Usher: Improving Data Quality With Dynamic Forms

Kuang Chen
UC Berkeley
kuangc@cs.berkeley.edu

Harr Chen
MIT
harr@csail.mit.edu

Neil Conway
UC Berkeley
nrc@cs.berkeley.edu

Joseph M. Hellerstein
UC Berkeley
hellerstein@cs.berkeley.edu

Tapan S. Parikh
UC Berkeley
parikh@ischool.berkeley.edu

ABSTRACT

Data quality is a critical problem in modern databases. Data entry forms present the first and arguably best opportunity for detecting and mitigating errors, but there has been little research into automatic methods for improving data quality at entry time. In this paper, we propose USHER, an end-to-end system for form design, filling, and data quality assurance. Using previous form submissions, USHER learns a probabilistic model over the questions of the form. USHER then applies this model at every step of the data entry process to ensure high quality. Before entry, it induces a form layout that captures the most important data values of a form instance as quickly as possible. During entry, it dynamically adapts the form to the values being entered, and provides real-time feedback to guide the data enterer toward more likely values. After entry, it re-asks questions that it deems likely to have been entered incorrectly. We evaluate all three components of USHER using two real-world data sets. Our results demonstrate that each component has the potential to improve data quality considerably, at a reduced cost when compared to current practice.

1. INTRODUCTION

Governments, companies, and individuals routinely make important decisions based on inaccurate data stored in supposedly authoritative databases. In medicine, for instance, a single input error may have fatal consequences. Data quality can and should be addressed at every stage of the data lifecycle. While there has been extensive work on improving data quality within databases via *data cleaning* [8, 2], the database community has paid relatively little attention to how *data entry* mechanisms can improve quality.

In data entry form design and execution, traditional practices based on paper forms have prevailed for decades. The *Survey Design* literature contains many longstanding principles about data encodings, manually specified constraints, and static post-entry validation [11]. For electronic forms, ensuring data quality *during* data entry has centered around the ubiquitous and costly practice of *double-entry* [9]: enter everything twice, compare, and flag discrepancies for human arbitration.

For many organizations, particularly those operating with limited resources, double-entry is neither practical nor attainable. Recent work on data collection in resource-poor settings reports that lack of expertise and difficulty of remote data collection are the chief obstacles to high data quality [6]. Too often in such settings, form design is an *ad hoc* practice, consisting of mapping desired information to a set of entry widgets (text fields, combo boxes, etc.), guided only by the designer's intuition [21]. In our field work with a HIV/AIDS treatment program in Tanzania, we observed that little thought was given to form design, and a haphazard double-entry program bottlenecked the data entry process. Only after a significant entry lag were researchers able to enjoy the benefits of data in electronic form. The cost and overhead involved in double-entry meant that patients and clinicians at the point of care were limited to using paper forms, and the benefits of automated data analysis were not available to the core population the organization was chartered to serve.

To address such limitations, we have developed USHER, an end-to-end system that uses previously-entered data to improve the quality of data at the point of entry. USHER learns a *probabilistic model* from existing data, which stochastically relates the questions of a data entry form. This model is used to guide the data entry process in three ways:

1. The model is used to automatically lay out a form by selecting an optimal *ordering* for the questions, according to a probabilistic objective function that aims to maximize the information content of a form submission as early as possible.

2. During entry, the model's probability estimates are used to dynamically reorder questions, again to maximize information gain, and to provide context-dependent feedback that guides the user towards more likely answers.

3. After the submission of a complete form instance, the model is consulted to predict which responses may be erroneous, so as to *re-ask* those questions in order to verify their correctness. This focused re-asking achieves the benefits of double entry at a fraction of the cost.

Submitted for publication. Version as of 3/20/2009. Contact author: K. Chen.

USHER is an effort to push modern probabilistic techniques for data cleaning to the very beginning of the data lifecycle, when it is easiest to identify dirty data and replace it with ground truth.

The contributions of this paper are fourfold.

1. We describe how we build two probabilistic models for an arbitrary data entry form: first, we learn a Bayesian network over questions using structure learning and parameter estimation; and second, we expand this network into another model that directly captures form entry errors.

2. We describe how USHER uses these models to provide three forms of guidance: static form design, dynamic question ordering, and re-asking.

3. We present experiments showing that USHER has the potential to improve data quality at reduced cost on two representative data sets: responses to a survey about politics and race, and patient information records from an HIV/AIDS clinic in Tanzania.

4. Extending our ideas on form dynamics, we propose new user interface annotations that provide contextualized, intuitive feedback about the likelihood of data as it is entered. These ideas incorporate data cleaning visualizations directly into the entry process.

2. RELATED WORK

Our work builds upon several areas of related work. We provide an overview in this section.

2.1 Data Cleaning

In the database literature, data quality has typically been addressed under the rubric of *data cleaning* [2, 8]. Our work connects most directly to data cleaning via multivariate outlier detection; it is based in part on ideas first proposed in [14]. By the time such retrospective data cleaning is done, the physical source of the data is typically unavailable — thus, errors often become too difficult or time-consuming to be rectified. USHER addresses this issue by applying statistical data quality insights to data entry. Thus, it can catch errors when they are made, and when ground truth values are still available for verification.

2.2 User Interfaces

In the area of adaptive user interfaces, Gajos et al. [10] also cast user interface design as an optimization problem. Their goal is to improve *usability*, by using decision-theoretic optimization to automatically adapt user interfaces to specific devices, usage patterns, user preferences and capabilities. In contrast, USHER's focus is on improving data *quality*, and its probabilistic formalism is based on learning relationships within the underlying data that guide the user towards more correct entries.

A recent technical report by Ali and Meek also addresses the task of form-filling [1]. Like us, they propose to learn probabilistic models to predict values for questions, which they use to automatically populate "drop-down" lists. Their focus is on the *speed* of data entry, as opposed to our emphasis on data quality. Moreover, in addition to predicting question values, we develop and exploit probabilistic models of user error, and target a broader set of interaction design issues relevant to data quality, including question reordering and reasking, and interface customizations for data entry feedback. Some of the enhancements we make for data quality could also be applied to improve the speed of entry.

In a related approach, EcoPod [22] attempts to make data collection easier for amateurs contributing to nature conservation. They take an information-theoretic approach to generating type-ahead suggestions on a PDA to minimize work for the user. The suggestions are based on historical observations of a geographic area. Such context-aware use of previous data can be seen as a domain-specific instantiation of USHER's approach, but with a different aim: reduction of effort. They leave data quality as future work, which is where USHER begins.

2.3 Clinical Trials

Data quality assurance is a prominent topic in the science of clinical trials, where the practice of double-entry has been questioned and dissected, but nonetheless remains the gold standard [9, 16]. In particular, [17] takes a probabilistic approach towards choosing which forms to re-enter based on the individual performance of data entry staff. This *cross-form* validation has the same goal as our approach of reducing the need for complete double entry, but does so at a much coarser level of granularity. It requires historical performance records for each data enterer, and does not offer dynamic reconfirmation of individual questions. In contrast, USHER's *cross-question* validation adapts to the actual data being entered in light of previous form submissions, and allows for a principled assessment of the tradeoff between cost (of reconfirming more questions) versus quality (as predicted by the probabilistic model).

2.4 Survey Design

The survey design literature includes extensive work on proper form design for high data quality [11, 20]. This literature advocates the use of manually specified *constraints* on response values. These constraints may be univariate (e.g., a maximum value for an *age* question) or multivariate (e.g., disallowing *gender* to be *male* and *pregnant* to be *yes*). Some constraints may also be "soft", and only serve as warnings regarding unlikely combinations (e.g., *age* being 60 and *pregnant* being *yes*).

The manual specification of such constraints requires a domain expert, which can be prohibitive in many scenarios. By relying on prior data, USHER learns to automatically infer many of these same constraints without requiring their explicit specification. When these constraints are violated during entry of a new submission, USHER can then flag the relevant questions, or target them for re-asking.

Additionally, previous research into the psychological phenomena of survey filling has yielded common constraints not inherently learnable from prior data [11]. This work provides heuristics such as "groups of topically related questions should often be placed together", and "questions about race should appear at the end of a survey." While prior data cannot be used to infer these constraints, USHER can accommodate them directly in its question ordering framework.

3. SYSTEM OVERVIEW

In this section, we give a brief overview of the system and illustrate its functionality with an example. Further details, particularly regarding the probabilistic model, follow in the

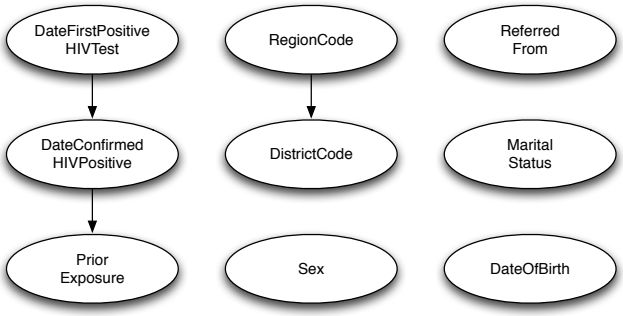

Figure 1: Example network for the HIV/AIDS domain, showing probabilistic relationships between form questions.

ensuing sections.

3.1 Design

Our running example is a patient-registration data collection form actively used by a network of HIV/AIDS clinics in Tanzania.[1] Data collection is done on paper; a team of data entry workers digitize the entered forms daily. Our data set contains 1,650 form submissions.

USHER builds a probabilistic model for an arbitrary data entry form in two steps: first, by learning the relationships between form questions via structure learning; and second, by estimating the parameters of a Bayesian network, which then allows us to generate predictions and error probabilities for the form.

In our example, a user begins by creating a simple specification of form questions and their prompts, response data types, constraints, etc. The training data set is made up of prior form responses. By running the learning algorithms we present in Section 4, USHER builds a network of dependence relationships from the data, which is shown in Figure 1: an edge in the graph captures a probabilistic dependency between two random variables (i.e., form questions). As another example, Figure 2 illustrates a denser graph learned from a set of political survey responses. Note that a standard joint distribution would show correlations among *all pairs* of questions; the sparsity of these examples reflects statistical independences learned from the data, which both clarify the structure of the form and make probabilistic inference more efficient.

Our learning algorithm works off of training data. In practice, a data entry backlog can serve as this training set. In the absence of sufficient training data, USHER can bootstrap itself on a "uniform prior," generating a form based on the assumption that all inputs are equally likely. Subsequently, a training set can gradually be constructed by iteratively capturing data from designers and potential users in "learning runs." This process of semi-automated form design can help institutionalize new forms before they are deployed in production.

Next, USHER uses the learned model to automatically *order* a form's questions to maximize information gain, as described in Section 5. This mimics survey design principles, while respecting the form designer's topical and question-

[1]We have pruned out questions with identifying information about patients, and free-text comment fields.

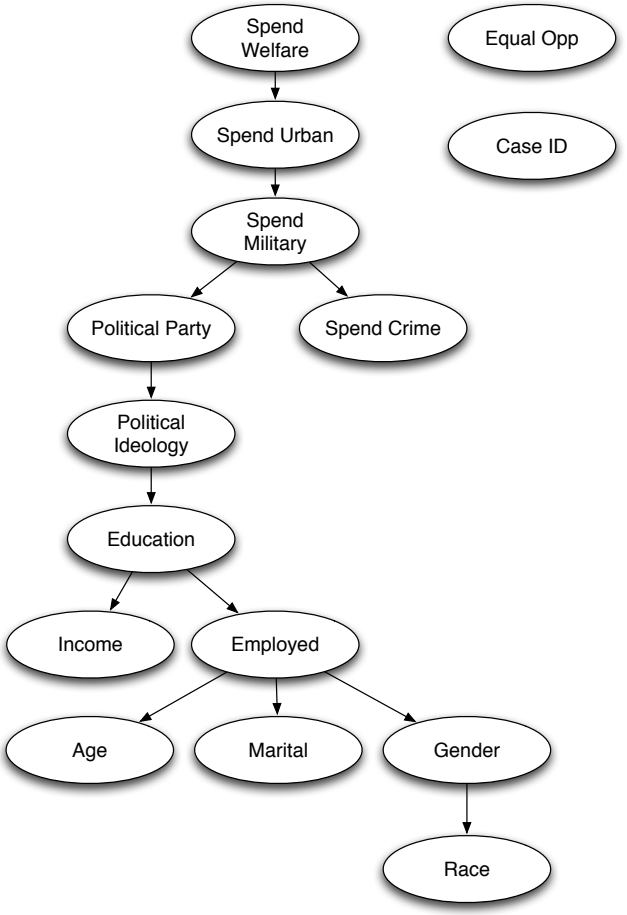

Figure 2: Example network for the survey domain. The probabilistic relationships are dense for this data set.

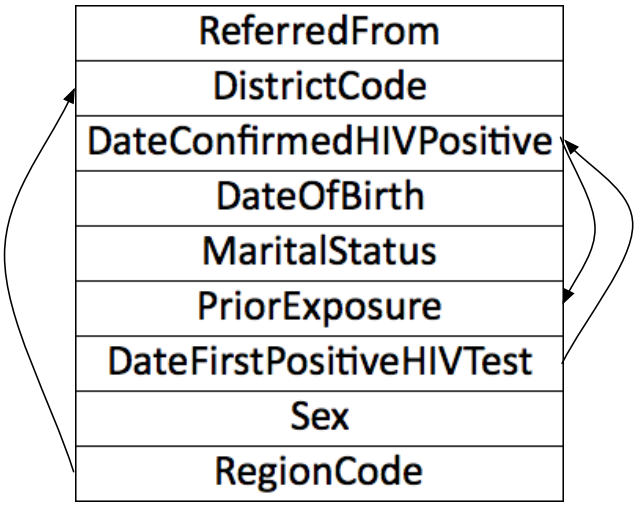

Figure 3: Example question layout generated by our ordering algorithm. The arrows reflect the probabilistic dependencies from Figure 1.

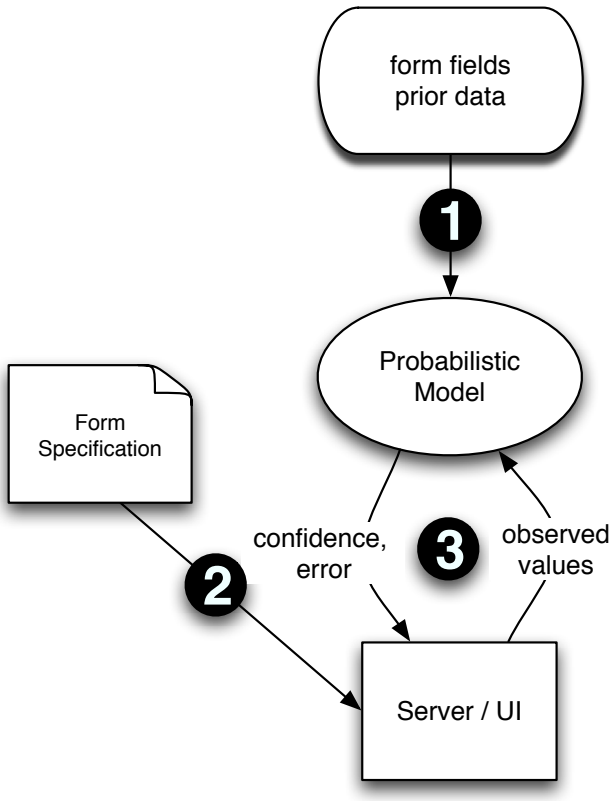

Figure 4: USHER system diagram and data flow.

ordering constraints. In our example, USHER generated the ordering found in Figure 3.

During data entry, USHER uses the probabilistic machinery to drive dynamic updates to the form. These updates can reorder unentered question widgets to better capture this particular form instance's uncertainty, or may consist of data quality hints to the user, e.g., warnings about an answer's likelihood. A discussion of feedback mechanisms is provided in Section 8.

At the end of a data entry run, USHER calculates error probabilities for the whole form and for each question. If there are responses with error probabilities exceeding some threshold, USHER re-asks those questions one by one, ordered by the highest error probability, as described in Section 6

3.2 Implementation

We implemented USHER as a J2EE web application with an Adobe Flex user interface (Figure 4). The UI loads simple form specification files which contain form question details, and the location of the training data set. Form details include question name, prompt, data type, widget type, and constraints. Constraints can be univariate, such as enumerations or valid ranges, or multivariate, specifying required orderings or groupings of subsets of questions. The server instantiates a model per form. The server passes information about question responses to the model as they are filled in; in exchange, the model returns predictions and error probabilities.

Models are created from the form specification, the training data set, and a graph of learned structural relationships. We perform structure learning offline with BANJO [12], an open source Java package for structure learning of Bayesian networks. Our graphical model is implemented in two variations: one is based on a modified version of JavaBayes [7], an open source Java software for Bayesian inference. Because JavaBayes only supports discrete probability variables, we implemented the error prediction version of our model using Infer.NET [18], a Microsoft .NET Framework toolkit for Bayesian inference.

4. LEARNING A PROBABILISTIC MODEL

The core of the USHER system is its probabilistic model of the data, represented as a *Bayesian network* over form questions. This network captures relationships between different question elements in a stochastic manner. In particular, given input values for some subset of the questions of a particular form instance, the model can predict probability distributions over values of that instance's remaining unanswered questions. In this section, we show how standard machine learning techniques can be used to induce this model from previous form entries.

We will use $\mathbf{F} = \{F_1, \ldots, F_n\}$ to denote a set of random variables representing the values of n unknown questions comprising a data entry form. We assume that each question response takes on a finite set of discrete values; continuous values can be discretized by dividing the data range into intervals and assigning each interval one value.[2] To bootstrap learning of the probabilistic model, we assume access to prior entries for the same form.

USHER first builds a Bayesian network over the form questions, which will allow it to compute probability distributions over arbitrary subsets $\mathbf{G}$ of form question random variables, given already entered question responses $\mathbf{G}' = \mathbf{g}'$ for that instance, i.e., $P(\mathbf{G} \mid \mathbf{G}' = \mathbf{g}')$. Constructing this network requires two steps: first, the induction of the graph *structure* of the network, which encodes the conditional independences between the question random variables $\mathbf{F}$; and second, the estimation of the resulting network's *parameters*.

The naïve approach to structure selection would be to assume complete dependence of each question on every other question. However, this would blow up the number of free parameters in our model, leading to both poor generalization performance of our predictions, and prohibitively slow model queries. Instead, we learn the structure using the prior form submissions in the database. In our implementation, we use the BANJO software [12] for structure learning, which searches through the space of possible structures using simulated annealing, and chooses the best structure according to the Bayesian Dirichlet Equivalence criterion [13]. Figures 1 and 2 show example automatically learned structures for two data domains.[3]

Note that in certain domains, form designers may already have strong common sense notions of questions that should be related (e.g., education level and income). Our automatic predictive approach allows for manually tuning the resulting model to incorporate such intuitions. In fact, the

[2]Our present formulation ignores the rich dependencies between ordinal values; modeling such relationships is an important direction of future work.

[3]It is important to note that the arrows in the network do *not* represent causality, only that there is a probabilistic relationship between the questions.

entire structure could be manually constructed in domains where an expert has comprehensive prior knowledge of the questions' interdependencies.

Given a graphical structure of the questions, we can then estimate the *conditional probability tables* at each node in a straightforward manner, by counting the proportion of previous form submissions with those response assignments. The probability mass function for a single question F_i with m possible discrete values, conditioned on its set of parent nodes $\mathbf{G}$ from the Bayesian network, is:

$$P(F_i = f_i \mid \{F_j = f_j : F_j \in \mathbf{G}\}) = \frac{N(F_i = f_i, \{F_j = f_j : F_j \in \mathbf{G}\})}{N(\{F_j = f_j : F_j \in \mathbf{G}\})}. \quad (1)$$

In this notation, $P(F_i = f_i \mid \{F_j = f_j : F_j \in \mathbf{G}\})$ refers to the conditional probability of question F_i taking value f_i, given that some of the other questions have already been entered — specifically, that each question F_j in set $\mathbf{G}$ takes on value f_j. $N(\mathbf{X})$ is the number of prior form submissions that match the conditions $\mathbf{X}$ — in the denominator, we count the number of times a previous submission had the subset $\mathbf{G}$ of its questions set according to the listed f_j values; and in the numerator, we count the number of times when those previous submissions additionally had F_i set to f_i.

Because the number of prior form instances may be limited, and thus may not account for all possible combinations of prior question responses, equation 2 may assign zero probability to some combinations of responses. Typically, this is undesirable; just because a particular combination of values has not occurred in the past does not mean that combination cannot occur at all. We overcome this obstacle by *smoothing* these parameter estimates, interpolating each with a background uniform distribution. In particular, we revise our estimates to:

$$P(F_i = f_i \mid \{F_j = f_j : F_j \in \mathbf{G}\}) = (1-\alpha)\frac{N(F_i = f_i, \{F_j = f_j : F_j \in \mathbf{G}\})}{N(\{F_j = f_j : F_j \in \mathbf{G}\})} + \frac{\alpha}{m}, \quad (2)$$

where m is the number of possible values question F_i can take on, and α is the fixed smoothing parameter, which was set to 0.1 in our implementation. This approach is essentially a form of Jelinek-Mercer smoothing with a uniform backoff distribution [15].

Once the Bayesian network is constructed, we can query it for distributions of the form $P(\mathbf{G} \mid \mathbf{G}' = \mathbf{g}')$ for arbitrary $\mathbf{G}, \mathbf{G}' \in \mathbf{F}$ — that is, the *marginal* distributions over sets of random variables, optionally conditioned on observed values for other variables. Performing these queries is known as the *inference* task in graphical models, for which there exist a variety of different techniques. In our experiments, the Bayesian networks are small enough that exact techniques such as the *junction tree algorithm* [5] can be used. For larger models, faster approximate inference techniques may be preferable.

5. QUESTION ORDERING

Once we have constructed the Bayesian network, we can turn to its applications in the USHER system. We first consider ways of ordering the questions of a data entry form. Our ordering selection is driven by simple information-theoretic insights. We first note that regardless of how questions are ordered, the total amount of *uncertainty* about all of the responses taken together — and hence the total amount of information that can be acquired during form-filling of a single form submission — is fixed. Thus, by reducing this uncertainty as early as possible, we can be more certain about the values of later questions. The benefits of stronger certainty about later questions are two-fold. First, it allows us to more accurately provide data entry feedback for those questions, because we are more certain about the probability distribution of their values. Second, if the data enterer is interrupted and leaves the rest of the form incomplete, we will have more reliable predictions over the remaining questions.

Input: Model $\mathcal{G}$ with questions $\mathbf{F} = \{F_1, \ldots, F_n\}$
Output: Ordering of questions $O = (o_1, \ldots, o_n)$
$O \leftarrow \emptyset$;
while *O does not include all questions* **do**
  $f \leftarrow \arg\max_{i \notin O} H(F_i \mid \{F_j : j \in O\})$;
  $O \leftarrow (O, f)$;
end

Algorithm 1: Static ordering algorithm for form layout.

We can quantify uncertainty using *information entropy*. A question whose random variable has high entropy reflects greater underlying uncertainty about the responses that question can take on. Formally, the entropy of random variable F_i is given by:

$$H(F_i) = -\sum_{f_i} P(f_i) \log P(f_i), \quad (3)$$

where the sum is over all possible values f_i that question F_i can take on.

As question values are entered for a single form instance, the uncertainty about remaining questions of that instance changes. For example, in the race and politics survey, knowing the respondent's political party provides strong evidence about his or her political ideology. We can quantify the amount of uncertainty remaining in a question F_i, assuming that other questions $\mathbf{G} = \{F_1, \ldots, F_n\}$ have been previously encountered, with its *conditional entropy*:

$$H(F_i \mid \mathbf{G}) = -\sum_{\mathbf{g}=(f_1,\ldots,f_n)} \sum_{f_i} P(\mathbf{G} = \mathbf{g}, F_i = f_i) \log P(F_i = f_i \mid \mathbf{G} = \mathbf{g}), \quad (4)$$

where the sum is over all possible question responses in the Cartesian product of $F_1, \ldots, F_n, F_i$. Conditional entropy measures the weighted average of the entropy of question F_j's conditional distribution, given every possible assignment of the previously observed variables. This value is obtained by performing inference on the Bayesian network to compute the necessary distributions. By taking advantage of the conditional independences encoded in the network, we can often drop terms from the conditioning in equation 4 for faster computation.

Conditional entropy can also be expressed as the incremental difference in joint entropy due to F_i:

$$H(F_i \mid \mathbf{G}) = H(F_i, \mathbf{G}) - H(\mathbf{G}). \quad (5)$$

This equation confirms our previous intuition that no matter what ordering we select, the *total* amount of uncertainty is

still the same.[4]

Our full *static ordering* algorithm is presented in Algorithm 1. We select the entire question ordering in a greedy manner, starting with the first question. At the ith step, we choose the question with the highest conditional entropy, given the questions that have already been selected. We call this ordering static because the algorithm is run offline, based only on the learned Bayesian network, and does not change during actual data entry.

In many scenarios the form designer may also want to specify natural groupings of questions that should be presented to the user as one section. Our model can be easily adapted to handle this constraint, by maximizing entropy between specified groups of questions. We can select these groups according to joint entropy:

$$\arg\max_{\mathbf{G}} H(\mathbf{G} \mid F_1, \ldots, F_{i-1}), \tag{6}$$

where $\mathbf{G}$ is over the form designers' specified groups of questions. We can then further apply the static ordering algorithm to order questions *within* each individual section. In this way, we capture the highest possible amount of uncertainty while still conforming to ordering constraints imposed by the form designer.

5.1 Reordering Questions during Data Entry

Thanks to the interactive nature of USHER forms, we can take our ordering notion a step further, and dynamically *reorder* questions in a form as they are entered. This approach can be appropriate for scenarios when data enterers input one value at a time, such as on small mobile devices. We can apply the same greedy selection criterion as in Algorithm 1, but update the calculations with the actual responses to previous questions. Assuming questions $\mathbf{G} = \{F_1, \ldots, F_i\}$ have already been filled in with values $\mathbf{g} = \{f_1, \ldots, f_n\}$, the next question is selected by maximizing:

$$\begin{aligned} &H(F_i \mid \mathbf{G} = \mathbf{g}) \\ &= -\sum_{f_i} P(F_i = f_i \mid \mathbf{G} = \mathbf{g}) \log P(F_i = f_i \mid \mathbf{G} = \mathbf{g}). \end{aligned} \tag{7}$$

Notice that this objective is the same as equation 4, except using the actual responses entered into previous questions, rather than taking a weighted average over all possible values. Constraints specified by the form designer, such as topical grouping, can also be respected in the dynamic framework by restricting the selection of next questions at every step.

In general, dynamic reordering can be particularly useful in scenarios where the input of one value determines the value of another. For example, in a form with questions for *gender* and *pregnant*, a response of *male* for the former dictates the value of the latter. However, dynamic reordering presents a drawback in that it may confuse data enterers who routinely enter information into the same form, and have come to expect a specific question order. Determining the tradeoff between these opposing concerns is an important direction of future work.

6. QUESTION RE-ASKING

[4]Writing out the sum of entropies using equation 5 yields a telescoping sum that reduces to the fixed value $H(\mathbf{F})$.

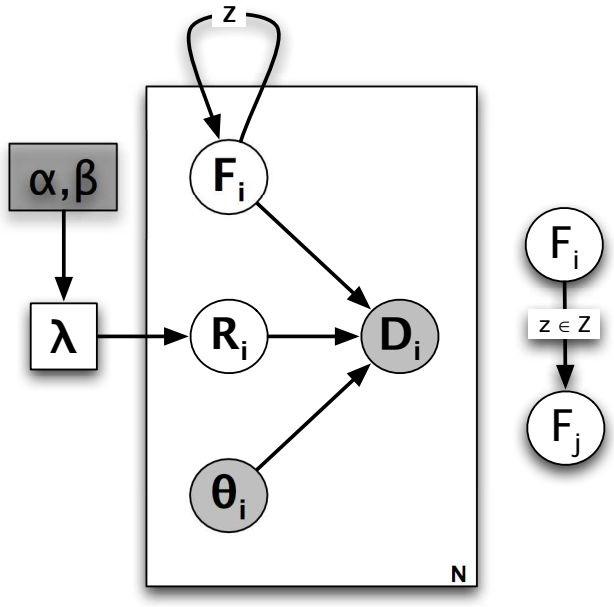

Figure 5: A graphical model with explicit error modeling. Here, D_i represents the actual input provided by the data enterer for the ith question, and F_i is the correct unobserved value of that question. F variables can be connected by edges $z \in Z$, representing the relationships discovered in the structure learning process. θ_i represents the "error" distribution, which in our current model is uniform over all possible values. R_i is a hidden binary random variable specifying whether the entered data was erroneous; its probability λ_i is drawn from a Beta prior with fixed hyperparameters α and β.

After a form instance is entered, the probabilistic model is again applied for the purpose of identifying *errors* made during entry. Because this determination is made immediately after form submission, USHER can choose to *re-ask* questions for which there may be an error. By focusing the re-asking effort only on questions that were likely to be mis-entered, USHER is likely to catch mistakes at a small incremental cost to the data enterer.

USHER estimates the probability that an error was made for each question response. The intuition behind this determination is straightforward: questions whose responses are "unexpected," with respect to the rest of the input responses, are more likely to be incorrect. To formally incorporate this notion in our stochastic approach, we augment the Bayesian network from Section 4 with additional nodes capturing a probabilistic view of entry error. Under this new representation, the ith question is represented with the following set of random variables:

- F_i: the *correct* value for the question, which is unknown to the system, and thus a *hidden* variable.
- D_i: the question response provided by the data enterer, an *observed* variable.
- θ_i: the probability distribution of values that are entered as mistakes, which is a single fixed distribution per question. We call θ_i the *error distribution*.

- R_i: a binary variable specifying whether an error was made in this question.

Additionally, we introduce a random variable λ shared across all questions, specifying how likely errors are to occur for a typical question of that form submission. We call the Bayesian network augmented with these additional random variables the *error model*.

As before, the F_i random variables are connected according to the learned structure explained in Section 4. Within an individual question, the relationships between the newly introduced variables is shown in Figure 5. Node $R_i \in \{0, 1\}$ is a hidden indicator variable specifying whether an error will happen at this question. Our model posits that a data enterer implicitly flips a coin for R_i when entering a response for question i, with probability of one equal to λ. If $R_i = 0$, no error occurs and the data enterer inputs the correct value for D_i, and thus $F_i = D_i$. However, if $R_i = 1$, then the data enterer makes a mistake, and instead chooses a response for the question from a fixed distribution θ_i. In our present implementation θ_i is a uniform distribution over all possible values for question i.[5]

Formally, the conditional probability distribution of each random variable is defined as follows. $P(F_i \mid \ldots)$ is still defined as in Section 4.

$$D_i \mid F_i, \theta_i, R_i \sim \begin{cases} \text{PointMass}(F_i) & \text{if } R_i = 0, \\ \text{Discrete}(\theta_i) & \text{otherwise,} \end{cases} \quad (8)$$

All of D_i's probability is concentrated around F_i (i.e., a point mass at F_i) if R_i is zero; otherwise its probability distribution is the error distribution.

$$R_i \mid \lambda \sim \text{Bernoulli}(\lambda) \quad (9)$$

Conditioned only on its parent, the probability of making a mistake in an arbitrary question is the value λ.

$$\lambda \sim \text{Beta}(\alpha, \beta) \quad (10)$$

The probability of mistake λ is itself an unknown random variable, so we model its flexibility by defining it as a *Beta* distribution, which is a continuous distribution over the real numbers from zero to one. The Beta distribution itself takes two parameters α and β, which we set to fixed constants. The use of a Beta prior distribution for a Bernoulli random variable is standard practice in Bayesian modeling, partially because this combination is mathematically convenient [4].

The ultimate variable of interest in the error model is R_i: we wish to induce the probability of making an error for each question, given the actual question responses:

$$P(R_i \mid D_1, \ldots, D_n). \quad (11)$$

Again, because of our Bayesian network formulation, we can use standard Bayesian inference procedures to compute this probability. In our implementation, we use the Infer.NET toolkit [18] with the Expectation Propagation algorithm [19] for this estimation.

Once we have inferred a probability of error for each question, actually performing the re-asking is a simple matter. Questions whose error probability estimates exceed a threshold value, up to a customizable limit, are presented to the data enterer for re-entry; if the new value does not match the previous value, the question is flagged for further manual reconciliation, as in double entry.

[5]The error distribution θ_i could itself be estimated from prior logs about frequently made errors for each question. If we have such data, this would automatically train our system to be especially wary of commonly made mistakes.

7. EVALUATION

We evaluated the benefits of USHER by simulating several data entry scenarios to show how our system can improve data quality. In this section, we first present our experimental data sets, and then our simulation experiments and their results.

7.1 Data Sets and Model Setup

We examine the benefits of USHER's design using two data sets. The *survey* data set comprises responses from a 1986 about race and politics in the San Francisco-Oakland metropolitan area [3]. The UC Berkeley Survey Research Center interviewed 1,113 persons by random-digit telephone dialing. The *patient* data set was collected from anonymized vaccination and visit records at a rural HIV/AIDS clinic in Tanzania, previously described in Section 3. Our goal was to focus on form questions that can benefit from data cleaning. As a result, in the patient data set, we dropped from the original forms any questions resulting in free-text values (e.g., comment boxes). In total we had fifteen questions for the survey and nine for the patient data. We also discretized continuous values using fixed-length intervals, and treated the absence of a response to a question as a separate value to be predicted.

For both data sets, we randomly divided the available prior submissions into *training* and *test* sets. We performed structure learning and parameter estimation using the training set. The test set was then used for the data entry scenarios presented below. For the survey, we had 891 training instances and 222 test; for patients, 1,320 training and 330 test.

We performed structure learning for both data sets as described in Section 4, resulting in the graphical models shown in Figures 2 and 1. Our parameter estimates were smoothed by mixing each probability table with 0.1 times the uniform distribution. USHER computes the static orderings for each data set ahead of time, based directly on this graphical model.

7.2 Simulation Experiments

In our simulation experiments, we aim to verify hypotheses regarding two components of our system: first, that our data-driven question orderings ask the most uncertain questions first, improving our ability to predict missing responses; and second, that our re-asking model is able to identify erroneous responses accurately, so that we can target those questions for verification.

7.2.1 Ordering

For the ordering experiment, we posit a scenario where the data enterer is interrupted while entering a form submission, and thus is not able to complete the entire instance. Our goal is to measure how well we can predict those remaining questions, under four different question orderings: USHER's precomputed static ordering, USHER's dynamic ordering (where the order can adjust in response to individual question responses), the original form designer's ordering, and a random ordering. In each case, predictions are made by computing the maximum position of the probabil-

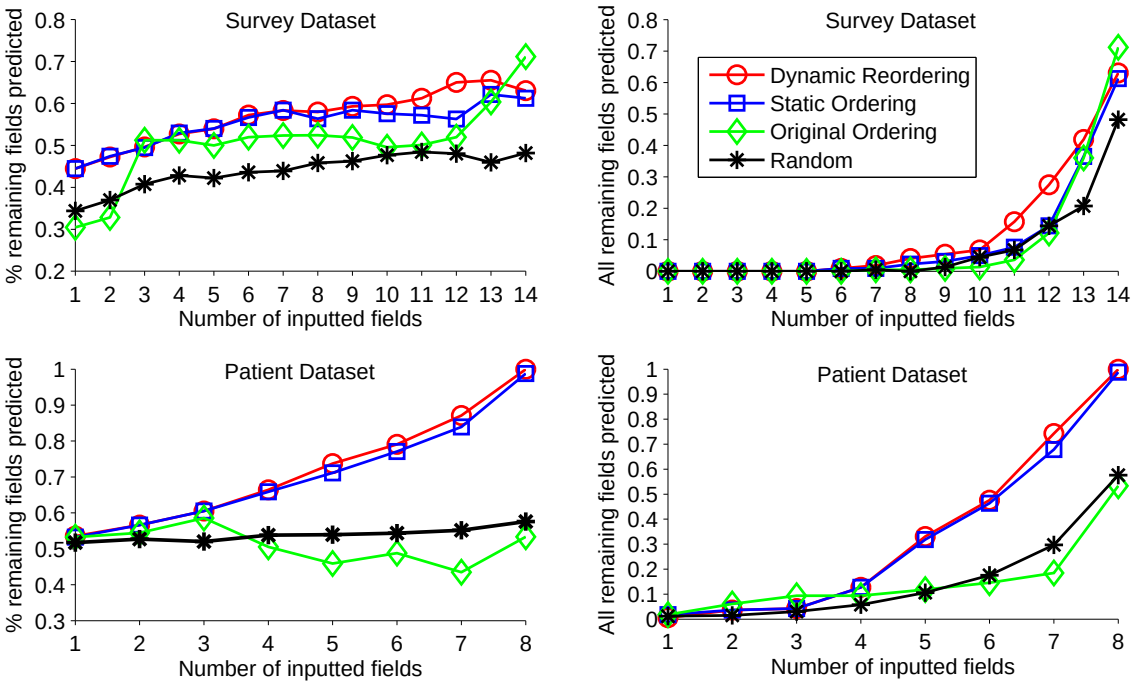

Figure 6: Results of the ordering simulation experiment. In each case, the x-axis measures how many questions are filled before the submission is truncated. In the figures at the left, the y-axis plots the average proportion of remaining question whose responses are predicted correctly. In the figures at the right, the y-axis plots the proportion of form instances for which *all* remaining questions are predicted correctly. Results for the survey data are shown at top, and for the HIV/AIDS data at bottom.

ity distribution over unentered questions, given the known responses.[6] Results are averaged over each instance in the test set.

The left-hand graphs of Figure 6 measures the average number of correctly predicted unfilled questions, as a function of how many responses the data enterer did enter before being interrupted. In each case, the USHER orderings are able to predict question responses with greater accuracy than both the original form ordering and a random ordering for most truncation points. Similar relative performance is exhibited when we measure the percentage of test set instances where *all* unfilled questions are predicted correctly, as shown in the right side of Figure 6.

The original form orderings tend to underperform their USHER counterparts; human form designers typically do not optimize for asking the most difficult questions first, instead often focusing on boilerplate material at the beginning of a form. Such design methodology is detrimental for automatic filling in of missing values for incomplete form submissions.

As expected, between the two USHER approaches the dynamic ordering yields slightly greater predictive power than the static ordering. Because the dynamic approach is able to adapt the form to the data being entered, it can focus its question selection on high-uncertainty questions specific to the current form instance. In contrast, the static approach effectively averages over all possible uncertainty paths.

[6]In machine learning parlance, this is the mode of the posterior marginal distribution over the unknown questions conditioned on the known questions.

7.2.2 *Re-asking*

For the re-asking experiment, our hypothetical scenario is one where the data enterer enters a complete form instance, but possibly with erroneous values for some question responses. Specifically, we assume that for each data value, the enterer has some fixed chance p of making a mistake. When a mistake occurs, we assume that an erroneous value is chosen uniformly at random. Once the entire instance is entered, we feed the entered values to our error model, and compute the probability of error for each question. We then re-ask the questions with the highest error probabilities, and measure whether we chose to re-ask the questions that were actually wrong. Results are averaged over 10 random trials for each test set instance.

Figure 7 plots the percentage of instances where we choose to re-ask all of the erroneous questions, as a function of the number of questions that are re-asked, for error probabilities of 0.05, 0.1, and 0.2. In each case, our error model is able to make significantly better choices about which questions to re-ask than a random baseline. In fact, for $p = 0.05$, which is a representative error rate that we observe in the field, USHER successfully re-asks all errors over 80% of the time within the first three questions in both data sets. We observe that the traditional approach of double entry corresponds to re-asking every question; under reasonable assumptions about the occurrence of errors, our model is often able to achieve the same result as double entry (of identify-

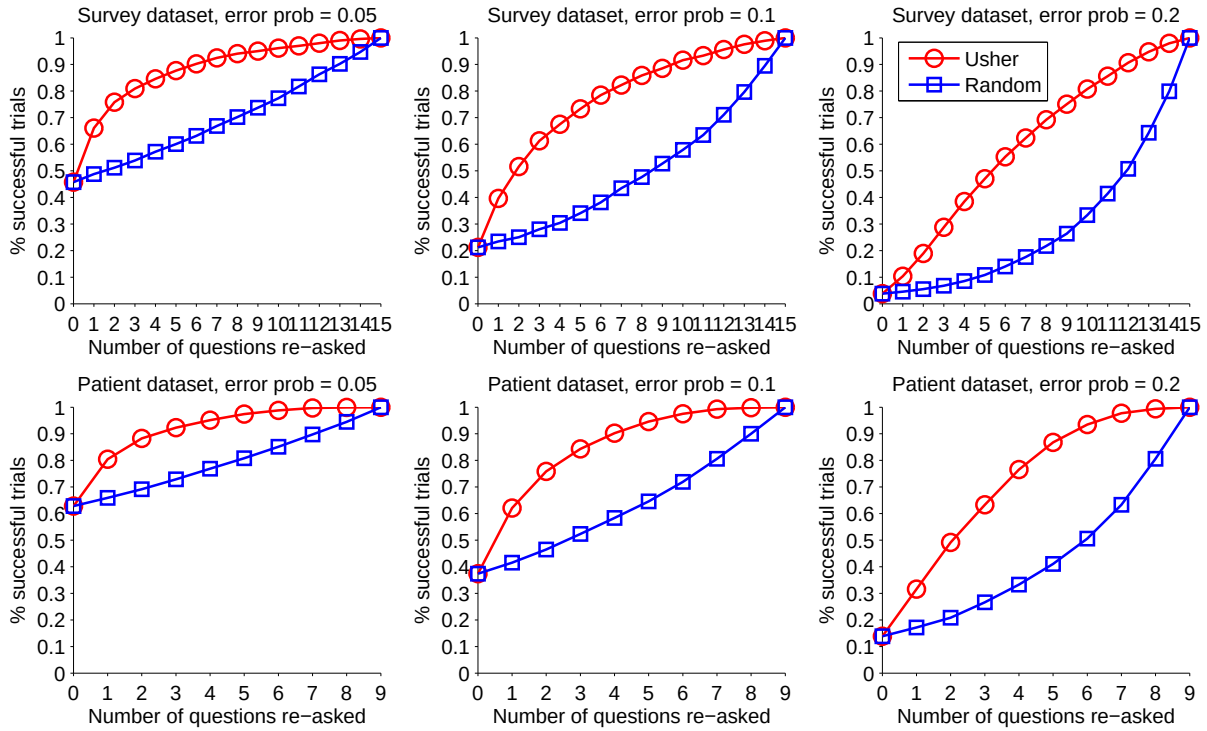

Figure 7: Results of the re-asking simulation experiment. In each case, the x-axis measures how many questions we are allowed to re-ask, and the y-axis measures whether we correctly identify all erroneous questions within that number of re-asks. The error probability indicates the rate at which we simulate errors in the original data. Results for the survey data are shown at top, and for the HIV/AIDS data at bottom.

ing all erroneous responses) at a substantially reduced cost, in terms of number of questions asked.

8. DYNAMIC DATA ENTRY FEEDBACK

In the previous sections, we showed how USHER's probabilistic model enables automatic form layout ordering and optimized answer confirmation. In this section, we describe additional opportunities to provide real-time data entry feedback to a user. During form entry, USHER's probabilistic machinery provides conditional probabilities, conditional entropies, and error probabilities. With these tools, we can provide feedback in several new ways.

First, we can use conditional probabilities to *assess* a newly entered value. A dropdown list widget can change color based on the likelihood of the selected value. This type of dynamic feedback is not intrusive, and importantly, the mechanism is *non-biasing*. Feedback is only shown to the user after an answer is chosen.

Second, we can use conditional expectations to gently *nudge* users towards expected values. We introduce the idea of interface "friction" – the difficulty of entering a value – as the inverse of expectation, and posit that low-expectation values should have higher friction to access. Take for example a dropdown list in which the answers are ordered from most to least likely; the extra time to scroll to unlikely answers captures their friction. Similarly, a text box with "type-ahead" has less friction for expected values (which get auto-completed) than for values with low expectation (which need to be typed in full).

Third, we can directly communicate answer probabilities to the user, by graphically displaying probabilities in a quantitative way. No longer a subconscious exercise, this approach *guides* the user toward expected results. Consider a set of radio buttons, decorated with percentile values next to each choice. In this scenario, the user is assisted (and arguably biased) by the interface to enter high likelihood answers.

The notion of expectation in all these cases should be *conditioned* on earlier answers provided by the user. Conditional probabilities can be surprising if they differ significantly from unconditioned versions. For example, the likelihood of entering "milk" as a favorite beverage may be small in the general population, but a high-probability answer if an earlier question about age was answered "< 5". To account for this, it is important that the data enterer know where they are and where they came from in the probability space of their answers. This can be addressed in a simple fashion by providing a "path" visualization (similar to that of a website traversal) showing answers to previous questions that led up to the current question. The visualization serves as a quantitative breadcrumb trail that communicates the historical and contextual impact of answering the current question. It can help users understand conditional probabilities and the context of widget feedback.

We have prototyped many of these widget modifications in a popular Flash-based widget toolkit, as shown in Figure 8.

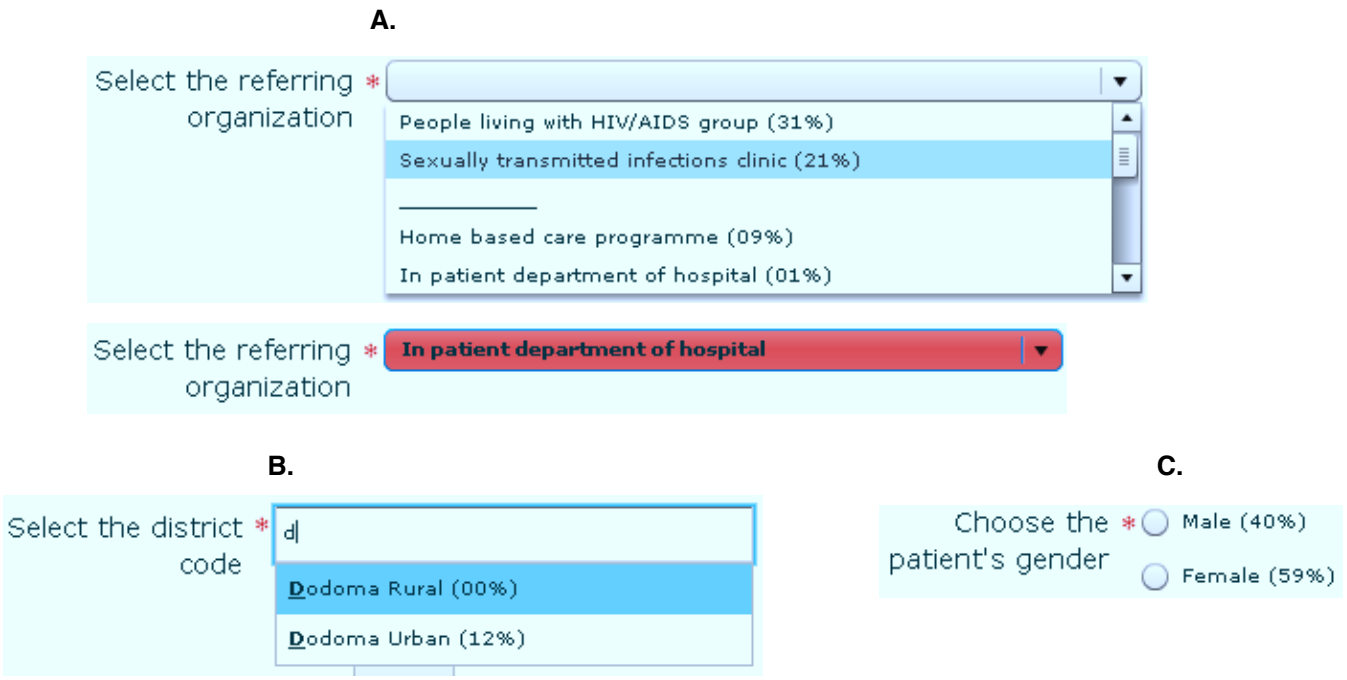

Figure 8: Dynamic data entry widgets. A. A drop down widget that assesses user input with color feedback. B. A type ahead widget showing matches ranked by likelihood. C. A radio button widget showing probabilities with each choice.

A robust evaluation of these human interface approaches is a research undertaking in its own right, requiring user studies to be performed in a number of contexts: controlled online surveys, as well as field studies in diverse settings from well-funded call center applications to resource-constrained organizations in the developing world. We have a number of these efforts in progress.

9. SUMMARY AND FUTURE WORK

In this paper, we have shown that completely automated probabilistic approaches can be used to design intelligent data entry forms that promote high data quality. Our USHER system encompasses every step of the data entry process. Before entry, we find an ordering of form entry widgets that promotes rapid data capture. During entry, we adapt the form ordering based on entered values. After entry, we automatically identify possibly erroneous inputs, and re-ask those questions to verify their correctness. Our empirical evaluations demonstrate the data quality benefits of each of these USHER components: question ordering allows for effective recovery of truncated form submissions, and the re-asking model identifies erroneous responses with high precision.

There are a variety of ways in which this work can be extended. A major piece of future work alluded to in Section 8 is to study how our probabilistic model can inform intelligent adaptations of the user interface during data entry. We intend to answer this problem in greater depth with user studies and field deployments of our system. On the modeling side, our present probabilistic approach assumes that every question is discrete and takes on a series of unrelated values. Relaxing these assumptions would make for a richer and potentially more accurate predictive model for many domains. Additionally, we may want to consider models that reflect temporal changes in the underlying data. Our present error model makes strong assumptions both about how errors are distributed, and what errors look like. Recording common data entry errors, and adapting our system to catch those errors, is an interesting line of future work.

Finally, we plan to measure the practical impact of our system, by deploying USHER with our field partners, including HIV/AIDS treatment clinics in Tanzania and Malawi, and migrant farmworker service providers in Northern California. These organizations' data quality concerns were the original motivation for this work, and thus serve as the ultimate litmus test for our system. Our hope is to make a positive societal impact through this technology.

10. ACKNOWLEDGMENTS

The authors thank Yaw Anokwa, Michael Bernstein, S.R.K. Branavan, Tyson Condie, Heather Dolan, Max van Kleek, Jamie Lockwood, Bob McCarthy, Tom Piazza, and Christine Robson for their helpful comments and suggestions.

11. REFERENCES

[1] A. Ali and C. Meek. Predictive models of form filling. Technical Report MSR-TR-2009-1, Microsoft Research, Jan. 2009.

[2] C. Batini and M. Scannapieco. *Data Quality: Concepts, Methodologies and Techniques*. Springer, 2006.

[3] U. C. Berkeley. Survey documentation and analysis. `http://sda.berkeley.edu`.

[4] J. M. Bernardo and A. F. Smith. *Bayesian Theory*. Wiley Series in Probability and Statistics, 2000.

[5] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2007.

[6] J. V. D. Broeck, M. Mackay, N. Mpontshane, A. K. K. Luabeya, M. Chhagan, and M. L. Bennish. Maintaining data integrity in a rural clinical trial. *Controlled Clinical Trials*, 2007.

[7] F. G. Cozman. Javabayes - bayesian networks in java. `http://www.cs.cmu.edu/~javabayes`.

[8] T. Dasu and T. Johnson. *Exploratory Data Mining and Data Cleaning*. Wiley Series in Probability and Statistics, 2003.

[9] S. Day, P. Fayers, and D. Harvey. Double data entry: what value, what price? *Controlled Clinical Trials*, 1998.

[10] K. Z. Gajos, D. S. Weld, and J. O. Wobbrock. Decision-theoretic user interface generation. In *AAAI'08*, pages 1532–1536. AAAI Press, 2008.

[11] R. M. Graves, F. J. Fowler, M. P. Couper, J. M. Lepkowski, E. Singer, and R. Tourangeau. *Survey Methodology*. Wiley-Interscience, 2004.

[12] A. Hartemink. Banjo: Bayesian network inference with java objects. `http://www.cs.duke.edu/~amink/software/banjo`.

[13] D. Heckerman, D. Geiger, and D. M. Chickering. Learning bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, 20(3):197–243, 1995.

[14] J. M. Hellerstein. Quantitative data cleaning for large databases. United Nations Economic Commission for Europe (UNECE), 2008.

[15] F. Jelinek and R. L. Mercer. Interpolated estimation of markov source parameters from sparse data. In *Proceedings of the Workshop on Pattern Recognition in Practice*, 1980.

[16] D. W. King and R. Lashley. A quantifiable alternative to double data entry. *Controlled Clinical Trials*, 2000.

[17] K. Kleinman. Adaptive double data entry: a probabilistic tool for choosing which forms to reenter. *Controlled Clinical Trials*, 2001.

[18] C. Microsoft Research. Infer.net. `http://research.microsoft.com/en-us/um/cambridge/projects/infernet`.

[19] T. P. Minka. Expectation propagation for approximate bayesian inference. In *Proceedings of the Conference in Uncertainty in Artificial Intelligence*, 2001.

[20] K. L. Norman. Online survey design guide. `http://lap.umd.edu/survey\_design`.

[21] B. M. Tom Piazza. Personal interview, 2009.

[22] Y. Yu, J. A. Stamberger, A. Manoharan, and A. Paepcke. Ecopod: a mobile tool for community based biodiversity collection building. In *JCDL*, 2006.