under a smaller number of training examples. Specifically, we compare the average prediction accuracy of a BN against a J4.8 DT when an important context suffers from various rates of error. For each of our five modeled users, we reserve the last 100 of their examples for testing. In each trial, we randomly select k of the remaining examples for training, where k ranges from 10 to 60 examples. We present the average results from 10 such trials in Fig. 8.

With reference to Fig. 8, we observe that the advantage of our approach based on the BN is related to the number of training examples. Using a statistical paired sample t -test

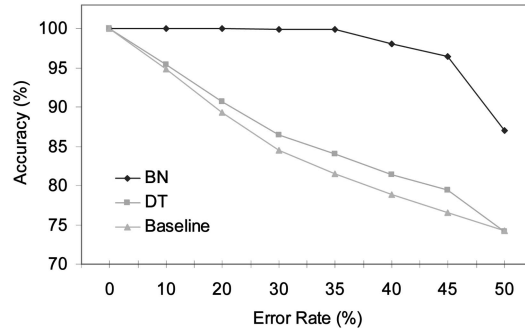


Fig. 7. Average prediction accuracy with a single context in error. Baseline is DT learned on uncleaned data.

[33] for the difference in mean values, we confirm that our method significantly outperforms the DT at the 0.05 level when there are at least 30 training examples, and that we get an increasingly superior performance beyond 40 examples. The BN learning is inferior only when the training set is restricted to a size of only 10 or 20 examples. This observation is consistent with the *CaMML*'s authors expectation that *CaMML* would not perform as well as simpler machine learning methods like DTs when the amount of data is so small that it is unable to find the correct causal model during its search through the model space [35].

Our results show that when there are erroneous context inputs, the BN outperforms the J4.8 DT as more examples become available. This is a significant advantage because within the mobile recommender environment, erroneous context inputs are the norm. It is hence important that our method effectively overcomes these after a prolonged use. Earlier results (Fig. 7) have shown that with ample examples, our method significantly outperforms the J4.8 DT.

It is interesting to note that data cleansing appears to have a negative effect on the J4.8 DT in these experiments, where the number of training examples is small. With reference to the plots in Fig. 8, it appears that any further elimination of suspicious examples from an originally small set of training examples can in fact make it harder to discover meaningful patterns. However, we expect data cleansing to have a positive effect on performance given sufficient training examples, as evident from the convergent trend for lines "DT" and "Baseline" in all the plots as the number of training examples goes up to 60. This is in agreement with earlier results from Section 7.3, where the benefits from data cleansing with ample training examples are clearly demonstrated.

Our approach outperforms the J4.8 DT as the number of training examples increases. However, BN learning using the *CaMML* is less effective when the number of training examples is so small that the correct causal model cannot be identified [35]. As users are generally reluctant to rate many, say more than 30 examples, various machine-learning techniques designed to maximize the utility from the smaller data sets can be adopted in practice. One popular approach, *semisupervised learning* [36], selectively labels (for example, through the use of statistical Expectation Maximization (EM)) the unlabeled (unrated) examples to augment the labeled points. Another popular approach

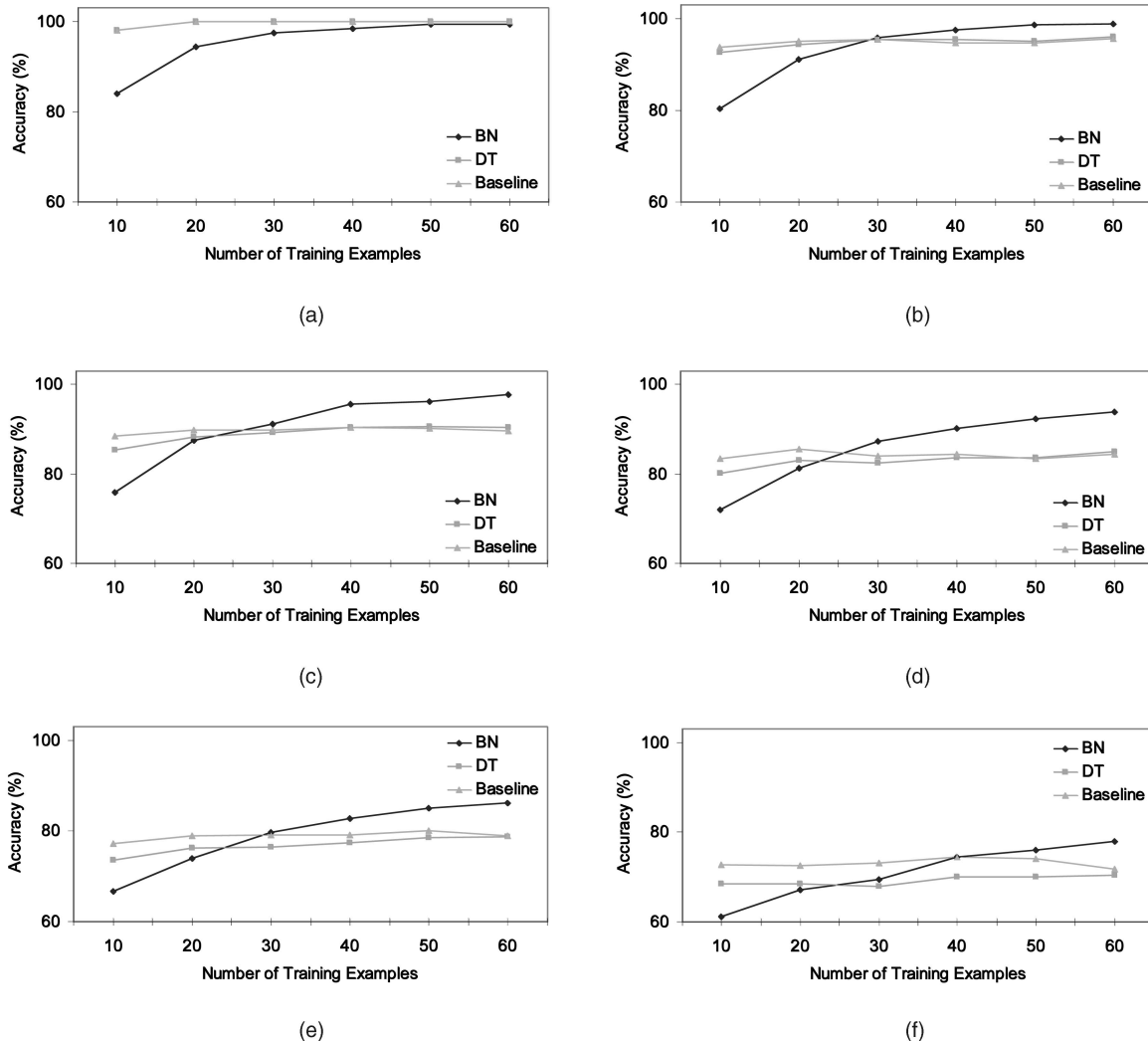


Fig. 8. Average prediction accuracy as the context error rate varies. Baseline is DT learned on uncleaned data. (a) Error rate 0 percent. (b) Error rate 10 percent. (c) Error rate 20 percent. (d) Error rate 30 percent. (e) Error rate 40 percent. (f) Error rate 50 percent.

that can be explored is that of *active learning* or *sample selection* in which training examples are acquired incrementally, and the system attempts to use what it has already learned to select only the most informative new examples for the user to rate [37]. Specific techniques for applying each of these approaches have been developed, and these techniques have been shown to significantly reduce the labeled examples required for real-world problems [19].

7.5 Prediction with Multiple Erroneous Context

In previous experiments with errors in context values, we assume the knowledge that just a single context is in error. To validate our procedure with an unknown number of error context, we simulate a sixth user with eight important markers (Table 5). We conduct a two-fold cross validation with three context having errors at different rates unknown to the learning system.

We observe that the correct minimal set of markers is identified in both validation folds. Applying our Algorithm 2 from Section 5.3 to the data with errors, we are able to identify all the three erroneous context and to deduce close estimates of their error rates. The evolution of our

model at each step of our error-discovery procedure is traced in Table 6. Error discovery stops when the observed error rate of the next most erroneous context falls to 0.0.

In Table 7, we compare the performance of the methods when just the misses for UP5 (the “most erroneous” context) are removed, when misses for either UP5 or UP3 (the two most erroneous context) are removed and also when the misses for all three identified error context of UP5, UP3, and UP2 are removed from the training set. The results confirm that test prediction accuracy increases steadily as suspicious examples are cleansed from the training set. The mean error is significant at the 0.05 level (one-tailed, that is, whether mean error is larger than 0.0) only for the first step where no example has yet been removed. As such, the results suggest

TABLE 5
User Preference Rule and Model for User 6

User	Preference Rule	Causal Model
6	M3, M5, M10, M11, M12, M14, M15, M21	Model 6

TABLE 6
Results of Error Discovery for User 6

Fold	Step	Err-Ctx added	Err-Ctx found	Err-Rate est.
1	0	UP5	UP5	0.321
	1	UP3	UP5, UP3	0.321, 0.212
	2	UP2	UP5, UP3, UP2	0.321, 0.212, 0.099
	3	-	UP5, UP3, UP2	0.319, 0.210, 0.099
2	0	UP5	UP5	0.307
	1	UP3	UP5, UP3	0.307, 0.201
	2	UP2	UP5, UP3, UP2	0.298, 0.201, 0.09
	3	-	UP5, UP3, UP2	0.298, 0.201, 0.09

Err-Ctx added: error context added in this step/iteration; *Err-Ctx found*: the error context found so far; *Err-Rate est.*: estimated error rates for use in the next step.

the removal of just the single most erroneous context (UP5) to arrive at the final model. From Table 7, this yields an average test accuracy exceeding 90 percent, compared to around 70 percent for the DT.

Fig. 9 compares the prediction performance of the methods over all of the other modeled users in Table 1. Fig. 9a shows the results when two contexts (unknown) are in error, whereas Fig. 9b shows the results with three erroneous contexts. As expected with the larger proportion of context in error, performance deteriorates across all of the methods. However, as per our observations in Section 7.3, the DT and BN benefit from data cleansing, albeit this benefit reduces as more contexts are in error. These results suggest that, by accounting for the uncertainty in the evidence, our procedure outperforms the DT even when multiple contexts are in error.

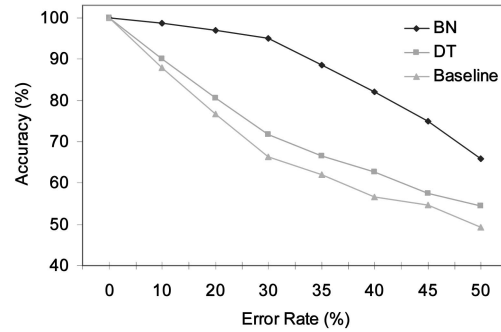
8 FURTHER VALIDATION: PERSONALIZED WEB PAGE RECOMMENDATION

In this section, we extend our experiments to a real user recommendation data set known as the Syskill and Webert Web Page Ratings data set [38] (available from the University of California (UCI)). This data set comprises the HTML sources of Web pages and their corresponding ratings (“hot” and “medium/cold”) from a real user. Seventy Web pages on the subject of “Goats” are used. The attributes are English words that are extracted from these Web pages, excluding those words from the standard SMART list of stopwords [39]. The original study [40] has reported the average prediction accuracy, that is, the percentage of misclassified test examples, for algorithms including naive Bayesian [41], nearest neighbor [41], perceptron nets [42], and ID3 DTs [43]. Based on this data set, we explore if causal dependencies among word

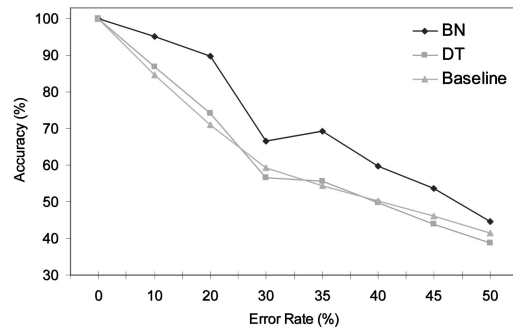
TABLE 7
Prediction Performance for User 6

Ctx_for_cleansing	Avg. acc. (%) for DT	Avg. acc. (%) for BN
UP5	69.81	92.44
UP5, UP3	71.92	98.12
UP5, UP3, UP2	80.17	100.0

Ctx_for_cleansing: context based on which misses are removed; *Avg. Acc. (percentages)*: average accuracy on score; *DT*: J4.8 DT; *BN*: BN, using likelihood estimates.



(a)



(b)

Fig. 9. Average prediction accuracy when there are multiple error contexts. Baseline is DT learned on uncleansed data. (a) With two error contexts. (b) With three error contexts.

attributes can be effectively discovered and exploited by our proposed BN-based mechanism for predicting the user ratings of Web pages.

Following the setup described in [40], we train J4.8 DT and BN on 20 randomly selected examples and test on the rest. The average prediction accuracies over 10 trials obtained by J4.8 DT and BN are presented in Table 8 together with the previously reported results. In each trial, the top-k most informative words are selected using the same information gain criterion as in [40]. However, because Netica-J API [26] has a hard-coded restriction on the network size, only the top-32 most informative words are used for BNs, whereas 96 word attributes are used for the others. As highlighted in [40], using fewer attributes can cause problems for the BN because important discriminating attributes may be excluded. Despite this, employing our proposed minimal set identification procedure, we are able

TABLE 8
Average Prediction Accuracy of Various Classification Algorithms on the Goats Data Set

Accuracy (%)				
N Neigh	ID3	Percept	BackP	PEBLS
62.0	64.7	66.3	67.0	62.7
Naive Bayes	Rocchio	J4.8	BN	
62.9	69.4	61.0	64.8	

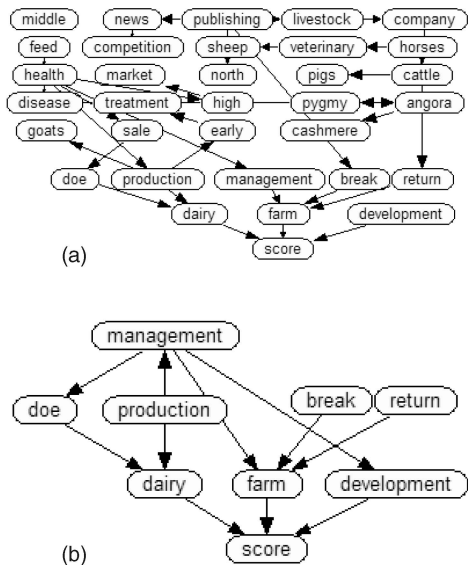


Fig. 10. BNs learned from the goats Web page data set for a real user. (a) With all 32 word attributes. (b) With the two-tier minimal set of attributes.

to learn BNs that outperform, on the average, many of the other algorithms including the J4.8 DT. This shows that the minimal set that our iterative approach has automatically identified is effective in predicting the real user’s rating, although for the small training sample size, BN learning using the search-based *CaMML* tool is not the best approach to predict with perfect data. However, we note that, like DT, the other classification methods do not encode variable dependencies and thus would not be as robust as our BN-based approach for handling missing and erroneous values.

Next, training on 60 randomly chosen examples, we discover the minimal set of important attributes for that user using our iterative learning procedure. Fig. 10a shows the network that is learned from the 32 most informative words in a trial. Our procedure has retained the “dairy,” “farm,” and “development” attributes for a second round of learning in which all these three attributes have remained connected to the *score*. As such, they constitute the minimal set of important attributes. Although markers are unavailable, our two-tiered context model can effectively capture the causal dependencies by including also those word attributes that are connected to the minimal set (“doe,” “production,” “management,” “break,” and “return”). The final learned network is given in Fig. 10b. The proposed approaches for handling missing and erroneous values that we have evaluated previously with the restaurant recommender data can now be applied to this problem.

Fig. 11 compares the prediction accuracies of BN and J4.8 DT for various rates of missing values in the test sets. In the experiments, only one attribute from the minimal context set is missing at a time. A missing rate of x percent means that the problematic attribute has a x percent probability of missing from an example. The results show that BN consistently outperforms the J4.8 DT in overcoming missing inputs to predict more accurately on the test sets. Fig. 12 compares the performance when various probabilities of errors are introduced. Specifically, in each of the 10 trials, a

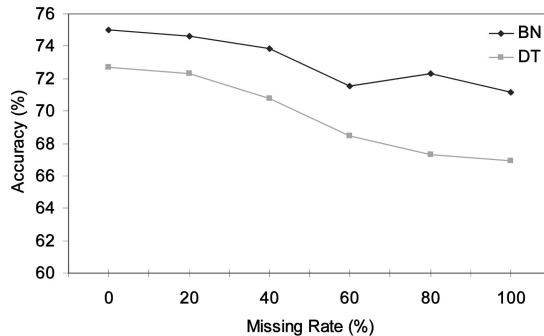


Fig. 11. Average prediction accuracy of the BN against the J4.8 DT when there are missing values.

randomly chosen attribute from the minimal set is injected with errors. We apply our automatic error-discovery and data-cleansing procedure of Algorithm 2 to this erroneous data until the error estimate for the next most erroneous word falls below 10 percent, so as to overlook trivial noises in the data set. The results show that BN consistently outperforms J4.8 DT when there are errors among inputs for important word attributes.

We note that this Web page data set is challenging because there are no domain markers to help our learning. Also, causal dependencies among word attributes in this data set are weak, evident from the fact that J4.8 DT has not performed very badly in spite of its inability to exploit dependencies. Other text data sets may contain stronger dependencies due to the redundancies among data fields, which our proposed techniques can readily exploit. For example, the integrity constraints [44] are a form of strong dependencies. In general, we expect the causal dependencies among context parameters in the mobile recommenders to be much stronger than those in text recommendation. We have shown that our approach reliably identifies the minimal set of important parameters for a particular real user and also effectively discovers and exploits the dependencies among these parameters to produce a recommender that is robust against both missing and erroneous inputs.

9 CONCLUSION AND FUTURE WORK

In mobile context-aware recommenders, operational constraints in context acquisition require us to minimize the

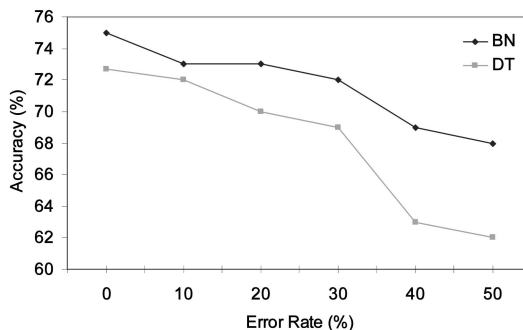


Fig. 12. Average prediction accuracy of the BN against the J4.8 DT when there are erroneous values.

context to be acquired, as well as to compensate for missing and erroneous context inputs effectively. In this paper, we have presented a BN-based approach to developing context-aware recommender systems that operate robustly under such challenges.

Specifically, we present a learning procedure that discovers the minimal set of important context parameters for a user. By iteratively trimming the learned network of those parameters without connection to the user rating, or *score* variable, we identify the minimal context for that user. In applications that reason on defined markers, we can apply our procedure on a typically much smaller set of markers. This allows us to identify the minimal context for a user from fewer examples when there are many context parameters to be considered. In addition, we present a two-tiered context model that effectively captures the causal dependencies among context parameters to overcome missing context inputs. To assist the learned BN in overcoming erroneous context inputs, we have also presented an automatic error-discovery and data-cleansing procedure.

Through an extensive series of experiments, we validate that our system can indeed accurately discover the causal dependencies among parameters to yield a clear graphical model of the problem. We are able to consistently identify the minimal set of important context parameters specific to each user. Also, results show that because the BN intrinsically encodes the causal dependencies among context parameters, the learned BN predicts accurately even when important context inputs are unavailable. Furthermore, our proposed error-handling procedure effectively harnesses the captured causal dependencies in the learned BN to overcome erroneous context inputs and achieve accurate predictions on unseen examples.

Note that BN learning is not without its limitations. The MML-based approach that is adopted in our work is one of the best available methods for learning the network structure from the data. However, it is expected that even this method might not find the correct causal model when the number of examples is too small or when the number of attributes is too large [35]. Furthermore, to the best of our knowledge, there is not yet any widely accepted mechanism to *adapt* the learned network's *structure* to new examples. This means that the BN in the recommender for a user has to be relearned offline when substantial new data becomes available over time. As the state of technologies for automatic BN learning from the data is advancing rapidly, these and other limiting issues should get resolved in the near future.

Going forward, there are many other possible directions for future work. In particular, an alternative approach for identifying the minimal set of important parameters of a target node is to consider all of the nodes that are captured in its Markov blanket, instead of just the set of nodes directly connected to it in the learned network. The motivation is that a BN node is in theory independent of all other nodes given just the nodes in its Markov blanket. However, we can only ascertain through further experiments whether analyzing this larger set of parameters would be beneficial in practice. In addition, although BN is superior in handling missing

and erroneous context, the *interactions* within a BN are not as interpretable as the logic that is represented by a DT. For instance, suppose J4.8 DT splits on attribute "*weather*," which can have a value of "*good*" or "*bad*." In this case, we may observe that for "*bad weather*," "*location*" yields the greatest entropy reduction but not so for "*good weather*." Such logic provides explanations for important questions like "*Is location important to that user when the weather is good/bad?*" It is our intention to explore, in addition to the other issues mentioned above, the extraction of similar explanations from learned BNs.

ACKNOWLEDGMENTS

Ghim-Eng Yap is sponsored by a graduate scholarship from the Agency for Science, Technology, and Research (A*Star). Hwee-Hwa Pang is partially supported by a grant from the Office of Research, Singapore Management University. The reported work is supported in part by the I²R-SCE Joint Lab on Intelligent Media. The authors would like to thank the associate editor and the anonymous reviewers for their invaluable comments.

REFERENCES

- [1] P. Resnick and H.R. Varian, "Recommender Systems," *Comm. ACM*, vol. 40, no. 3, pp. 56-58, 1997.
- [2] H.W. Tung and V.W. Soo, "A Personalized Restaurant Recommender Agent for Mobile E-service," *Proc. IEEE Int'l Conf. E-Technology, E-Commerce, and E-Service (EEE '04)*, pp. 259-262, 2004.
- [3] J. Ji, C. Liu, J. Yan, and N. Zhong, "Bayesian Networks Structure Learning and Its Application to Personalized Recommendation in a B2C Portal," *Proc. IEEE/WIC/ACM Int'l Conf. Web Intelligence (WI '04)*, pp. 179-184, Sept. 2004.
- [4] D. Goldberg, D. Nicholas, B.M. Oki, and D.B. Terry, "Using Collaborative Filtering to Weave an Information Tapestry," *Comm. ACM*, vol. 35, no. 12, pp. 61-70, 1992.
- [5] M. Balabanovic and Y. Shoham, "Fab: Content-Based, Collaborative Recommendation," *Comm. ACM*, vol. 40, no. 3, pp. 66-72, Mar. 1997.
- [6] R. Burke, "Hybrid Recommender Systems: Survey and Experiments," *User Modeling and User-Adapted Interaction*, vol. 12, pp. 331-370, 2002.
- [7] A.K. Dey and G.D. Abowd, "Towards a Better Understanding of Context and Context-Awareness," Technical Report GIT-GVU-99-22, panel paper at HUC 1999, June 1999.
- [8] O. Madani and D. DeCoste, "Contextual Recommender Problems," *Proc. Workshop Utility-Based Data Mining (UBDM '05)*, pp. 86-89, Aug. 2005.
- [9] G. Adomavicius, R. Sankaranarayanan, S. Sen, and A. Tuzhilin, "Incorporating Contextual Information in Recommender Systems Using a Multidimensional Approach," *ACM Trans. Information Systems*, vol. 23, no. 1, pp. 103-145, 2005.
- [10] M. van Setten, S. Pokraev, and J. Koolwaaij, "Context-Aware Recommendations in the Mobile Tourist Application COMPASS," *Proc. Third Int'l Conf. Adaptive Hypermedia and Adaptive Web-Based Systems (AH '04)*, pp. 235-244, 2004.
- [11] G.-E. Yap, A.-H. Tan, and H.-H. Pang, "Dynamically-Optimized Context in Recommender Systems," *Proc. Int'l Conf. Mobile Data Management (MDM '05)*, pp. 265-272, May 2005.
- [12] C.S. Wallace and K.B. Korb, "Learning Linear Causal Models by MML Sampling," *Causal Models and Intelligent Data Management*, A. Gammerman, ed., Springer, 1999.
- [13] D. Heckerman, "A Tutorial on Learning with Bayesian Networks," *Learning in Graphical Models*, M. Jordan, ed., MIT Press, 1999.
- [14] J.H. Friedman, "On Bias, Variance, 0/1-Loss and the Curse of Dimensionality," *Data Mining and Knowledge Discovery*, vol. 1, no. 1, pp. 55-77, Mar. 1997.

- [15] M. Singh and G. Provan, "Efficient Learning of Selective Bayesian Network Classifiers," Technical Report MS-CIS-95-36, Computer and Information Science Dept., Univ. of Pennsylvania, Nov. 1995.
- [16] T. Gu, H.K. Pung, and D.Q. Zhang, "A Bayesian Approach for Dealing with Uncertain Contexts," *Proc. Second Int'l Conf. Pervasive Computing (Pervasive '04)*, Apr. 2004.
- [17] K.B. Korb and A.E. Nicholson, *Bayesian Artificial Intelligence*. CRC Press, 2003.
- [18] A. Chen, "Context-Aware Collaborative Filtering System: Predicting the User's Preference in the Ubiquitous Computing Environment," *Proc. Int'l Workshop Location- and Context-Awareness (LoCA '05)*, T. Strang and C. Linnhoff-Popien, eds., pp. 244-253, 2005.
- [19] R.J. Mooney and L. Roy, "Content-Based Book Recommending Using Learning for Text Categorization," *Proc. Fifth ACM Conf. Digital Libraries*, pp. 195-240, June 2000.
- [20] G. Lacey and S. MacNamara, "Context-Aware Shared Control of a Robot Mobility Aid for the Elderly Blind," *Robotics Research*, vol. 19, no. 11, pp. 1054-1065, Nov. 2000.
- [21] J.S. Breese, D. Heckerman, and C. Kadie, "Empirical Analysis of Predictive Algorithms for Collaborative Filtering," *Proc. 14th Conf. Uncertainty in Artificial Intelligence (UAI '98)*, pp. 43-52, 1998.
- [22] J. Ji, Z. Sha, C. Liu, and N. Zhong, "Online Recommendation Based on Customer Shopping Model in E-Commerce," *Proc. IEEE/WIC/ACM Int'l Conf. Web Intelligence (WI '03)*, pp. 68-74, Oct. 2003.
- [23] J.W. Koolwaaij and P. Strating, "Service Frameworks for Mobile Context-Aware Applications," *Proc. eChallenges 2003 Workshop*, Oct. 2003.
- [24] D. Zhang, X.H. Wang, and K. Hackbarth, "OSGi Based Service Infrastructure for Context Aware Automotive Telematics," *Proc. IEEE Vehicular Technology Conf. (VTC '04)*, pp. 2957-2961, May 2004.
- [25] T. Olsson, "Bootstrapping and Decentralizing Recommender Systems," dissertation, Computer Science Division, Dept. of Information Technology, Uppsala Univ., June 2003.
- [26] Norsys Software Corp., "Netica-J: Java Netica API," <http://www.norsys.com/netica-j.html>, retrieved on June 2005.
- [27] R.E. Neapolitan, *Probabilistic Reasoning in Expert Systems: Theory and Algorithms*. John Wiley & Sons, 1990.
- [28] D.B. Rubin, "Multiple Imputation after 18+ Years (with Discussion)," *J. Am. Statistical Assoc.*, vol. 91, pp. 473-520, 1996.
- [29] R.J.A. Little and D.B. Rubin, *Statistical Analysis with Missing Data*. John Wiley & Sons, 2002.
- [30] V.J. Hodge and J. Austin, "A Survey of Outlier Detection Methodologies," *Artificial Intelligence Rev.*, vol. 22, pp. 85-126, 2004.
- [31] H. Xiong, G. Pandey, M. Steinbach, and V. Kumar, "Enhancing Data Analysis with Noise Removal," *IEEE Trans. Knowledge and Data Eng.*, vol. 18, no. 3, pp. 304-319, Mar. 2006.
- [32] C.E. Brodley and M.A. Friedl, "Identifying Mislabeled Training Data," *J. Artificial Intelligence Research*, vol. 11, pp. 131-167, 1999.
- [33] S.C. Kachigan, *Statistical Analysis*. Radius Press, 1986.
- [34] I.H. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques*, second ed. Morgan Kaufmann, 2005.
- [35] L.R. Hope and K.B. Korb, "A Bayesian Metric for Evaluating Machine Learning Algorithms," *Proc. Australian Joint Conf. Artificial Intelligence (AI '04)*, G.I. Webb and X. Yu, eds., pp. 991-997, Springer, 2004.
- [36] X. Zhu, "Semi-Supervised Learning Literature Survey," Technical Report 1530, Dept. of Computer Sciences, Univ. of Wisconsin-Madison, 2005, http://www.cs.wisc.edu/~jerryzhu/pub/ssl_survey.pdf.
- [37] D. Cohn, L. Atlas, and R. Ladner, "Improving Generalization with Active Learning," *Machine Learning*, vol. 15, no. 2, pp. 201-221, 1994.
- [38] M.J. Pazzani, J. Muramatsu, and D. Billsus, "Syskill and Webert: Identifying Interesting Web Sites," *Proc. 13th Nat'l Conf. Artificial Intelligence (AAAI '96) and Eighth Innovative Applications of Artificial Intelligence Conf. (IAAI '96)*, pp. 54-61, 1996.
- [39] G. Salton, *The SMART Retrieval System—Experiments in Automatic Document Processing*. Prentice Hall, 1971.
- [40] M.J. Pazzani and D. Billsus, "Learning and Revising User Profiles: The Identification of Interesting Web Sites," *Machine Learning*, vol. 27, pp. 313-331, 1997.
- [41] R. Duda and P. Hart, *Pattern Classification and Scene Analysis*. John Wiley & Sons, 1973.
- [42] M. Minsky and S. Papert, *Perceptrons*. MIT Press, 1969.

- [43] J.R. Quinlan, "Induction of Decision Trees," *Machine Learning*, vol. 1, pp. 81-106, 1986.
- [44] P. Godfrey, J. Grant, J. Gryz, and J. Minker, "Integrity Constraints: Semantics and Applications," *Proc. Logics for Databases and Information Systems*, pp. 265-306, 1998.



Ghim-Eng Yap received the bachelor's degree in computer engineering—with first class honors—from Nanyang Technological University in 2004 and is currently pursuing the PhD degree at the Nanyang Technological University, Singapore, under a full-time graduate scholarship awarded by the Agency for Science, Technology, and Research (A*Star). His current research interests include context awareness, recommender systems, reasoning under uncertainty, causal interpretation, graphical models, computational intelligence, and biological knowledge discovery.



Ah-Hwee Tan received the BSc (first class honors) and MSc degrees in computer science from the National University of Singapore and the PhD degree in cognitive and neural systems from Boston University. He is an associate professor and the director of the Emerging Research Laboratory, School of Computer Engineering, Nanyang Technological University. He is also a faculty associate of A*Star Institute for Infocomm Research, where he was formally the manager of the Text Mining and Intelligent Cyber Agents Groups. His current research interests include cognitive and neural systems, intelligent agents, machine learning, media fusion, and information mining. He holds several patents and has successfully commercialized a suite of document analysis and text mining technologies. He is a senior member of the IEEE and an editorial board member of *Applied Intelligence*.



Hwee-Hwa Pang received the BSc (first class honors) and MS degrees from the National University of Singapore in 1989 and 1991, respectively, and the PhD degree from the University of Wisconsin at Madison in 1994, all in computer science. He is an associate professor at the Singapore Management University. He holds a joint appointment as a principal scientist at the A*Star Institute for Infocomm Research. His current research interests include database management systems, data security, and information retrieval. He has many years of hands-on experience in system implementation and project management. He has also participated in transferring some of his research results to industry.