Leveraging Graphical Models to Improve Accuracy and Reduce Privacy Risks of Mobile Sensing

Abhinav Parate Meng-Chieh Chiu

Deepak Ganesan Benjamin M. Marlin

Department of Computer Science
University of Massachusetts, Amherst
Amherst, MA 01003-9264
{aparate,joechiu,dganesan,marlin}@cs.umass.edu

ABSTRACT

The proliferation of sensors on mobile phones and wearables has led to a plethora of context classifiers designed to sense the individual's context. We argue that a key missing piece in mobile inference is a layer that fuses the outputs of several classifiers to learn deeper insights into an individual's habitual patterns and associated correlations between contexts, thereby enabling new systems optimizations and opportunities. In this paper, we design *CQue*, a dynamic bayesian network that operates over classifiers for individual contexts, observes relations across these outputs across time, and identifies opportunities for improving energy-efficiency and accuracy by taking advantage of relations. In addition, such a layer provides insights into privacy leakage that might occur when seemingly innocuous user context revealed to different applications on a phone may be combined to reveal more information than originally intended. In terms of system architecture, our key contribution is a clean separation between the detection layer and the fusion layer, enabling classifiers to solely focus on detecting the context, and leverage temporal smoothing and fusion mechanisms to further boost performance by just connecting to our higher-level inference engine. To applications and users, *CQue* provides a query interface, allowing a) applications to obtain more accurate context results while remaining agnostic of what classifiers/sensors are used and when, and b) users to specify what contexts they wish to keep private, and only allow information that has low leakage with the private context to be revealed. We implemented *CQue* in Android, and our results show that *CQue* can i) improve activity classification accuracy up to 42%, ii) reduce energy consumption in classifying social, location and activity contexts with high accuracy(>90%) by reducing the number of required classifiers by at least 33%, and iii) effectively detect and suppress contexts that reveal private information.

MobiSys'13, June 25-28, 2013, Taipei, Taiwan

Keywords

Continuous context-sensing; Energy-Accuracy-Privacy optimizations; Mobile computing

Categories and Subject Descriptors

C.5.3 [**Computer System Implementation**]: Microcomputers—*Portable devices*; D.4.8 [**Operating Systems**]: [Performance Modeling and prediction]; K.8 [**Personal Computing**]: [General]

1. INTRODUCTION

The past decade has seen unprecedented growth in sensor-rich mobile phones and wearable accessories such as fitness monitors, sleep monitors, heart monitors and others. With the proliferation of such devices, there has been significant emphasis on techniques to draw higher-level inferences from continuous sensor data as we move around in our day-to-day lives. Research has shown that several aspects of our behavior can be inferred including physical activity, sleep behavior, social context, movement patterns, emotional or affective context, and mental disorders, with varying degrees of accuracy.

The growing landscape of high-level inferences presents an interesting opportunity: *can we combine these inferences to obtain deeper insights into individual behavior?* Intuition suggests that the outputs of individual inference algorithms must be correlated across space and time; after all, they sense various dimensions of an individual's habits, behaviors and physiology all of which are inter-linked. As a simple example, take the case of an individual's mobility patterns and how much of it can be inferred without using any location sensor such as GPS, cell tower or WiFi. Location is often correlated to social context — the fact that phones of colleagues are in proximity (detected via bluetooth) can indicate that the likely location is the workplace whereas the fact that a family member's phone is nearby means that one is likely at home. Similarly, location relates to activity context — a user may be more sedentary at work than at other times during the day, therefore one could infer that the most likely location is the workplace by observing the level of activity. More broadly, almost all inferences are related in one way or another — studies have shown that sleep (or lack of it) affects our mood and productivity, social interactions influence our emotional context, addictive behavior is correlated to specific locations and social interactions, and so on

The existence of a rich interaction graph between inferences presents an opportunity and a challenge. On one hand, a model of the re-

lations across the different inferences can be leveraged to gain insights into individual behavior, thereby enabling new systems optimization opportunities. For example, a high level inference framework might observe relations between semantic location and activity levels, and learn that an individual has mostly sedentary behavior at the workplace. This provides two opportunities to optimize inference of the location context, "work": a) energy-efficiency can be improved by relying primarily on activity inference rather than expensive GPS, and b) accuracy can be improved when there is significant error (e.g. indoor settings) by fusing it with activity levels.

On the other hand, the ability to leverage relations across inferences comes with a steep price tag — loss of privacy. A mobile application that purports to merely be using the accelerometer to detect activity levels may indeed be inferring your location. More disturbing is the possibility that a single application developer may have several applications, one that monitors sensors for activity level, and perhaps another that uses bluetooth, which may be leveraged in conjunction to reveal much more than from the seemingly innocuous individual applications. Compounding this issue is the fact that we do not have tools that allow us to reason about how much of privacy exposure occurs from revealing seemingly innocuous sensor data if the adversary were to have a good model of the relations across contexts.

In this paper, we present a mobile "big data" inference framework, *CQue*, that takes as input streams of inferences from diverse on-body and smartphone-based sensors, and provides a unified way for exploiting and exploring relations across these inferences. *CQue* can be used in several ways: a) an inference algorithm can leverage *CQue* to improve accuracy and/or energy-efficiency, while remaining agnostic of how this is achieved, b) a user can issue "what if" queries that explore the extent of privacy leakage of sensitive information that might be possible if certain sensor data were revealed to one or more applications, and c) a user can specify privacy policies to provide a first order protection against privacy leakage of sensitive information to the untrusted applications.

In terms of system architecture, the key benefit of *CQue* is that it separates *detection* from *fusion*. Existing classifiers are largely designed in a stovepipe manner to address a specific context sensing goal as best as possible. While they start with detection of the specific activity (e.g. conversation, walking, etc), real-world vagaries often result in spurious state transitions due to several confounding factors. To address these issues, classifiers often rely on other sensor sources that can identify confounders and reduce errors. *CQue* provides a clean separation between the detection layer and the fusion layer — in this way, a classifier can be designed to just focus on detecting the phenomena of interest, and leverage fusion mechanisms to further boost performance by just hooking into *CQue*.

To users and applications, *CQue* offers a simple context querying interface with support for several types of queries. A "context query" can request different contexts, while optionally specifying constraints such as confidence requirements and delay bounds. For example, a query might request sedentary activity context with 90% certainty and a maximum notification delay of two minutes. *CQue* uses the query constraints to reason about how to duty-cycle inference algorithms, and how much temporal history to leverage to improve accuracy and confidence. A "what if" query provides a measure of potential leakage of sensitive information if applications were allowed access to specific sensor sources. For example, a query can request the expected leakage of "home" location context if an application had access to accelerometer and bluetooth data. Internally, *CQue* would execute such a query by providing a measure of the correlation between the outputs of inference algorithms that operate on accelerometer and bluetooth data, and the specific location context that the user does not wish to reveal. Finally, through privacy policies, a user can specify a context as private that should not be revealed, and suppress the non-private contexts in real-time that can be used to infer private context using correlations among these contexts. A user can specify this policy specific to an application or a group of applications that can potentially collude.

Our results show that:

- *CQue* can answer context queries with high confidence and improve accuracy up to 42% by performing fusion of information from multiple context-inference algorithms
- When energy is limited, *CQue* can lower execution costs for multiple context queries by exploiting context relations to run fewer inference algorithms. If energy is plentiful, *CQue* can decide what context algorithms in addition to the query set should be executed to improve accuracy.
- *CQue* is effective in assessing privacy risks and provides privacy while ensuring high utility to the applications.

The rest of the paper is organized as follows. §2 describes the prior work done in the related research areas. In §3, we provide an high-level overview of the *CQue* query engine along with the description of how a context query can be specified in our framework. In §4.1, we describe the relationship model used by *CQue*. Rest of §4 describes the components of *CQue* query engine in detail that are responsible for optimizing multiple context queries and addressing privacy. We describe the implementation details and the experimental evaluation of *CQue* in §5 and §6 respectively. We discuss possible extensions of this work in §7. Finally, the conclusions are provided in §8.

2. RELATED WORK

In this section, we describe three areas of related work — model-driven sensor data acquisition, context inference for mobile phones and privacy in temporal data.

Graphical models for sensor data acquisition: There has been substantial work on leveraging spatial and temporal models of correlations between distributed sensor sources to optimize sampling and communication in a sensor network [3, 5, 6, 12, 14]. For example, BBQ [6] uses a Dynamic Bayesian Network to select the minimum number of sensor nodes for data acquisition such that it can answer range queries within query-desired confidence bounds. Meliou et al [14] extend this work to sensor network routing where a query can be answered with desired confidence bounds while traversing the minimum number of nodes in a network spanning tree. Graphical models are also used in [5], which explored the problem of answering range queries while minimizing the energy cost of sampling sensors, and in [3], where sensors use models to reduce the communication costs by transmitting samples to a base station only when the ground truth is significantly different from the prediction made by the model. The similarities with prior efforts are only in that they leverage DBNs — we use DBNs in a novel application context which is for real-time inference on a mobile phone to enable energy-accuracy-delay tradeoffs and protect privacy.

Context-Sensing: Our work is closest to ACE [15] that proposes a context sensing engine that exploits relationships among contexts to infer an energy-expensive context from a cheaper context. The central difference between ACE and *CQue* is that the former uses Associate Rule Mining to learn rules among contexts (e.g. *Driving* $\Rightarrow \neg$*At Home*), but such mining approaches cannot take a context

classifier's uncertainty into account. The ability to take uncertainty into account is critical when using classifiers that operate on raw sensor data — for example, if a classifier may detect *Driving* with a low confidence, say 0.7, GPS may detect *At Home* with an error radius of 100m, and so on. In these cases, the rules would assume perfect context information whereas *CQue* would use the uncertainty in inferring the relations. Thus, a probabilistic approach is strictly superior to a rule based one.

Also related to *CQue* are efforts to develop a context-sensing engine for phone that can be used by applications to request contexts [4, 13]. The Jigsaw context sensing engine [13] comprises of a set of sensing pipelines for accelerometer, microphone and GPS. More recently, Kobe [4] proposed a context querying engine for mobile phones that can be used to plugin different classifiers, and that balances energy consumption, latency and accuracy of the classifiers by offloading computation to the cloud. Unlike these approaches, *CQue* can leverage probabilistic relations across contexts.

CQue is complementary to prior work on context sensing engines which have largely explored optimizations of individual classifiers (e.g. [2, 11, 13, 20]). *CQue* allows individual classifiers to be easily integrated without worrying about how to leverage other contexts to improve performance and efficiency.

Privacy in temporal data: There has been some work in addressing privacy where the adversary is aware of the temporal correlations [8, 17, 19]. MaskIT [8] presents a privacy-preserving mechanism to filter a stream of contexts that can be used to answer context queries requested by the phone applications. It presents two privacy checks to decide when a user context can be released while providing privacy guarantees against an adversary knowing temporal correlations modeled as a Hidden Markov model(HMM) of user contexts.

Our goal in this work is not to to prove privacy guarantees or design new privacy metrics, rather our argument is that the use of DBN enables considerably more flexible privacy policies in comparison with an HMM-based approach. When MaskIT identifies that a privacy breach might occur, it suppresses all contexts at that time without considering which subset of them might contribute to the breach. In contrast, *CQue* enables more fine-grained reasoning of which specific context(s) lead to a privacy breach such that only that information can be suppressed.

3. CQUE OVERVIEW

The *CQue* framework has been designed with the goal of providing an easy-to-use abstraction to application developers and end users, both for exploiting correlations across inference to improve energy-efficiency and accuracy, as well as to use these correlations to ascertain privacy leakage or contextual insights. This goal is accomplished using a high level query language and an underlying execution environment running as a service on the phone. In the following, we provide an high-level overview of the *CQue* framework.

Context Query Interface. At the top level of *CQue* are applications that issue context queries using a high-level declarative query language. *CQue* provides a simple query language for the applications to request contexts. Consider the following three queries:

```
Q1. (drive | accel+gyro+gps, 240s, 0.8)
Q2. (home? | accel+gyro)
Q3. (home? | with friends)
```

The first query requests the context *drive*, and includes three other optional fields. The keywords *accel*, *gyro* and *gps* suggest that

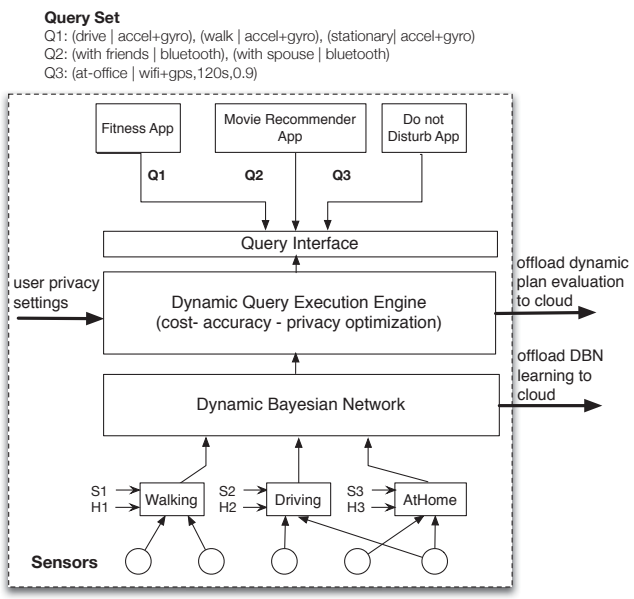

Figure 1: High-level overview of the framework describing the workflow

any combination of classifiers that use those sensors may be used in answering the query (to ensure that the app only uses sensors for which it has approval from the user). These keywords could also refer to other inferences, if the application needs more explicit control over other contexts that should be fused. Finally, the query can also replace this with a '*' which would let *CQue* automatically choose sensors and classifiers to optimize the system. The value '240s' in the query is the *delay tolerance* and denotes that the application can tolerate the maximum delay of 240 seconds for each record in the query response. The query processor uses this delay window to improve the confidence in the output context value. The query can also provide a *confidence threshold*. In the above example, the query processor will output context as soon as it reaches confidence of '0.8', provided it is within the delay tolerance period. If confidence values associated with the context output are lower than the query-desired *confidence threshold*, the context is reported as 'unknown'.

The next two queries marked by '?' are privacy queries and provide a measure of *information gain* in queried context "home" if any inference algorithm operating over accelerometer and gyroscope sensors were used (Q2) or if *with friends* inference algorithm was used (Q3). In turn, this measure provides a *privacy score*, an indicator of privacy leakage. Note that in providing a *privacy score*, *CQue* is limited to using inference algorithms that are available in its library.

CQue Architecture. The central tenet of *CQue* is an architectural separation between the mechanisms that inference algorithms use to improve performance. Inference algorithms are often designed in a stovepipe manner and integrate a combination of the following components: a) *feature extraction*, which involves extracting time-domain or frequency-domain features from the raw sensor data, b) *classification*, where they look at a short temporal sequence and detect a particular event or state, for example, walking, running, cycling, smoking, etc, c) *temporal consistency* mechanisms such as Hidden Markov Models (HMMs) to correct

mis-classifications that occur in the classification, and d) *context fusion* where they leverage other information such as location or time to provide additional input to correct classification errors.

In contrast to the stovepipe approach, our design separates these components into a cleaner, layered architecture where any classifier can simply hook into *CQue*, and leverage temporal consistency and context fusion mechanisms to improve performance. *CQue* models temporal consistency with a time-series of previous observations, and uses active learning mechanisms to automatically determine when user input is needed to improve this model. *CQue* also automatically determines the relationships between a classifier output and all other observations from other classifiers/sensors that it has access to, and determines when and how to use additional observations to improve accuracy. In short, it can make the process of designing a new inference algorithm a lot easier for a designer.

Execution Environment. The execution environment of *CQue* is shown in Figure 1 and consists of following core components: (i) a graphical model, namely, Dynamic Bayesian Network, and (ii) a query processor. The *Dynamic Bayesian Network* (DBN) is the at the core of *CQue* and plays an important role in making various decisions during query processing. The DBN provides a model of the relationships across multiple contexts for the phone user and keeps track of the time-series of observations for various contexts. It then uses context relationships to boost confidence in observations made by individual classifiers or to correct them. The second component is a *query processor* which is responsible for interaction with applications including receiving context queries and generating the responses to the queries. In addition, depending on the current state of the DBN and the time-series of observations, the query processor needs to determine which context-inference algorithms should be executed such that it can provide best answers for the queries within the specified constraints. These decisions are made by the query processor at every time step during execution. As shown in Figure 1, the result of these decisions are the signals $S_1, S_2, ..., S_n$ indicating which context-algorithms be executed. Finally, upon observing the output of selected context-inference algorithms, *query processor* enforces user-specified privacy policies (defined in §4.2) by deciding a maximal set of queried contexts whose values can be released to the applications without violating the privacy policies.

In addition to the query execution, the query processor is responsible for sampling human-provided context values during the learning phase of the DBN. Such human input can be minimized through active learning to only obtain input at appropriate times to improve the structure of the DBN. These human inputs shown as signals $H_1, H_2, ..., H_n$ in Figure 1 are provided to the context-inference algorithms that can use it to retrain themselves to personalize for the phone user.

4. THE EXECUTION ENVIRONMENT

The execution environment of *CQue* is responsible for i) modeling and learning the relationships across contexts, ii) multi-query optimization such that uncertainty in queried contexts is minimized while operating within the budget and delay constraints of a query, and iii) executing privacy queries and enforcing privacy policies. In the following, we describe each component of the execution environment in detail.

4.1 Inference Framework

The core learning framework in *CQue* is a Dynamic Bayesian network (DBN), which is a class of Bayesian networks that can represent a time-series of random variables to model temporal consistency. A DBN can describe both temporal and static relationships among random variables at different time instances. This model is represented as a directed acyclic graph consisting of nodes corresponding to each random variable at each time instance. The static dependencies across random variables at time instance t are represented by edges connecting nodes corresponding to these random variables. The temporal dependencies are represented by transition edges connecting nodes at time instance $t-1$ with nodes at time instance t. Each node in this graph has a conditional probability table (CPT) describing dependencies on its parent nodes. Figure 2 shows an example of 2-step DBN for two time slices which can be unrolled to accommodate a time-series of any length by duplicating the time-slices and transition edges. We use notation $\mathbf{X}$ to denote the set of random variables. We use $\mathbf{X_t}$ to denote set of nodes representing random variables $\mathbf{X}$ at time instance t and use $\mathbf{X_{1...t}}$ as short notation to describe time series of nodes $\mathbf{X_1}, ..., \mathbf{X_t}$. With this notation, the joint probability distribution for a time series of nodes is given by

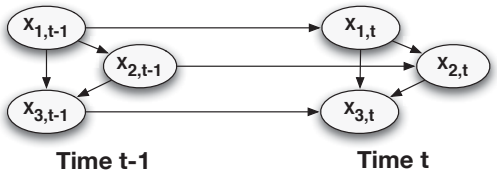

Figure 2: An example of 2-step Dynamic Bayesian Network. The nodes in the graph model the set of random variables $\mathbf{X} = \{X_1, X_2, X_3\}$.

$$Pr(\mathbf{X_{1...t}}) = Pr(\mathbf{X_1}) \prod_{t=2}^{t} Pr(\mathbf{X_t}|\mathbf{X_{t-1}})$$

In the above equation, probability $Pr(\mathbf{X_t}|\mathbf{X_{t-1}})$ is computed using the DBN.

DBN Model for CQue. In *CQue*, we have three types of DBN nodes — sensors, classifier-provided context, and real-world context. The goal of the DBN is to model relationships across real-world contexts while taking into account the uncertainty associated with the classifier-provided contexts. We considered two criteria in constructing the DBN model: a) classifiers are black boxes, and we do not assume knowledge of what sensors they utilize, what sampling rates they use, how they duty-cycle sensors to save energy, and what features they extract, and b) we wish to keep the model simple and low complexity so that DBN inference can be performed in real-time on the mobile phone.

To address these considerations, classifiers and sensors are separated from the DBN into a different layer. Figure 3 shows an example where classifiers provide the probability of a context through observation nodes called virtual evidences, $\mathbf{O}$, to the DBN. The uncertainty in evidence is accounted in these observation nodes using Pearl's method [18]. In this method, for some context X_i and corresponding observation node O_i, if $L(x)$ gives the likelihood of classifier stating that $X_i = x$ if X_i is actually in state x and if $\overline{L(x)}$ gives the likelihood of a classifier stating that $X_i = x$ if X_i is not in state x, then the conditional probabilities for observation node O_i given the current state for real world context X_i satisfies:

$$Pr(O_i = x|X_i = x) : Pr(O_i = x|X_i \neq x) = L(x) : \overline{L(x)}$$

With these conditional probabilities and given a series of virtual evidences $\mathbf{O_{1...t}}$, we can compute the joint probabilities for time series of real world contexts $\mathbf{X_{1...t}}$ as follows:

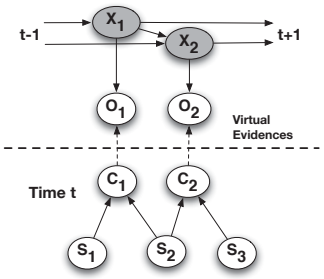

Figure 3: A model separating DBN from the layer of sensors and classifiers, where classifiers provide virtual evidences to the DBN.

$$Pr(\mathbf{X_{1\ldots t}}, \mathbf{O_{1\ldots t}}) = Pr(\mathbf{X_1}|\mathbf{O_1}) \prod_{t=2}^{t} Pr(\mathbf{X_t}|\mathbf{X_{t-1}}) Pr(\mathbf{O_t}|\mathbf{X_t})$$

This model has several benefits: a) it is computationally cheap both when classifiers are turned on or off since it avoids the complexity of modeling sensor and classifier variables in the DBN itself, b) it requires no additional human labels for modeling these variables while training the DBN, and c) it can easily be extended to handle multiple classifier implementations for the same context without changing the DBN structure.

DBN usage in CQue. The output of the DBN is used in several ways by *CQue*.

1. **Filtering/Smoothing** Most of the contexts that we observe in the real world have some temporal consistency i.e they last for some time period. Such temporal relationships can be used to correct intermittent misclassifications made by context-inference algorithms. Since a DBN models temporal consistency with the time-series of previous observations, a DBN can correct the current output and reduce the probability of a context taking an incorrect value. Apart from the temporal consistency, a misclassification can be corrected using relationships with other contexts if the observations are available for them.

2. **HindSight** The previous case described using history to correct output at the current time. Similarly, a DBN can be used to improve the confidences in historical observations, which perhaps had low confidence. Future observations with high confidence can be used to improve the confidence of previous observations. This capability can be exploited when queries specify a higher delay tolerance threshold.

3. **Value of Information** The DBN can also be used to assess the value of a context-inference algorithm. *Value of information* (VOI) is defined as the expected gain in certainty for the random variables in DBN if an additional observation is made. This is useful for the query processor, which can decide what context observation can provide highest utility.

4.2 Query Processor

The query processor is responsible for i) multi-query optimization i.e. achieving the confidence requirements of context queries while operating within query constraints, ii) execution of privacy leakage queries, and iii) enforcement of user-defined privacy policies. Multi-query optimization in *CQue* raises following challenges: a) how to choose the inference algorithms to execute given the budget?, and b) how to handle different delay tolerance requirements of queries? In addition, privacy queries and policies require us to answer: a) how to execute privacy queries?, and how to enforce privacy policies? We address these question in the rest of this section.

Which inferences to execute given a budget?

The *CQue* query processor needs to execute the set of inference algorithms that provides maximum value of information (VOI) for the queried contexts within the budget constraints. The budget is typically in the form of energy since inference algorithms consume energy, but in cases where the user is interrupted to provide labels, the budget may be the number of interruptions allowed per day.

Due to the relationships across contexts, the set providing maximum VOI may be larger than the query set as some contexts that are not part of the query set may be useful to improve the inference accuracy for contexts that are in the query set. Similarly, this set can be smaller than the query set if some of the queried contexts are strongly implied by other contexts in the query. In addition, the set of inferences that provides the maximum VOI can change over time, depending on the user's current context as well as dynamics in the user's behavior patterns. In this section, we describe an adaptive approach to deciding the set of inferences that need to be executed to satisfy query demands while remaining within the budget.

The problem of selecting the optimal set of context-algorithms to execute given a budget and the set of queries can be formalized as follows. Let Q be the set of context queries, let E_{t-1} be the set of context classifiers undergoing execution currently and v be the corresponding classifier outputs. Let $\mathcal{F}(Q, E, E_{t-1}, v)$ be the utility function giving the VOI of the contexts, E, for the query set where E is the subset of observation nodes $\mathbf{O}$ in the DBN. Let $C(E)$ give the cost of executing algorithms for contexts in E. If $\mathcal{B}$ is the maximum budget, then our goal can be defined as identifying a set E_t i.e. the set of context classifiers to be executed next such that:

$$E_t = \underset{E \subseteq \mathbf{O_t}: C(E) \leq \mathcal{B}}{\arg\max} \mathcal{F}(Q, E, E_{t-1}, v) \quad (1)$$

Since our goal is to have high certainty for query set Q, we consider the information gain for query set Q (also called VOI) as the utility function which is given by:

$$\mathcal{F}(Q, E, E_{t-1}, v) = H(Q|E_{t-1} = v) - H(Q|E, E_{t-1} = v) \quad (2)$$

where $H(.)$ is the entropy function. We must note that computing optimal-set of classifiers to be executed at the next iteration is an **NP**-hard problem[10]. We solve this optimization problem using a greedy approach as described in Algorithm 1. The results by Nemhauser et al. [16] and Krause et al. [10] have shown that a greedy algorithm provides a solution within the constant factor $(1 - \frac{1}{e})$ of the optimal solution for a general bayesian network.

Our extension of optimization problem to the DBN does not change the results as long as we select nodes E_t from a set S such that the nodes corresponding to S in DBN are independent to each other given the query set Q. In a DBN, we select E_t from a set of observation nodes O_t where each observation node is connected to the DBN only through a parent hidden node. Hence, any path between any two observation nodes is always blocked by one of the parent hidden nodes. Thus, any two observation nodes in DBN are **d**-separated [21] and hence, independent of each other.

We note that Algorithm 1 gives near-optimal solution when each classifier is assigned a unit cost. In a specific scenario, there may be several considerations in determining the cost, which depends

on the cost of sensing, processing, or obtaining human input. If the resulting costs are non-uniform in nature, we refer the readers to an algorithm as described in [10].

Algorithm 1 Compute Optimal Set

Input: Budget k; query set Q; set $\mathbf{O_t}$ from DBN; utility function $\mathcal{F}$; previous observed set E_{t-1} and corresponding set of classifier outputs v; Cost function C.
Output: Set of observations $E \subseteq \mathbf{O_t}$
Let $E = \phi$
for $i = 1$ to k **do**
  $o^* = \arg\max_{o \in \mathbf{O_t} \setminus E} \mathcal{F}(Q, E \cup \{o\}, E_{t-1}, v)$
  $E = E \cup \{o^*\}$
end for
return E

How to handle delay requirements of queries?

In *CQue*, we unroll the DBN such that it can accommodate a series of observations of length W. At each time-step, we slide the DBN window over observations such that the oldest observations are dropped and the latest observations are added to the DBN. As a result of this sliding window, any context observation resides in DBN for exactly W time-steps. Hence, we can compute the probability for queried contexts W times, corresponding to having $0, 1, 2, ..., W-1$ observations from the future. This allows our framework to answer a query with a delay upto $W-1$ time-steps. Our framework can use this delay period in three cases: i) if the sequence of historical observations have low confidence resulting in low confidence for the latest observation, ii) if the sequence of historical observations have fluctuating values for the contexts indicating low confidence in the output of context inference algorithms, and iii) if the latest observation is different from the historical observations. In the first two cases, we wait for the future observations and hope that these observations are consistent and have high confidence. If this happens, we boost the probability of queried context in hindsight. In the third case where the latest observation is different from the historical observations, it can happen either because of intermittent misclassification by the classifier or because the context value has actually changed. Any future observation can help in distinguishing these cases and improve the certainty in the context.

How to execute privacy queries?

Unlike queries that request current state of an individual, *privacy queries* look for aggregate information regarding the mutual information shared between a context and sensor data or other contexts available in *CQue*. In our framework, we provide two types of privacy queries: i) user specifies a query context, Q and wants to know information revealed about the queried context from a specified list of sensors (S) and contexts (C); and ii) user specifies a query context Q and wants to know ranked list of contexts along with a *privacy score* that indicates how much information is revealed by a ranked context about the queried context. Thus, these queries help user understand how much information can be revealed about certain context from other contexts. Together, these two query types enable users to make a decision regarding their own privacy.

For the first query type, given a user specified sensor list S we identify context classifiers C^S that can execute given sensors S. In addition to S, user can provide a set of contexts C. Now, we want to have an uncertainty measure in queried context Q if contexts in $C' = C \cup C^S$ are observed. We use a normalized variant of mutual information based on information theory that gives a distance metric between Q and C' and is given as follows :

$$D(Q|C') = 1 - \frac{H(Q) - H(Q|C')}{H(Q)}$$

This metric takes value 1 if Q is independent of C' and takes value 0 if Q is fully determined by C'. We use this measure as a *privacy score* where a higher score indicates less information leakage.

For the second query type, a user is interested in identifying ordered list of contexts that reveal most information about queried context Q. For each context c supported in *CQue*, we use *privacy score* $D(Q|c)$ as described above to measure information leaked about queried context Q. We rank contexts based on this metric.

How to enforce privacy policies?

As described in §2, one of the benefits of *CQue* is that the DBN provides in-depth and real-time information about correlations between different context outputs in contrast with HMMs and other techniques that have been leveraged in prior work. *Our focus is not on identifying the best privacy exposure policy or proving its privacy properties; rather it is to demonstrate that CQue can be used to develop such methods.*

To demonstrate these benefits, *CQue* supports a privacy policy that can suppress context values that leads to change in confidence or the probabilistic belief about a private context greater than threshold δ. Such a policy can provide greater privacy control than a simpler policy that just blocks a specific private context from being revealed. In *CQue*, a user can define these policies specific to a potential adversarial entity where this entity can be an application or a group of applications or all the applications on the user's phone. A policy for a group of applications can be useful to protect against information collusion among these applications in a group.

In *CQue*, we assume a strong adversary having access to the user's DBN. Now, we enforce user policy by using the DBN to calculate the confidence or the posterior probability for the private context using the output for the set of contexts, requested by the adversarial entity, as observations in DBN. Also, we compute prior probability for the private context without using any observations in the DBN. If the difference between posterior probability and the prior probability is greater than δ, then we suppress the value of the context that causes the maximum change in probabilities. If the change in probabilities is still high with the remaining context outputs, we repeat the process of removing context value that causes maximum change until we reach the threshold limit δ. We release the remaining contexts to the adversarial entity.

4.3 Learning the Graphical Model

Now, we describe the challenges involved in learning the DBN model for the phone user. There are two challenges in learning the DBN: a) how to learn structure without executing all sensors and context inference algorithms continuously on the phone since this consumes significant energy, and b) how to minimize interruptions of the phone user to obtain ground truth labels for learning.

Context relationship hints: The learning process can degrade user experience due to the energy cost of running several context inference algorithms concurrently on the phone. While it is possible to randomly sample a few contexts at a time, this can slow down the learning process as the random approach may not always sample related contexts together. To perform efficient sampling and facilitate faster learning of personalized relationships, we maintain, for each context, a list of contexts to be sampled together that may possibly have soft relationships. The soft relationships may or may not hold for individual users and it does not necessarily result in edges in DBN connecting softly-related contexts. As it may not be

possible to envision all the soft relationships, we still perform random sampling and bias the sampling for contexts that seem to be related.

While the use of soft relationships can address the efficient sampling problem, we can speed up the learning process and parameter estimation in DBN by the use of *hints* that can be provided by the context-inference algorithm developers. A context developer can include hints for each context that identify positive causes and negative causes for it. As an example, *driving* is a direct positive cause for user's location being *on street* whereas user's location being *at home* is a negative cause for *on street*. These hints can be directly utilized in setting the parameters for relationships that are true for a majority of users.

Minimizing human-provided labels: One of the challenges in learning the dynamic bayesian network is that we do not have access to ground truth for contexts. While ground truth can be obtained by having the user provide ground truth for an initial training period, we wish to minimize interruptions and hence use this option sparingly. Instead, we can sample the contexts inferred by the algorithms along with the confidence value associated with the output. The inferred context values provide us with the partial observations of the underlying Markovian process. We use the Structural EM algorithm described in [7] to learn the structure of the DBN from the partial observations. In this algorithm, the structure of the DBN is improved iteratively until the MDL score of the structure B given a training data set D converges. Let us suppose a structure B consisting of n random variables $X_1, ..., X_n$. We use the notation x_i and Π_{x_i} to denote an assignment for X_i and its parent set $\mathbf{Pa}(X_i)$ respectively. Also, $N(x_i, \Pi_{x_i})$ denotes number of instances in the training data D where $X_i = x_i$ and $\mathbf{Pa}(X_i) = \Pi_{x_i}$. Let N be the total number of instances in training data. Then, the MDL score for a DBN structure B given data set D is given as:

$$\sum_i \{ \sum_{x_i, \Pi_{x_i}} E[N(x_i, \Pi_{x_i})|D] \log \theta(x_i, \Pi_{x_i}) - \frac{\log N}{2} \#(X_i, \mathbf{Pa}(X_i)) \}$$

where $\theta(x_i, \Pi_{x_i}) = \frac{E[N(x_i)|D]}{E[N(x_i, \Pi_{x_i})|D]}$ and $\#(X_i, \mathbf{Pa}(X_i)$ is the number of parameters needed to represent $Pr(X_i|\mathbf{Pa}(X_i))$. We need to take expected counts as we have access to only partial observations. We simplify the learning process by first learning the static network of the DBN followed by learning the transition edges of the network. This process is suboptimal in nature but it has been shown in practice to yield parameters estimates close to optimal.

While learning the DBN without any human input is ideal, this can lead to errors particularly if the training and test context distributions are very different. Thus, there is a need for at least some corrective human input. We randomly sample human input for a small fraction of contexts in-order to correct DBN parameters, and assign these human-provided contexts higher weight over the instances obtained using context-inference algorithms.

4.4 Classifier Personalization

So far, our discussion has assumed that the underlying context classifiers are black-boxes *i.e.* they do not expose their internal behavior or allow changes. Thus, the DBN is limited to the context output and uncertainty provided by the classifiers. A natural question is whether we can do better if we were able provide *feedback* from the DBN to the classifier in-order to improve its performance further.

In *CQue*, any human input provided to the DBN can not only be used to correct DBN parameters, but also be used to personalize the classifiers. A classifier personalized for an individual user is beneficial as it provides higher classification accuracy and better uncertainty estimates. The use of personalized classifiers also benefits the DBN — with personalized classifiers, the correct DBN parameters can be learnt using fewer future human input resulting in shorter training period. *CQue* provides an API to the developers of context classifiers to receive human-provided labels that can be leveraged for personalization.

5. IMPLEMENTATION

We now describe the implementation of *CQue* on Android smartphones running Android OS version 2.2 or higher.

Inference Engine: The *CQue* inference engine is designed to run in real-time on a mobile phone to avoid any delays incurred in accessing the cloud. The query processor maintains a model of a DBN that is personalized to the phone user. This model is stored as a file in an XMLBIF format which is an interchange format that is recognized by most bayesian inference softwares. In our implementation, we use the well known variable-elimination algorithm's implementation for probabilistic inference provided by the Java Bayes package [1]. One downside is that accurate inference using variable-elimination can take exponential time in the number of variables in the bayesian network. While this was not an issue for the number of contexts in our current implementation, this may be a bottleneck as the number of contexts increase. Future implementations of *CQue* will use fast approximate inference algorithms such as importance sampling and MCMC simulation to optimize inference performance on the phone [21].

Query Plan: As discussed in §4.2, *CQue* uses a dynamic query plan where the decision regarding what contexts to execute given a certain energy budget is periodically re-visited to capture dynamics in the incoming context stream. Such dynamic adaptation can be expensive to perform on the phone since it depends on the number of contexts in the DBN. So, for infrequent plan evaluation or if the size of the DBN is small, our implementation performs such planning on the phone, but otherwise offloads the computation to the cloud.

DBN Learning: While the execution engine can execute on a mobile phone, learning the DBN is more computationally intensive and requires cloud support. We implemented the DBN learning algorithm using partial observations by modifying the WEKA package [9]. *CQue* offloads the process of learning the DBN to the cloud by sending the appropriate context instances to the server at the end of each day. The DBN is learnt at the end of every day in the cloud and is sent back to the *CQue* framework in XMLBIF format. Over time, the frequency of updating the DBN reduces as its structure and parameters stabilize. Also, to facilitate faster learning of DBN parameters, we maintain a configuration file that provides initial parameter estimates for the DBN, based on average numbers from a general population. The configuration file has entries for each context containing a list of positive and negative causes for it along with the conditional probability. For example, the negative cause list for the context *at office* looks like: *at home,0.99; at store,0.85; at restaurant,0.93; driving,1.0*.

Query Processor: We implemented the context engine as a background service running on android phones. This service is responsible for activating context-inference algorithms and appropriate sensors, and providing the context values generated by the algorithms to the query processor. Communication between the query processor and the context engine uses Android IPC. Our implementation currently supports semantic location contexts, social contexts and activity contexts: a) **Activity contexts**: *walking*, *driving* and *stationary*, b) **Social contexts:** *with friends*, *with colleagues* and

alone, and c) **Location contexts:** *at restaurant*, *at home*, *at office*, *at store* and *on street*.

Our implementation uses decision-tree classifiers for activity recognition. We used well-known set of features extracted from the 3-axis accelerometer and gyroscope readings. The following features were computed for accelerometer readings along each of the three axis: mean, standard deviation, mean-crossing rate, energy given as normalized sum of squared discrete FFT component, peak frequency in FFT and peak energy. In addition, we computed correlation between each pair of axes. The gyroscope readings were used to compute mean and standard deviation of angular velocity around each of the three axis. For classifying social and location contexts, we used a user-provided mapping from bluetooth devices to the social context and a mapping from WiFi access points to the corresponding semantic location (either user provided or using publicly available databases). Unlike activity contexts, the classified social and location contexts have higher confidence associated with them. While we restricted ourselves to the above contexts due to the nature of our datasets, our architecture is general is designed to accommodate other context-engines that may be added by practitioners.

User Interface: The *CQue* framework provides an interface to the user to specify an energy budget and preferences for interruption (during training) and privacy. The energy budget can be specified as a fraction of a full battery that may be used for running *CQue* and a battery threshold below which *CQue* should be stopped. For interruption preferences, a user can specify the total interrupts permitted per day while training the DBN. For privacy preferences, a user can select private contexts, select a level of protection against adversary from low, medium and high where these values map to values for threshold for change in adversarial confidence $\delta = 0.4$, $\delta = 0.25$ and $\delta = 0.05$ respectively. Additionally, user can select the applications for which these preferences apply.

6. EXPERIMENTAL RESULTS

In this section, we describe the set of experiments performed to evaluate our framework. We first describe the data sets and the evaluation metrics used for the experimental evaluation. Next, we present our analysis of privacy for various contexts.We then look at the energy-accuracy tradeoffs and demonstrate the benefits of using DBN over context-classifiers using a set of experiments, and conclude with an evaluation of how various parameters such as delay, and interruption budget impact results.

6.1 Data Sets and Evaluation Metrics

Datasets.

In order to conduct our experiments, we used following two datasets:

Reality Mining Dataset: This data set contains data collected continuously for 100 students and staff at MIT over academic year 2004-2005. This data provides various contexts like user's location (work, home, other) based on cell-tower observations, social contexts based on proximity of bluetooth devices and physical activities like stationary, walking and driving based on self-reports. The duration of user traces varied from 30 days to 269 days. In our evaluation, we used data from 37 users who had data for at least 10 weeks. The traces for each of these users contains 3 location contexts and 3 activity contexts. In addition, we derive social contexts using clustering of bluetooth devices that appear closer in time, thus the number of social contexts varied from user to user and there were at least 2 such contexts.

Activity Dataset: While the reality mining dataset has rich multi-sensor data, it relies on self-reports for activity classification, hence it does not allow us to understand how accuracy of activity classifiers can be improved through the use of *CQue*. To address this, we collected a trace of user contexts for two weeks from seven users. Each of the seven users provided about 50 context labels per day. Since there could be labeling error, we corrected activity context information using GPS information and manual correction at the end of each day. All users providing data were graduate students. We used data from four users to train the activity classifiers. For testing, we used these classifiers to classify activities for the remaining three users. For each user in the test set, we divided their data into two weeks — in the first week, a personalized user-specific DBN is trained using the outputs of the classifier algorithms, and a few human labels obtained from the ground-truth set (depending on the interruption limits, and label selection scheme), and the data in the second week is used to evaluate *CQue*.

Set of Contexts.

In both the datasets, we have a set of contexts that includes location contexts like *at home*, *at office*, *on street* and three activity contexts, namely, *walking, driving* and *stationary*. For social contexts, we use the groups of bluetooth devices in the close proximity of a user as the social context. In Activity dataset, we have these groups explicitly labeled as *friends, colleagues, roommates, spouse* whereas in Reality mining dataset, we do not have explicit labels but we identify these groups as *group-1, group-2* and so on. For each context, the corresponding classifier outputs *true* or *false* value along with the confidence value associated with the classification output. We use these values as input in DBN.

Evaluation metrics.

We use three performance metrics in our evaluation: 1) **Accuracy**, which is computed as the fraction of these contexts that are correctly classified, 2) **Confidence**, or the probability of the most likely value for the context provided by the DBN, and 3) **F-Measure** which gives the harmonic mean of recall and precision (higher score is better)

6.2 Cost-Accuracy Tradeoffs using DBN

One of the benefits of a DBN is that it allows a tradeoff cost for accuracy — by understanding the relations across different contexts, an inference algorithm can be turned off to reduce overhead. While the "cost" can be different depending on the sensing and communication needs of the inference algorithm, or the burden of user input, we use a simplistic model where we assume that all classifiers have equal cost. This model provides an intuitive understanding of the cost-accuracy tradeoffs.

Classifiers < Queries. We first look at the case where the system runs fewer classifiers than the number of queries and exploits the context-relationships to answer context queries for which there are no observations coming from the classifier. This case demonstrates that the DBN can be used to answer the context queries in expectation which is not possible otherwise using classifiers alone. We evaluate this model using the Reality Mining dataset.

In this experiment, we consider all the contexts that we support to be in the query set. We then vary the budget such that it can execute between 1 to 5 classifiers. Based on the budget, the set of classifiers are chosen that can provide maximum information gain as described in §4.2.

Figure 4 shows the accuracy and F-measure averaged over 37 users as the number of classifiers increases. We see that both these

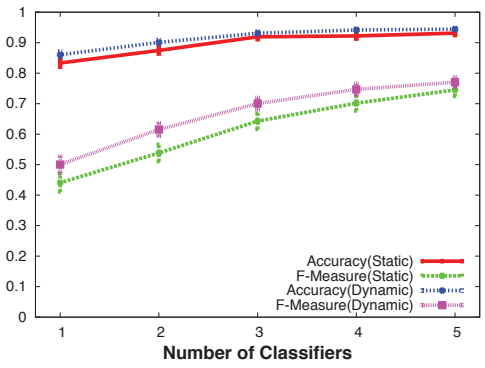

Figure 4: Effect of varying number of executing context classifiers when number of queries (≥ 6) is higher than the number of executing contexts. The figure shows aggregate accuracy and F-measure over 37 users for static and dynamic plan execution.

metrics improve as we increase the number of classifiers, and even with two classifiers executing, we get fairly good accuracy and F-measure. Thus, significant benefits can be obtained with only a small number of classifiers by leveraging a DBN model of the relations across contexts.

It might seem surprising that very high accuracy can be achieved even with a single classifier. In fact, this is because most contexts have biased distribution (e.g. the stationary state is true ($> 90\%$) of the time), hence high accuracy can be achieved by just using the model in expectation with no classifiers executing! But this gives poor recall, precision and consequently the F-measure is low, and more context input is required to improve these metrics. Figure 4 shows that DBN achieves 75% F-measure and 94% accuracy using only 4 classifiers resulting in at least 33% cost reduction.

Figure 4 also compares the use of a dynamic plan where the query plan is re-evaluated every 20 minutes vs a static plan where the query plan is evaluated once. The dynamic plan provides better accuracy and F-measure than static plan across the board. This is because the dynamic plan selects appropriate set of classifiers using value of information provided by the observed classifier outputs in real-time whereas static plan selects the set of classifiers without considering the real-time classifier outputs.

Classifiers > Queries. We now look at the benefits of using a DBN when the number of classifers that execute are greater than the number of queries. We use the Activity dataset in this study, since we have raw data for activity inferences, which are typically the contexts with highest inaccuracy. We look at the performance for the three activity contexts (walking, driving and stationary), which have highest uncertainty. We compare two mechanisms: a) using just the activity recognition classifiers, and b) using all the context classifiers with DBN. We show results for three representative users in the dataset.

Figure 5(a) shows that accuracy is worst when we use only classifiers alone, and the use of DBN improves improve accuracy significantly. The improvement in accuracy by using DBN for users U1 and U2 is above 24% for both *stationary* and *driving* contexts. Improvements are small for *walking* activity (2%) since DBN neither observed any context strongly related with *walking* nor the temporal consistency. This was because *walking* was a rare context and rarely lasted for longer than a few minutes.

Figure 5 (b) shows F-measure for various classes. We see that this metric generally improves since the main role of the DBN is to

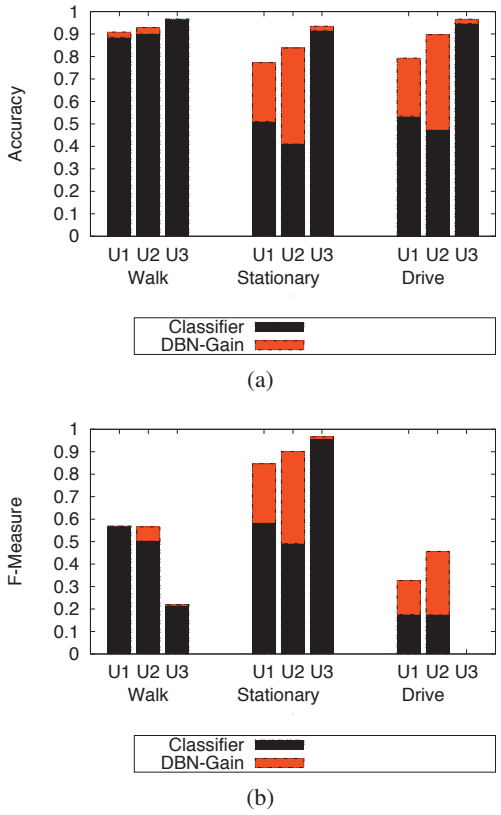

Figure 5: Accuracy and F-Measure for various activity classes for Classifier only and gain provided by Classifier+DBN mechanism on data from 3 users. Results show that using the DBN is beneficial in most cases, except in the case of rarely occurring contexts.

correct the outputs provided by the classifiers and hence, improve recall and precision. Since the DBN performs corrections using the temporal consistency, the recall value for a certain context may drop if the context is rarely seen and lasts for a very short duration. In our traces, we observed that the *walking* activity was infrequent. In such a case, if the classifier does not generate output with high confidence the DBN can smooth out the walking context resulting in reduced recall but higher precision. As a result, F-measure does not see significant gain. In contrast, the *driving* context though rarely seen as compared to *stationary* lasts longer and hence, sees improvement in recall and hence, improves F-measure.

6.3 Evaluating Privacy

In this section, we evaluate two aspects of how *CQue* can be useful in dealing with privacy breaches due to correlations across contexts.

Location-Social Relations: In our first experiment, we look at a two sensor scenario and understand the correlations between these contexts across different datasets. Recall that location is obtained through GPS and social interactions through bluetooth. We use *CQue* to understand how a privacy breach can occur for a *privacy-sensitive* context through indirect observations. Here, we assume a *strong adversary* that has access to the personalized DBN for a user. Potentially, this is possible if various apps on a phone that observe different contexts decide to collude and combine their data to generate a complete DBN.

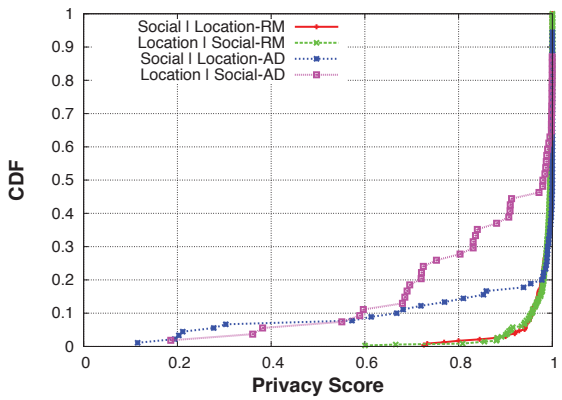

Figure 6: Cumulative distribution of privacy scores for cross-context pairs in i)Reality Mining dataset (RM), and ii)Activity dataset(AD). From the distribution of privacy scores, we can conclude that there is significant cross-context privacy leakage in Activity dataset whereas no such leakage is observed in Reality Mining dataset. This emphasizes the importance of personalized DBN to evaluate privacy for a user.

Given such an adversary, a *privacy-sensitive* context can be inferred by observing one or more other contexts. Using our *privacy score*, we can rank contexts in the decreasing order of its capability to infer *privacy-sensitive* context. In this experiment, we look two scenarios: a) all cases where we use one location context as a private context (p) and a social context o as observation, and b) the reverse scenario where a social context are private and location is observed. We evaluate *privacy score* $D(p|o)$ for each pair $\langle p, o\rangle$ of contexts. Figure 6 gives distribution of privacy scores for these scenarios for the Reality Mining and Activity Datasets.

The results are interesting — we see that there is considerably higher correlation across location and social context in the case of the activity dataset than in the reality mining data. This is also reflected in Table 1, which shows the accuracy of inferring location/social only based on the prior distribution of these contexts v.s. observing the other context. The results show that there is only a small change in the case of the RM dataset, but the accuracy increases by more than 25% in the case of the AD dataset. In other words, an adversary would be able to infer an individual's location with substantial accuracy if they only had access to the bluetooth information on the AD dataset. Our explanation for these results is that bluetooth usage is far more prevalent in recent times, therefore the correlations have increased. Overall, our results show that *CQue* can be used to provide intuition about the correlations across contexts, thereby enabling more informed decision about what to expose.

Suppression Policy for Privacy: In our second experiment, we look at how *CQue* can be used to implement a real-time suppression policy for protecting privacy. Our policy is intended to be representative and illustrate how the DBN may be used, and we make no formal claims regarding its privacy properties. In our experiment, the user can define a policy that permits releasing a maximal set of context observations such that the change in adversarial confidence for private context upon observing this set is less than threshold δ. As a result, some of the non-private contexts can be suppressed in real-time to control the change in adversarial confidence. Thus, it results in lower utility for the applications seeking contexts. Figure 7 gives an example showing the fraction of context data that can be

(a) Privacy leakage in All Location Contexts

Metric	Observation	
	None	All social contexts
Accuracy (RM)	73.9±3.0	74.81±2.73
Accuracy (AD)	61.11±5.55	88.9±6.26

(b) Privacy leakage in All Social Contexts

Metric	Observation	
	None	All location contexts
Accuracy (RM)	73.86±2.9	74.18±2.82
Accuracy (AD)	64.01±15.29	87.3±5.94

Table 1: Cross-context privacy leakage in location and social contexts for i)Reality Mining dataset(RM), and ii)Activity dataset(AD). For activity dataset, we see significant increase in accuracy for all the location contexts when all the social contexts are observed and vice versa. For reality mining dataset, accuracies do not change with cross-context observations.

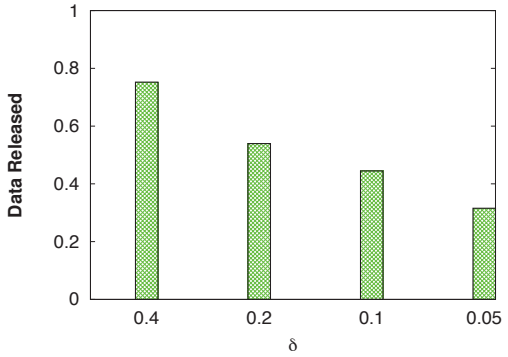

Figure 7: Fraction of data that can be released to prevent change in adversarial confidence greater than δ for a private context *at home*.

released as a result of this suppression for various values of δ where we chose *at home* as the private context and rest of the contexts as queries. A smaller value of δ results in much tighter privacy control but releases too few contexts.

Table 2 shows the fraction of data suppressed in context of each category: *social, location,* and *activity* for value of threshold $\delta \in \{0.1, 0.4\}$. In this case, we choose *at friend's place* as a private context available in Activity dataset. We can see that *CQue* suppresses in an intelligent manner where it suppresses highly correlated social context more frequently than the less correlated location or activity contexts. If we were to use a mechanism like MaskIT [8], it would suppress every context at the same level. Since *CQue* releases more contexts, it results in higher utility for the applications and the users who use these applications.

In conjunction, these experiments demonstrate the potential use of *CQue* both for understanding privacy implications of releasing a context, as well as to implement privacy policies that leverage relations across contexts.

6.4 DBN with Personalized Classifiers

While all our previous results have assumed classifiers to be black-box code that cannot be modified, we now look at the case where we can personalize the classifiers using the human input ob-

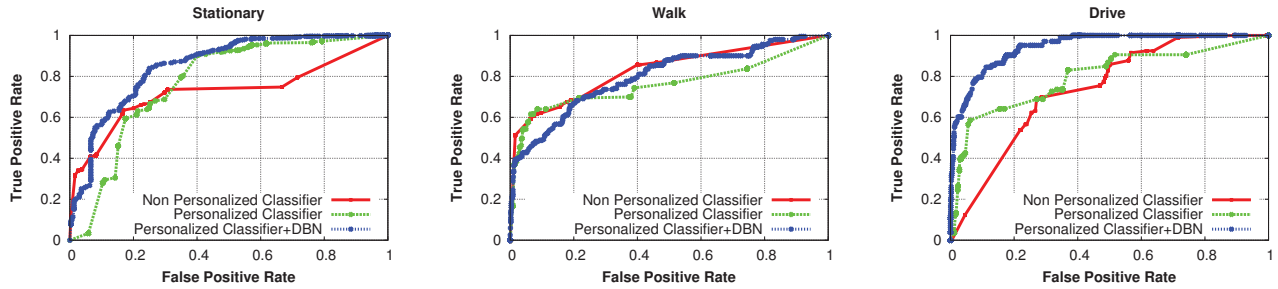

Figure 8: ROC curve for each activity class for i) non-personalized classifier, ii) personalized classifier, and iii) DBN using personalized classifier.

Threshold δ	Social Context	Location Context	Activity Context
0.4	27.67%	13.99%	17.52%
0.1	26.81%	16.01%	17.35%

Table 2: Table showing fraction of data suppressed in contexts of each category to prevent change in adversarial confidence greater than δ for a private context *at friend's place*.

tained by the DBN during its learning phase. This feature leverages the fact that contexts like user activity can see significant benefit with personalization as there are differences across individuals.

In our experiment, we personalize the classifiers for activity contexts in AD dataset by training it with additional user-specific labels obtained from the user's data for training DBN i.e. the first week of user's trace. Figure 8 gives ROC curves for each activity class for both personalized and non-personalized classifiers. The ROC curve shows the performance of the classification mechanism when the discrimination threshold is varied. A classification mechanism with a larger area under the ROC curve indicates that it is a better mechanism. In figure 8, it can be seen that the personalized classifier tends to have larger area under the ROC curve compared to the non-personalized classifiers. Moreover, the area under ROC curve for DBN is always larger than the area for either of the classifier-based schemes. Except for *walking* which has a small operating region where the classifier performs better than DBN, the DBN clearly dominates the classifiers over all operating regions in the ROC. This shows that classifier personalization together with a DBN can provide the best method to improve accuracy.

6.5 Impact of Delay

The delay tolerance period provided by queries provides a tuning knob that can influence results provided by the query engine. In this experimental setup, we study the impact of increasing value of this parameter on activity contexts in AD dataset. In general, whenever the DBN observes a change in context value output by the classifiers, the confidence of the DBN may drop. This confidence improves with more observations obtained in the future. Thus, the main role of using a delay-tolerance period is to improve confidence in the DBN output. Figure 9 shows the distribution of confidence when delay-tolerance is increased. Apart from boosting confidence, the use of a delay-tolerance window reduces the intermittent misclassifications by the classifiers.

6.6 Implementation Benchmarks

We benchmark our implementation on Samsung Galaxy Nexus phone running Android OS version 4.1.2. This device has 700MB

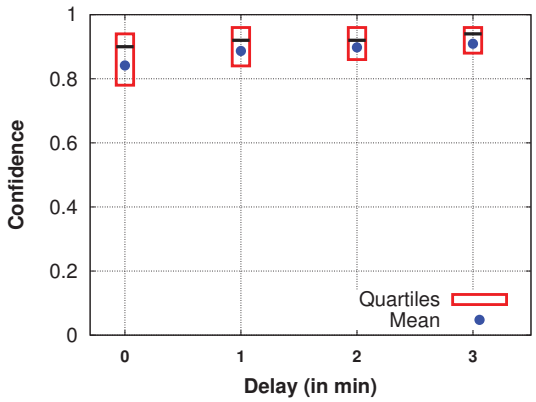

Figure 9: Effect of delay on confidence distribution, as we increase delay-tolerance for queries from zero to three minutes. We consider all the query results that had conf<0.98 when DBN was used with 0 delay in response (35% of the results), and show how summary statistics for this group improves as we increase the delay-tolerance for the queries from 0 to 3.

RAM and 1.2 GHz dual-core processor. In this evaluation, we look at the following two metrics: i) *inference time* i.e. the time required to make a probabilistic inference about a context given a set of evidences from the classifiers, and ii) *plan evaluation time* i.e. the time required to generate a dynamic plan for a given set of queries. All experiments are done using the Reality Mining dataset.

Inference Time: Since the computational complexity of an accurate inference algorithm to infer a context is exponential in number of parent nodes for the context in DBN, the actual required time for probabilistic inference varies from context to context and from user to user due to variations in the personalized DBN model structure. Thus, we measure *inference time* using traces from multiple users with varying size of the DBN model. In this experiment, we select one context as a query and use rest of the contexts as observations. We repeat this experiment for all the available contexts for each user. The user traces consisted of 5-11 contexts. Upon evaluation on Galaxy Nexus phone, we observed an average *inference time* across all the contexts and the users to be 9.31 ± 0.71ms. Also, the fastest context had an average *inference time* of 2.13ms for a user trace consisting of 7 contexts whereas the slowest context had an average *inference time* of 21ms for a user with DBN model consisting of 11 contexts.

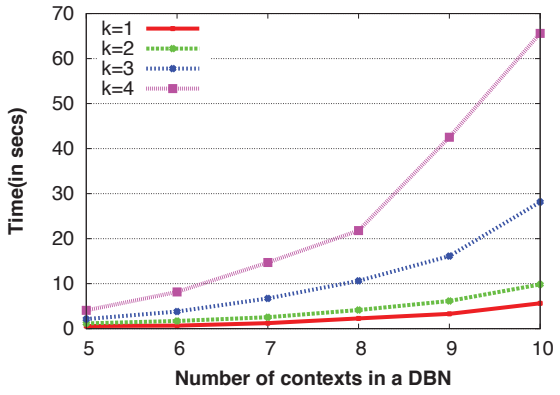

Figure 10: Time to generate a dynamic plan selecting best k classifiers where k varies from 1 to 4 and the query set consists of all the contexts available in a DBN. The figure shows the variation in time as the number of contexts in a DBN, N is varied from 5 to10. The plan evaluation time increases linearly in k and exponentially in N.

Plan Evaluation time: We evaluate *plan evaluation time* on Galaxy Nexus phone for multiple user traces consisting of 5-11 contexts. In this evaluation, we set all the contexts as queries and measure time to generate dynamic plan selecting best k classifiers where $k = 1, 2, 3, 4$. We note that the time complexity of generating dynamic plan with best k classifiers is linear in k and exponential in N where N is the number of contexts in the DBN. We show the results in Figure 10. From the figure, we can see that the *plan evaluation time* increases quickly with the number of contexts in DBN and requires more than a minute to evaluate best 4 classifiers for a DBN consisting 10 contexts. This might still be practical if we generate a plan only when the set of queries change, which might be infrequent. However, if the plan is re-evaluated every few minutes, this approach quickly becomes infeasible. This problem can be addressed in two ways: i) using an approximate inference algorithm with better time complexity, which is part of our ongoing work, or ii) evaluating the plan in the cloud when its complexity exceeds a certain threshold. Of course, dynamic plan evaluation on the cloud has the overhead of data transfer of the current classifier outputs with the appropriate confidence values. However, the DBN model itself is not required to be sent to the cloud as it is already trained on the cloud.

7. DISCUSSION

In our results, we have demonstrated the utility of *CQue* framework for energy-efficient and accurate continuous sensing of user context while understanding the privacy implications. In this section, we discuss potential opportunities for improvement and various interesting questions posed by our framework for further exploration.

Feature-level DBN structure: Our current implementation of the DBN supports fusion of high-level categorical contexts that are the outputs of classifiers. While this is beneficial in that we can take advantage of extensive work on the design of individual activity or behavior classifiers, the downside is that if a classifier outputs inaccurate uncertainty estimates, we may have incorrect inferences that can impact the accuracy of all the related contexts. One way to address this issue may be to use both categorical contexts from classifiers as well as low-level signal features extracted from the sensor data (e.g. spectral features, mean, variance, etc), and to use combination of these different inputs towards information fusion for context inference.

Privacy leakage: We have touched upon the issue of privacy leakage by leveraging correlations across contexts, but *CQue* does not capture all possible attacks that can use multi-sensor inference techniques. The privacy leak detection in *CQue* for a sensitive context is based on a model of adversary who is aware of the context relationships in DBN, but it is possible to have an undetected privacy leak if there are relationships that cannot be modeled by the DBN. For example, the DBN can model and evaluate privacy leakage over a short window of W adjacent time steps but it cannot detect privacy leaks before W time steps. A powerful adversary with alternative long-term models may still be able to execute privacy attacks. Thus, much work remains before we fully understand how to effectively understand the extent of privacy leakage that is possible using sophisticated multi-sensor inference techniques.

Duty-cycling classifiers: One question that we have not fully explored is the interaction between duty-cycled classifiers and the DBN. Several techniques have been proposed to leverage temporal consistency in classifiers for duty-cycling *i.e.* if the context value is not expected to change for some time period, then reduce the sampling rate of the sensor. *CQue* does not currently take advantage of such approaches to duty-cycle classifiers. One method to incorporate such approaches is to use the last observed classifier output as evidence in DBN for next few time-steps while the classifier is inactive. Another interesting question is whether the DBN can be used to learn the duty-cycle for each classifier. Intuitively, if the current state of DBN indicates that some context, say c, is unlikely to change soon then we can have a lower duty-cycle for context c where its classifier will remain inactive for a long time.

DBN learning time: How much training data is needed to fully learn an accurate DBN structure? Although one week of trace proved to be sufficient to learn good structures in our experiments, it might not be the case in general since there may be some longer-range patterns that cannot be captured, and the patterns may change over time. To answer this question, we need larger-scale datasets across more individuals that can help us understand how to learn the DBN effectively across a population while limiting burden to provide labels, as well as the learning period.

8. CONCLUSIONS

Context awareness distinguishes smartphones from traditional computing platforms and can play a key role in creating smart mobile applications. We argue that user contexts across several dimensions are correlated, and if we can learn these correlations in a personalized manner, we can leverage it to improve performance as well as understand the privacy implications of revealing seemingly unrelated contexts. In this paper, we described our context querying framework, *CQue*, that exploits probabilistic relationships across contexts for each individual user to improve energy-efficiency, accuracy and reliability of context sensing, as well as the ability to understand privacy implications.

The ability to understand relations across contexts can have far-reaching consequences. A growing area of healthcare is personalized behavioral monitoring using smartphones and on-body sensors, where behavioral scientists seek to understand relations between addictive or unhealthy behavior, and the individual's state in the real world (activities, social interactions, location, etc) . Our work can enable such understanding, and is a step towards a "big data inference toolkit" for mobile sensing.

9. ACKNOWLEDGEMENTS

We thank our shepherd, Nicholas Lane, and the anonymous reviewers for their comments. This research was supported by UMass S&T award and NSF grants CNS-0910900 and CNS-0855128.

10. REFERENCES

[1] Java bayes. *http://www.cs.cmu.edu/ javabayes/*.

[2] L. Bao and S. S. Intille. Activity Recognition from User-Annotated Acceleration Data. *Pervasive Computing*, pages 1–17, 2004.

[3] D. Chu, A. Deshpande, J. M. Hellerstein, and W. Hong. Approximate Data Collection in Sensor Networks using Probabilistic Models. In *ICDE*, page 48, 2006.

[4] D. Chu, N. Lane, T. Lai, C. Pang, F. Li, X. Meng, Q. Guo, and F. Zhao. Balancing energy, latency and accuracy for mobile sensor data classification. In *SenSys*, pages 54–67, 2011.

[5] A. Deshpande, C. Guestrin, W. Hong, and S. Madden. Exploiting correlated attributes in acquisitional query processing. In *ICDE*, pages 143–154, 2005.

[6] A. Deshpande, C. Guestrin, S. Madden, J. Hellerstein, and W. Hong. Model-driven data acquisition in sensor networks. In *VLDB*, pages 588–599, 2004.

[7] N. Friedman, K. Murphy, and S. Russell. Learning the Structure of Dynamic Probabilistic Networks. In *UAI*, pages 139–147, 1999.

[8] M. Götz, S. Nath, and J. Gehrke. Maskit: privately releasing user context streams for personalized mobile applications. In *SIGMOD*, pages 289–300, 2012. ACM.

[9] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The weka data mining software: An update. In *SIGKDD Explorations*, pages 10–18, 2009.

[10] A. Krause and C. Guestrin. Near-optimal Nonmyopic Value of Information in Graphical Models. In *UAI*, pages 324–331, 2005.

[11] J. Lester, T. Choudhury, N. Kern, G. Borriello, and B. Hannaford. A hybrid discriminative/generative approach for modeling human activities. In *IJCAI*, pages 766–772, 2005.

[12] M. Li, D. Ganesan, and P. Shenoy. Presto: feedback-driven data management in sensor networks. In *NSDI*, page 23, 2006.

[13] H. Lu, J. Yang, Z. Liu, N. D. Lane, T. Choudhury, and A. Campbell. The jigsaw continuous sensing engine for mobile phone applications. In *SenSys*, pages 71–84, 2010.

[14] A. Meliou, C. Guestrin, and J. M. Hellerstein. Approximating sensor network queries using in-network summaries. In *IPSN*, pages 229–240. IEEE, 2009.

[15] S. Nath. Ace: exploiting correlation for energy-efficient and continuous context sensing. In *Proceedings of the 10th international conference on Mobile systems, applications, and services*, MobiSys '12, pages 29–42, New York, NY, USA, 2012. ACM.

[16] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An analysis of approximations for maximizing submodular set functions-I. *Mathematical Programming*, 14(1):265–294, Dec. 1978.

[17] A. Parate and G. Miklau. A framework for safely publishing communication traces. In *Proceedings of the 18th ACM conference on Information and knowledge management*, CIKM '09, pages 1469–1472, New York, NY, USA, 2009. ACM.

[18] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Francisco, USA, 1988.

[19] R. A. Popa, H. Balakrishnan, and A. J. Blumberg. Vpriv: protecting privacy in location-based vehicular services. In *Proceedings of the 18th conference on USENIX security symposium*, SSYM'09, pages 335–350, Berkeley, CA, USA, 2009. USENIX Association.

[20] N. Ravi, N. Dandekar, P. Mysore, and M. L. Littman. Activity recognition from accelerometer data. In *IAAI*, pages 1541–1546, 2005.

[21] S. J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, second edition, 2002.