# Designing Adaptive Feedback for Improving Data Entry Accuracy

*Kuang Chen, Joseph M. Hellerstein*
Dept. of Electrical Engineering and Computer Science
University of California, Berkeley
387 Soda Hall, Berkeley, CA 94720-1776 USA
{kuangc, hellerstein}@cs.berkeley.edu

*Tapan S. Parikh*
School of Information
University of California, Berkeley
102 South Hall, Berkeley, CA 94720-4600 USA
parikh@ischool.berkeley.edu

## ABSTRACT
Data quality is critical for many information-intensive applications. One of the best opportunities to improve data quality is during entry. USHER provides a theoretical, data-driven foundation for improving data quality during entry. Based on prior data, USHER learns a probabilistic model of the dependencies between form questions and values. Using this information, USHER maximizes *information gain*. By asking the most unpredictable questions first, USHER is better able to predict answers for the remaining questions. In this paper, we use USHER's predictive ability to design a number of intelligent user interface adaptations that improve data entry accuracy and efficiency. Based on an underlying cognitive model of data entry, we apply these modifications before, during and after committing an answer. We evaluated these mechanisms with professional data entry clerks working with real patient data from six clinics in rural Uganda. The results show that our adaptations has the potential to reduce error (by up to 78%), with limited effect on entry time (varying between -14% and +6%). We believe this approach has wide applicability for improving the quality and availability of data, which is increasingly important for decision-making and resource allocation.

**ACM Classification:** H5.2 [Information interfaces and presentation]: User Interfaces - Graphical user interfaces.

**General terms:** Design, Experimentation, Human Factors

**Keywords:** Data quality, data entry, form design, adaptive interface, repetitive task

## INTRODUCTION
In today's world, says Carl Malamud, "information is a form of infrastructure; no less important to our modern life than our roads, electrical grid or water systems" [7]. As such, we should be as concerned about the *quality* of our information, as we are about our roads, water and power. Data quality is especially important for critical applications like health care, where even a single error can have drastic consequences. One of the best opportunities to improve data quality is during entry. Data entry is ubiquitous — organizations all over the world rely on clerks to transcribe information from paper forms into supposedly authoritative databases. However, many smaller organizations, particularly those operating in the developing world, struggle with maintaining high quality during transcription. Part of the reason is because they lack expertise in form design, failing to correctly specify field constraints and other validation logic. They also lack the resources needed for performing double entry — the standard practice of entering data twice to validate the results; or even for doing post hoc data cleaning. For low-resource organizations, data entry is the first and best opportunity to address data quality.

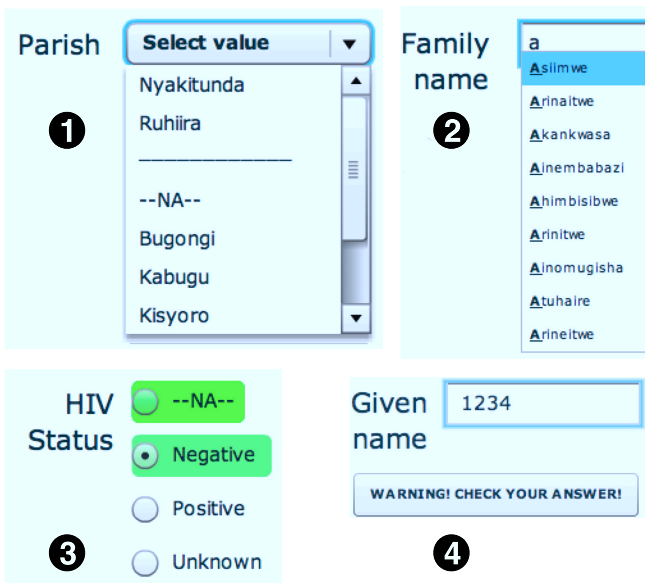

Figure 1: (1) drop down split-menu promotes the most likely items (2) text field ranks autocomplete suggestions by likelihood (3) radio buttons highlights promote most likely labels (4) warning message appears when an answer is a multivariate outlier.

USHER provides a theoretical, data-driven foundation for improving data quality during entry [10]. Based on prior data, USHER learns a probabilistic model of the dependencies between form questions and values. Using this information, USHER reorders questions to maximize *information gain*. By asking the most unpredictable questions first, USHER is better able to predict answers for the remaining questions.

The theoretical basis for USHER's multivariate predictive model is discussed in [10], along with a set of simulation results demonstrating its accuracy on two sample data sets. In this paper, we use USHER's predictive ability to design a number of intelligent user interface adaptations that can directly improve data entry accuracy and efficiency. We evaluated each of these mechanisms with professional data entry clerks working with real patient data from six clinics in rural Uganda. The results show that our adaptations have the potential to improve entry accuracy: for radio buttons, the error rates decreased by 54-78%; for other widget types, error rates fell in a pattern of improvement, but were not statistically significant. The impact of our adaptations on entry cost (time) varied between -14% to +6%.

The specific adaptations we tested include: 1) setting defaults corresponding to highly likely answers, 2) dynamically reordering and highlighting other likely options, and 3) providing automatic warnings when the user has entered an unlikely value. The first technique changes an entry task to a confirmation task, which we show has the potential to significantly improve accuracy and efficiency. The second approach *guides* the user towards more likely values, and away from unlikely ones, which we show further improves accuracy. Finally, by warning the user about particularly unlikely values, we approximate double entry at a fraction of the cost.

The rest of this paper is organized as follows. In the next section, we discuss related work. Next, we introduce USHER — the probabilistic foundation of this work. In the fourth section, we provide a cognitive model of data entry based on our contextual inquiry, and in the fifth section we use that to motivate a number of specific intelligent user interface adaptations. In the sixth section we present the experimental setup for our evaluation, and in the seventh section we describe the results. Lastly, we conclude and discuss future work.

## RELATED WORK

In this section we summarize the major areas of related work: managing data quality, improving data entry efficiency, and designing dynamic, adaptive user interface widgets.

### Managing Data Quality

Best practices for form design include specifying pre-determined constraints to reject or warn the user when they enter illegal or unlikely values [32]. More sophisticated survey design techniques involve inserting additional *cross-validation* questions to double-check the accuracy of important fields [17]. For example, adding a "birth-year" question to cross-validate "age".

A standard practice in electronic form design is to set binary constraints that accept or reject an answer. Consider the example of using red highlighting to denote that an invalid answer has been rejected, post entry. In essence, the entry task is parameterized with 0% likelihood for the invalid value, a special case of our approach. With finer-grained probabilities during entry, we generalize the practice of setting constraints to one of apportioning *friction* [18] in proportion with answer likelihood.

Multivariate outlier detection is used for post hoc data cleaning, where data is "cleaned" after it resides in a database [18]. Our approach is similar, except we weed outliers using adaptive feedback mechanisms *during* entry, when it is often still possible to correct the error directly.

### Improving Data Entry Efficiency

Cockburn *et al.* modeled the performance of menu interfaces as control mechanisms [11]. They offer insight on the transition from novice to expert behavior. We focus on a wider class of input interfaces for expert entry.

There have been several efforts to improve data input with modeling and interface adaptation [8, 10, 19, 23, 30, 33, 34, 35, 37]. Most of these have only provided simulation results of predictive accuracy and improvements in efficiency.

Ali and Meek discuss a variety of intelligent approaches to suggest auto-complete values in text fields for web forms to improve the speed of entry [8]. Yu *et al*. generated type-ahead suggestions for collecting conservation data on mobile devices based on the location of the user [37]. Hermens and Schlimmer set default values in form fields using decision trees [19]. Warren *et al* worked with medical doctors entering diagnosis information into an electronic medical record [33, 34, 35]. They trained models on drugs and diagnoses to automatically populate a "hotlist" of potential drug choices for the provider. Their models rely only on bivariate relationships between a drug and a single diagnosis.

All these works were primarily concerned with improving data entry *efficiency*. While we have adapted several of the specific widgets they have described (setting defaults and ranking auto-complete suggestions), our primary goal is demonstrating improvement in bottom-line accuracy.

### Adaptive User Interfaces

The literature on adaptive user interfaces discusses feedback mechanisms' behavioral predictability, cognitive complexity, cost of being wrong, predictive accuracy, and the users ability to opt-out of an adaptation [16]. These guidelines were useful for framing our system. We also explored some specific adaptive widget types discussed in this literature, namely split or ephemeral selection boxes, and enlarging the clickable area for more likely options [14, 27, 31, 36].

## A DATA-DRIVEN FOUNDATION FOR ADAPTIVE FORMS

USHER is a data-driven foundation for adapting data entry forms. The theoretical basis of USHER's predictive model was discussed in [10]. Here, we summarize the aspects relevant to this paper.

### A Model for Any Form

The goal of an USHER model is simple: given a subset of answers for a form, accurately predict values for the unanswered questions. As shown in Figure 2, the model learns from previously entered form instances to improve entry in the future. The model itself is a Bayesian network over a form that captures the relationships between form questions

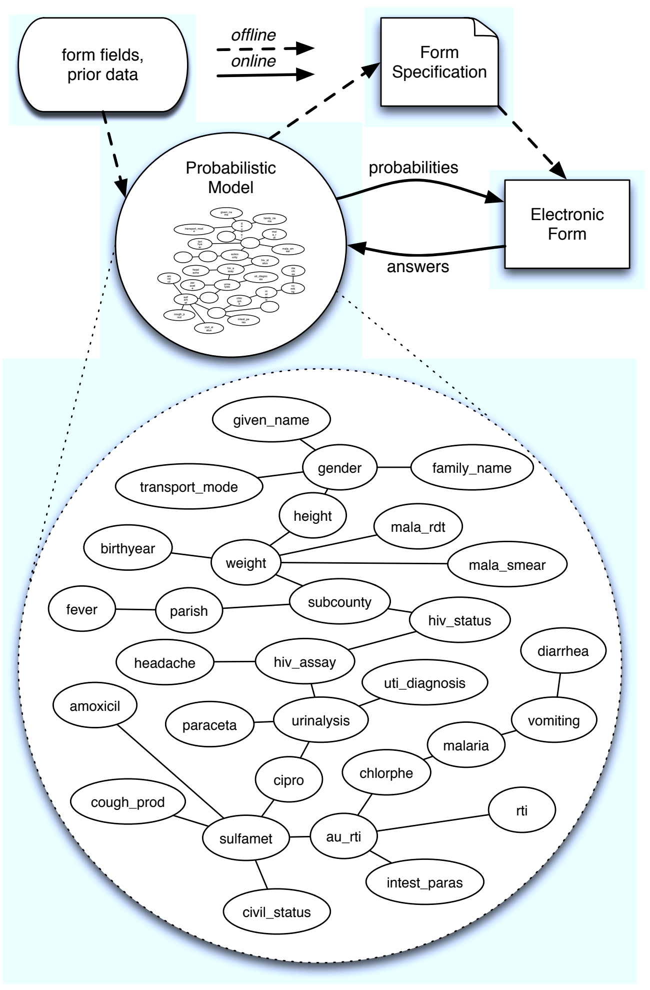

Figure 2: USHER components, data flow and probabilistic model: Above are USHER system components and data artifacts, with arrows showing data flow; zoom circle shows the Bayesian network learned from data.

based on prior data. Learning these relationships for an arbitrary form is the first step for building this model. The naive approach would assume compete dependence of each field on all other fields, but such an approach could lead to both poor predictions and slow queries. Instead, USHER learns relationships using a standard machine learning technique called structure learning [21]. Form designers can also specify *a priori* relationships and constraints to be included in the model. In Figure 2, we can see the Bayesian network generated for the data set that we used for the evaluation described later in this paper.

USHER estimates the parameters of the network by calculating, for each field, a conditional probability table, which holds the proportion of each possible answer value, given each possible assignment of parent values. To prevent the case of insufficient data causing zero probabilities and overfitting, parameter estimates are smoothed. Using this model, we infer a probability distribution for unanswered form questions given answered ones. More specifically, USHER calculates marginal distributions over sets of random variables conditioned on entered values using the Junction Tree Algorithm [9].

### Model Accuracy and Question Ordering

Improving the predictive accuracy of the model allows for more effective feedback mechanisms. During data entry, the order of questions in a form will greatly influence the model's predictive accuracy. We give an intuitive analogy: in the child's game *twenty-questions*, a *responder* thinks of a concept, like "airplane", and an *asker* can ask 20 yes-no questions to identify the concept. Considering the space of all possible questions to ask, the asker's success is vitally related to question selection and ordering — the asker wants to choose a question that cuts through the space of possibilities the most quickly. For example, the canonical opening question, "is it bigger than a breadbox?" has a high level of expected information gain — measured as information entropy.

For forms, USHER can calculate the entropy for each field and automatically create a question ordering, within user-specified constraints, that gains information as quickly as possible. This will allow USHER to provide better predictions for later questions. Again, we allow human form designers to specify any required ordering or grouping constraints. From a user perspective, asking questions with high information gain first will be more likely to avoid the typical pitfalls of repetitive work, where the user's interest is known to wane over time [22].

USHER provides two different algorithms for question ordering: *static* and *dynamic*. The *dynamic* ordering picks the next best question at runtime, by conditioning the model on *actual* observed answers. This approach is suitable for direct electronic data entry, for example when mobile devices are used for field data collection [28]. In contrast, the *static* ordering is based on expected conditional entropies before entry and is therefore calculated offline. An organization that relies on paper forms that are later transcribed would have a difficult time arbitrarily re-ordering fields, as data entry clerks rely on the strict correspondence between the paper and digital versions. In this case, they can periodically re-order the questions on the paper and digital versions of the form based on the static ordering. It is also important to note that reordering fields is entirely optional. An USHER model can provide predictions given *any* form ordering.

## OPPORTUNITIES FOR ADAPTATION

Figure 3 provides a simplified representation of the major physical and/or mental tasks involved in data entry. This cognitive model was derived based on our own observation of professional data entry clerks. It also assumes that the user is transcribing data from paper into an electronic equivalent.

- First, the user *acquires question nature* by looking at the screen for the question number, text, data type, widget type, etc.
- Then, the user searches for this question on the paper form to *acquire the answer from the source* and memorize it.
- Next, the user tries to *locate the answer on screen*. This may require a visual scan (for radio button widgets), scrolling and scanning (for drop down menus), or a set of individual

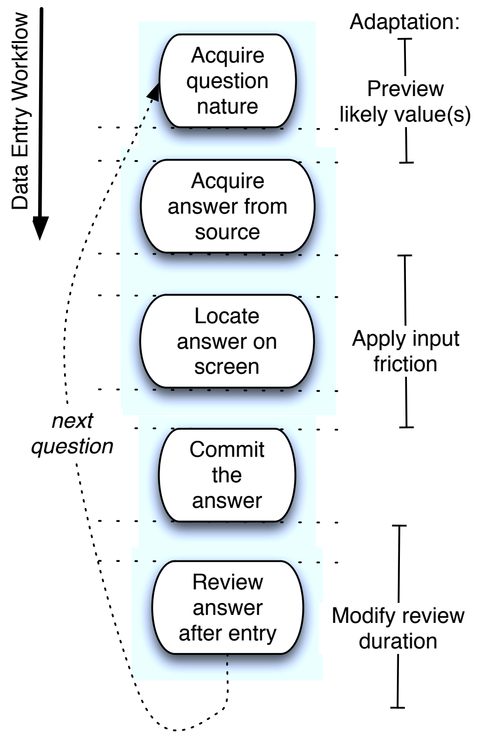

Figure 3: A cognitive task model for data entry. The vertical spans, on the right, show opportunities for dynamic adaptation.

keystrokes, each separated by a visual scan (for text fields with autocomplete).

- The user *commits the answer* after finding it, typically with a single physical operation, such as clicking on the "Next" button.
- The user may also *review answer after commit*.

Each of these stages creates its own opportunities to improve data entry accuracy and efficiency. In this paper, we discuss how intelligent interface adaptation can:

1. Before entry, allow the user to preview likely answers, or to convert an entry task to a *confirmation* task.
2. During entry, help the user locate the correct answer on the screen, and reduce the effort required to enter more likely values.
3. After entry, warn the user to review their answer if it is likely to be incorrect.

Before and during entry, we want to *decrease* the time and cognitive effort required for transferring an answer from paper, to reduce the possibility that the answer is lost from short-term memory in that period. In contrast, after entry, we want to *increase* the time and cognitive effort spent to verify the answer's correctness. Toward these goals, we can use USHER's fine-grained probabilities, in each stage of entry, to tune the amount of *friction* in the input interface in proportion with answer likelihood.

## ADAPTIVE FEEDBACK FOR DATA ENTRY

In this section, we propose a specific set of adaptive feedback mechanisms that leverage USHER's predictive capability, and describe how they were iteratively refined through user observation and discussion.

### Design Process

Figure 4: Two of our study participants.

To design appropriate feedback mechanisms, we worked closely over a period of three months with a team of professional data entry clerks working for an international health and research program in Uganda. The purpose of the design phase was two-fold: (1) acclimate users to our electronic forms implementation, so we could ignore any learning effects during the evaluation, and (2) leverage the clerks' experience and knowledge to design a better set of feedback mechanisms.

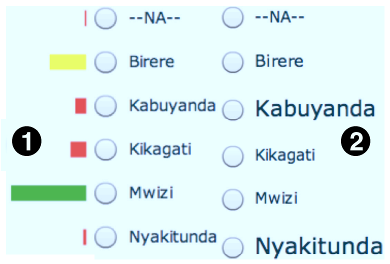

Figure 5: Alternative designs for radio button feedback: (1) radio buttons with bar-chart overlay. (2) radio buttons with scaled labels.

Working together, we made many interesting design observations. We include a few of those here:

- Any feedback we provide should be part of the user's mandatory visual path, by which we mean the locations on the screen where the user *must* look to complete the task. We learned this by experimenting with alternative designs that did not conform to this guideline (on the left side of Figure 5). The more visually complicated design yielded no significant improvement in terms of accuracy. The users explained that because the bar-charts were laid out beside the label, they did not have time to consult them. This is consistent with our observation that it is important to minimize the time between answer acquisition from paper and confirmation in the interface.

- Entry should not depend on semantic understanding of questions and answers. Through conversation with data entry clerks, we concluded that there is no time for this type of processing. The users commented that they see, but do not think about the *meaning* of the answer they are transcribing. This is consistent with the difference between pattern recognition and cognition, where the latter involves potentially expensive access to long-term memory [13].
- The visual layout of the form and individual questions should remain consistent. We experimented with an alternative radio button feedback mechanism that scaled each option in proportion with the value's likelihood, as shown on the right side of Figure 5. The users felt this mechanism was "annoying". This is consistent with prior work that discusses the importance of physical consistency in adaptive graphical user interfaces [16].
- Feedback should not depend on timed onset. Timed interaction techniques like ephemeral adaptation [14] can be effective in other selection contexts. However, for a repetitive, time-sensitive task like data entry, the delay-for-accuracy tradeoff is frustrating.
- Feedback should be accurate. Specifically, when warnings or defaults are triggered too frequently, these mechanisms feel untrustworthy. One user complained about this in our early trials, saying that the "computer is trying to fool me." Indeed, Gajos *et al.* found that user behavior can change based on predictive accuracy [15].

### Feedback Mechanisms
Here we present the adaptive feedback mechanisms that we implemented and tested based on user feedback from early trials. These mechanisms are shown in Figure 1.

- *defaults*: A default is automatically set when the expected likelihood of a value is above a threshold $t$. In our evaluation, we set this threshold to 75% for binary radio buttons.
- *widgets*: We implemented a set of feedback mechanisms for specific widget types.
  - *text autocomplete*: The ordering of autocomplete suggestions are changed from an alphabetical ordering to a likelihood-based ordering. For example, in Figure 1, "Asiimwe" is a popular name in Uganda, and so is ranked first, even though it is not alphabetically first.
  - *drop down*: A split-menu is added to the top of the menu, copying the most likely $k$ answer choices. In Figure 1, the most conditionally likely parishes are in the split-menu.
  - *radio*: $k$ radio button labels are highlighted according to the likelihood of the answers. After trying a few alternative designs (Figure 5), we decided to simply scale the opacity of the highlights according to a $log_2$-scale [26].
- *warnings*: A warning message is shown to the user when the likelihood of their answer is below some threshold $t$. In our evaluation, we set this threshold to 5% for binary choice radio buttons.

### Feedback Parameterization
As mentioned above, we parameterize each feedback mechanism with the question's conditional probability distribution over the answer domain. In addition, *defaults* and *warnings* require a threshold $t$ to determine whether they should be activated, and the *widget* mechanisms require an integer number of potentially highlighted or promoted items $k$. We also need to map each form question to the appropriate widget and feedback type. In this section, we discuss how we set these parameters.

*Mapping to widgets* Mapping of form questions to widgets is driven by the observation that both visual bandwidth and short-term memory are limited to a small number ($7 \pm 2$) of distinct items [25, 26]. When this number is exceeded, the potential for user error increases [24]. We mapped questions to widgets based on domain size: for questions with answer domain that could be shown all at once (domain size $D \leq 2^3$), we decided to use radio buttons; for questions with answer domain that could reasonably fit in a scrolling menu ($D \leq 2^6$), we chose drop down menus; and for questions with any larger answer domain, we decided on autocompleting text fields.

*When to trigger?* We want the triggering threshold $t$ for *defaults* and *warnings* to vary appropriately with the domain size. We formalize this simple notion as follows, with domain size $D$ and a constant $a$:

$$t = a/D, a > 0$$

Observe that when $a = 1$, the threshold $t$ is set to the likelihood of a value in the uniform distribution. For example, we can set $a = 1.5$ for *defaults* so that an answer to a binary question needs $> 75\%$ confidence to trigger setting the default value. Similarly, if we set $a = 0.1$ for *warnings*, a warning will be given when the chosen answer has $< 5\%$ confidence.

*How many values to show?* We verified this notion with a baseline experiment in which the data clerks entered randomly picked dictionary words. We varied the number of options (*feedback positions*) shown in our three widget types, found that *feedback position* $k$ to be strongly related to both error rate ($R^2 > 0.88$)) and entry duration ($R^2 > 0.76$). We use this intuition and specify $k$ as follows:

$$k = min(7, ceiling(log_b(D)), b > 1$$

The constant 7 is the maximum visual channel size; the log base $b$ is a constant; $D$ is the question's answer domain cardinality. For instance, we can set $b = 3$.

Table 1 shows a range of answer cardinalities and resulting number of feedback positions as determined by our parameterization heuristics.

## USER STUDY
To evaluate these adaptive, data-driven feedback mechanisms, we conducted an experimental study measuring improvements in accuracy and efficiency in a real-world data entry environment.

### Context and Participants
To conduct this research, we collaborated with a health facility in a village called Ruhiira, in rural Uganda. Ruhiira is actively supported by the Millennium Villages Project [1] (MVP). MVP conducts multi-pronged interventions in health,

| Domain size | # feedback pos. | Widget |
|---|---|---|
| 2 | 1 | radio button |
| 4 | 2 | radio button |
| 8 | 3 | radio. or drop down |
| 16 | 3 | drop down |
| 32 | 4 | drop down |
| 64 | 5 | drop down or autocomp. |
| 128 | 5 | autocomp. |
| 256 | 6 | autocomp. |
| 512 | 7 | autocomp. |
| $> 1024$ | 7 | autocomp. |

Table 1: Example answer domain cardinalities map to the number of appropriate feedback positions and the appropriate data entry widget.

agriculture, education and infrastructure to reduce poverty in Sub-Saharan Africa. The MVP health team in Ruhiira (also serving six surrounding villages), implemented an OpenMRS [6] electronic medical record system (EMR), but lack the resources and expertise necessary to ensure data quality in EMR data entry. To address these limitations, we are working closely with the health facility management and staff to design both long-term and short-term strategies for improving data quality and use.

The six study participants were professional data entry clerks working at this facility, entering health information on a daily basis. These are the same clerks that we observed to obtain the insights described in prior sections. Prior to the study, each of them became proficient with our electronic forms interface.

### Forms and Data

| Widget type | # questions | Answer domain sizes |
|---|---|---|
| radio button | 21 | 2-5 |
| drop down | 4 | 6-8 |
| autocomplete | 5 | >100 |

Table 2: "Adult Outpatient" form and dataset questions.

Data and forms came directly from the health facility's EMR. For this evaluation, we used a form that is filled out during an adult outpatient visit. We randomly sampled 3388 patient visits to train an USHER model. From the form, we removed the questions that are rarely answered, such as those related to medications that are not actively stocked. About half of the questions we chose described patient demographics and health background, while the rest asked about symptoms, laboratory tests, diagnoses and prescriptions. We mapped the questions to widgets according to answer domain size, specified in Table 1. More details about the data set can be found in Table 2.

### An USHER Model for Patient Visits

The zoom circle in Figure 2 graphically depicts the USHER model resulting from structure learning on this *patient visit* dataset. The edges denote correlations between pairs of variables.

After learning the model's parameters from our dataset, we conducted a simulation experiment to evaluate its predictive accuracy under 4 different question orderings: *static, dynamic, original* and *random*. Our experiment simulated a scenario in which a data entry clerk does not finish entering a form. In Figure 6, the x-axis represents the question position at which entry was interrupted. At each stopping point, we use the model to predict the answers for the rest of the questions. The y-axis shows the resulting accuracy. We see that the *original* ordering does better than *random*, but underperforms USHER's optimized information-theoretic orderings. The evaluation described in the next section is based on a *static* ordering.

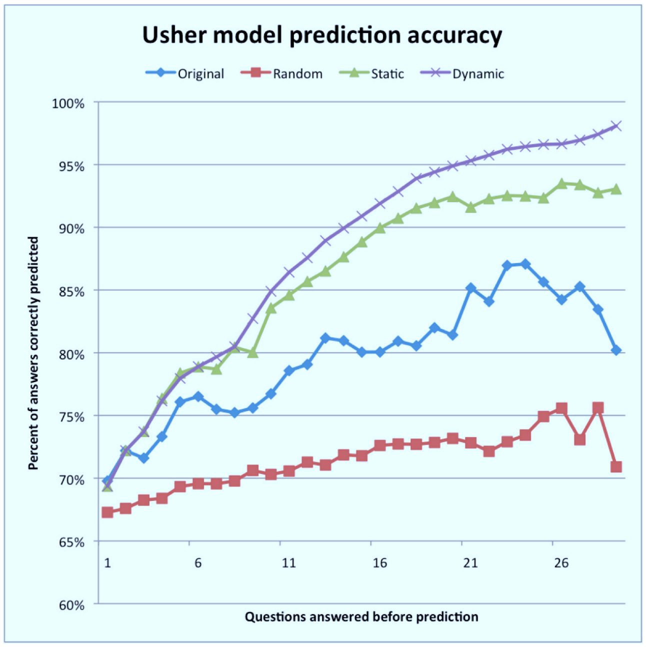

Figure 6: Results of the ordering experiment: x-axis measures the number of questions "entered"; y-axis plots the percentage of remaining answers correctly predicted.

### System

We built a web application using Java Servlets and Adobe Flex (see Figure 2). This included a Java implementation of USHER, running inside a J2EE application server. The USHER model was built offline and summarized in a Bayesian Inference File (BIF) format, instantiated as a singleton upon request from the client. The optimized question ordering was captured in an XML form specification, which included details like question ordering, question labels, data types, widget types and answer domains.

During data entry, the web form interface collected an answer from the user and submitted it to the USHER model. The model then calculated the conditional probability distribution for the next question, resulting in likelihood values for each possible choice. These probabilities were embedded in a XML question fragment that was rendered by the client-side code, which prompted the user to answer the next question. All adaptive feedback mechanisms were implemented within this client-side code.

### Task

A set of 120 *test* form instances were randomly drawn from the EMR. These instances were withheld from the *training* set used to build the model described in the previous section. We printed out these form instances on paper. To more closely mimic a real form, we used a cursive "script" font in the printout for answers that are typically handwritten. The electronic forms were presented to the user as a web table with 120 links. During the study, participants were instructed to click each link, enter the form serial number printed at the top of the form, and to perform data entry as they normally do for each form.

### Procedure

The study was set up as follows: participants sat at a desk and transcribed answers. Participants used their existing data entry client machines: low-power PCs running Windows XP with 256MB of RAM, and a full-sized keyboard and mouse. Each PC had an attached 12" color LCD display. The clients connected via LAN to a "server" that we provided: a dual-core, 2.2 GHz MacBook Pro laptop with 4G of RAM. To mitigate power concerns, we relied on existing solar power and a gasoline generator. Besides this, the working conditions were arguably not too different from that of any cramped office setting.

We conducted the study in 2 sessions, in the morning and afternoon of a single day. All six clerks were available for both entry sessions. In each session, participants entered as many form instances as they could. Ideally, we wanted the data clerks to treat the study like their normal work. As such, we employed an incentive scheme to pay piece-meal (500 USH or 0.25 USD per form) for each form entered. We also constrained the cumulative time allotted to enter all the forms. We felt this most closely matched the clerks' current incentive structure.

Our primary experimental variation was the type of feedback mechanism employed: *defaults*, *widgets*, *warnings* and *plain* (meaning no adaptive feedback). For each form entered, one of these four variants was chosen randomly by the system. For setting default values, we set $t = 1.5/D$. For example, when making a binary choice, if the most likely answer has likelihood $> 0.75$, we set that answer to be the default value. For triggering warnings, we set $t = 0.1/D$. For example, when making a binary choice, if the likelihood is less then $0.05$, we trigger a warning dialog.

For each question that was answered, we collected the following information: answer, correct answer, duration, and feedback provided for each possible answer option. At the end of the experiment, the data clerks each filled out a survey about their experience with the system.

## RESULTS

In total, we collected 14,630 question answers from 408 form instances. The 4 feedback types were randomly assigned across form instances, with the Widgets type receiving double weighting: the sample sizes were *plain* 84, *defaults* 79, *warnings* 88, and *widgets* 157. In this section, we present the results of this evaluation.

### Accuracy

To analyze the impact on entry accuracy, a mixed effect generalized linear analysis of variance (ANOVA) model was fitted using SAS procedure PROC GLIMMIX for determining the effect of feedback types. A Bernoulli distribution with logit link was specified for modeling the binary response of correct/incorrect entry for each question. The model included widget type, feedback type, and their interaction as fixed effects, and the participants as a random effect for accounting for the variation between testers. Dunnetts correction was applied to adjust for multiple-comparisons against the *plain* feedback variation.

The effect of adaptive feedback mechanisms on error rate are summarized and compared to *plain* at both the overall level (Table 3) and by each widget type (Table 4). Each error rate shown was estimated by its least square mean.

Tables 3 shows the error rates of each feedback mechanism, and compare each experimental mechanism to *plain* (using a one-tail test for lower error rate than that of *plain*). The *widget* and *warning* mechanisms improved quality by 52.2% and 56.1%, with marginal significance. The improvement by the *defaults* mechanism was not statistically significant.

| Feedback type | Error rate | vs. $plain$ | Adj. p-value |
|---|---|---|---|
| $plain$ | 1.04% | | |
| defaults | 0.82% | -21.0% | 0.497 |
| widgets | 0.50% | -52.2% | 0.097 |
| warnings | 0.45% | -56.1% | 0.069 |

Table 3: Mean error rates for each feedback variation (across all widget types) and comparisons to the $plain$ control-variation.

Breaking down the results by widget type leads to more statistically significant findings. Table 4 shows accuracy results for each of the *autocompete*, *drop down* and *radio buttons* widget types. Each feedback mechanism improves accuracy in form fields with radio button widgets: the highlighted radio button label (*widgets*) and the *warnings* mechanisms achieve 75% and 78% respective decreases in error %, with statistical significance. The *defaults* mechanism was marginally significant with a 53% decrease in error %.

As expected, the error rate tended to increase with the size of the answer domain for each widget. For the *drop down* and *autocomplete* widgets, we observed some improvements in accuracy, although given the rarity of errors, statistical significance was not observed due to the limited number of trials and participants.[1] Table 2 shows that there were fewer questions in the original form that mapped to these widget types. In general, studying rare events like data entry errors requires a very large number of trials before any effect can be observed. We further discuss the challenges inherent in studying entry errors below.

### Correct vs. Incorrect feedback

We want to investigate the impact of incorrect feedback, We define *correct* as when the true answer is set as a default or

[1]The *drop down* widget with *defaults* feedback was the only trial that resulted in an error rate higher than than of *plain*. A separate two-tail test showed that this difference was also not significant ($p = 0.65$).

| Widget type | Feedback Type | Error rate | vs. $plain$ | Adj. p-value | # wrong / # total |
|---|---|---|---|---|---|
| radio button | $plain$ | 0.99% | | | 22/1760 |
| | defaults | 0.46% | -53.04% | 0.0769 | 9/1659 |
| | widgets | 0.25% | -74.77% | 0.0005 | 10/3297 |
| | warnings | 0.22% | -77.89% | 0.0034 | 5/1846 |
| drop down | $plain$ | 0.47% | | | 2/334 |
| | defaults | 1.09% | 130.45% | 0.9645 | 4/316 |
| | widgets | 0.26% | -44.50% | 0.5041 | 2/628 |
| | warnings | 0.23% | -51.19% | 0.5055 | 1/350 |
| autocomplete | $plain$ | 2.37% | | | 10/336 |
| | defaults | 1.09% | -54.24% | 0.2118 | 4/316 |
| | widgets | 1.85% | -21.98% | 0.5135 | 14/628 |
| | warnings | 1.85% | -21.92% | 0.5494 | 8/352 |

Table 4: Mean error rates for each widget type with break down by feedback type; as well, comparisons to the $plain$ control-variation.

included as one of the $k$ promoted choices, and when the user is warned after entering an actually incorrect answer. We define *incorrect* as the converse of this, and ignore cases when there is no adaptive feedback. To do so, we fit a similar ANOVA model as above for each feedback type, but with one exception: *warnings* feedback given on correctly entered values (false positives) exhibited a 0% error rate, which caused anomalous model estimates. Instead, for the *warnings* variation, we used Fisher's exact test to determine statistical significance. Analysis results are shown in Table 5.

| | Error rate: feedback | | |
|---|---|---|---|
| Feedback type. | correct | incorrect * | Adj. p-value |
| defaults | 0.10% | 10.97% | 0.0001 |
| widgets | 0.28% | 0.84% | 0.0171 |
| warning | 52.94% | 0.00% | 0.0001 |

Table 5: Error rate when the feedback mechanism is correct versus when it is incorrect. Each comparison between correct and incorrect feedback is statistically significant (adjusted $p < 0.05$).

For each experimental variation, as expected, we see that when the feedback mechanism promotes the correct answer, the observed error rate is quite low for *defaults*: 0.1%, and *widgets*: 0.28%. As well, the error rate is quite high for *warnings*: 53%. For *warnings*, when feedback is *correct*, that is, a wrong answer is "caught" by the warning mechanism, the observed 53% error rate means that 47% of these potential errors were corrected. This may seem less than optimal, but given the low baseline error rate, a warning is much more likely to be a false positive for an unlikely answer than an actual input error. In our evaluation, we invoked warnings when the entered answer had less then a $10\%/D$ likelihood, and still only 8.7% of the warnings were issued in the case of actual user error. Given such a large number of false positives, and the low baseline error rate, even catching half of the errors during the warning phase can significantly improve data quality.

When the feedback mechanism promotes an incorrect answer (a false positive), the results are quite different across feedback mechanisms. For *defaults*, incorrect feedback mean that a wrong value is pre-set while the user *acquires the question nature*. In this situation, the observed error rate is an order of magnitude higher than in the $plain$ experiment. Indeed, this is the reason why *defaults* do not perform as well as the other feedback mechanisms. The result suggests that when users are presented with an automatic commitment to an incorrect value, then tend to accept it. Two things are happening: 1) when defaults are set, the user is primed with a candidate answer *prior* to the *acquire answer from source* step, causing a potential for confusion; 2) defaults transform an entry task into a confirmation task, which is easier (next Subsection), but more error prone.

The impact of incorrect feedback for *widgets*, on the other hand, is negligible: we can see that incorrect feedback does not increase the error rate much beyond the *plain* baseline. These during-entry adaptations stay within the *locate answer on screen* stage, and leaves the user to commit to an answer each time. [2]

When *warnings* give incorrect feedback, users receive false alarms on correctly entered values. Observe that the error rate in this situation is 0%. It would appear that users, having committed to a value, are unlikely to change it on a false alarm.

**Effort**

| Feedback type | Duration (sec) | vs $plain$ | Adj. p-value |
|---|---|---|---|
| $plain$ | 2.812 | | |
| defaults | 2.420 | -13.93% | 0.0001 |
| warnings | 2.995 | 6.49% | 0.0001 |
| widgets | 2.787 | -0.89% | 0.8391 |

Table 6: Mean time per question (seconds, estimated by least square mean), and comparison to $plain$ variation, for each experimental variation.

The effect of our feedback mechanisms on the amount of time required to answer a question is summarized in Table 6. A mixed effect ANOVA model with the same fixed and random effects as described in the Accuracy subsection was used to compare the duration between feedback types. We observe that *defaults* led to a $14\%$ faster average question completion time. We did not observe a statistically significant difference for *widgets*. Both results are in accordance

[2]By this logic, the only reason not to provide even *more* feedback is to maintain user trust.

with our goal of reducing the time required to go from paper to electronic. In the *warnings* experiments, we expected a slow-down, and found a small (6%) statistically significant increase in effort. The key conclusion is that our adaptive feedback mechanisms can moderately affect the amount of time required to enter a form, and each adaptation comes with a particular exchange rate when trading off effort vs. quality.

## DISCUSSION

### Every Error Counts

Our baseline $plain$-form gave an error rate of 1.04% . What might this mean, given that our data is used for critical decision-making? Consider the estimated 350-500 million cases of malaria in 2007 [2]. Taking the low-end estimate, if just half resulted in a clinic visit, and on the visit form, a single question recorded malaria-status, then approximately 1.8 million[3] patients' malaria status could be misrepresented in the official records due to entry error, putting their opportunity to receive drugs, consultation or follow-up at risk. Keep in mind that *each* field, on average, could have 1.8 million mistakes. In this admittedly contrived example, our adaptive feedback mechanisms could reduce the number of at-risk patients by more than *half*.

### Room for improvement

Data quality is taken very seriously in the high-resource science of clinical trials, which relies heavily on the practice of double entry [12, 20]. One study showed that double-entry reduced the error rate by 32%, from 0.22% to 0.15% ($p < 0.09$), but increased entry time by 37%, when compared to single-entry [29]. As a point of comparison, our results demonstrate the potential for greater relative improvement in accuracy, and *decrease* in entry time. However, due to the operating conditions, the error rates we observe are an *order of magnitude* higher than those of the clinical trials. We believe the error rates we observed are suppressed due to observer-effects, and that actual error rates may be even higher.

### Generalizability

Error rates vary greatly across operating conditions, depending on various factors including work environment, incentives, user capabilities and training, as well as the specific form questions and entry modality. We hypothesize that data entry programs with higher error rates could benefit even more from our approach. In particular, our approach could be particularly good for input-constrained devices like mobile phones, where even small reductions in effort could lead to dramatic improvements in accuracy (for example, by setting defaults or promoting likely values to the top of drop down menus, which are notoriously difficult to use on mobile devices). In a mobile entry settings, we can also use USHER *dynamic* orderings to further improve the predictive accuracy of the system.

### Studying Rare Events

Throughout this work, we have been constrained by the fundamental *rarity* of making an error. Our own results show that error rates typically range between 0-2%. Conducting a contextual inquiry to understand the source of these errors was exceedingly difficult. It is hard to ever directly observe an error being made, especially because the enterer is being observed. Moreover, because the phenomenon we wanted to observe is so rare, it meant we had to conduct many more trials to obtain statistical significance. We even experimented with various ways of increasing the error rate (do not punish wrong answers, constrain the time even further, etc.) but found that each of those approaches led to aberrant behavior that was not consistent with normal data entry practice.

[3] 350 million $\times \frac{1}{2}$ visit clinic $\times$ 1.04% error rate

## CONCLUSION

We have presented a set of dynamic user interface adaptation for improving data entry and efficiency, driven by a principled probabilistic approach. We evaluated these techniques with real forms and data, entered by professional data entry clerks in Ruhiira, Uganda. Our results show that these mechanisms can significantly improve data entry accuracy and efficiency. Our next step is to develop a version of this system that can work with existing data collection software on mobile phones. We expect that a large number of ongoing mobile data collection projects in the developing world [3, 4, 5] will benefit from this approach. Early discussions confirm this intuition.

Next, we plan to adapt this approach to the related problem of conducting online surveys. Here, we will have to deal explicitly with potential for user *bias* resulting from adaptive feedback. This concern is mitigated for *intermediated* entry, where the person doing the entry is typically not the same as who provides the data. We also plan to explore how USHER's predictive model can be used to detect problematic user behaviors, including detecting user fatigue and *satisficing*, where a respondent does just enough to satisfy form requirements, but nothing more [17]. In general, we believe this approach has wide applicability for improving the quality and availability of all kinds of data, as it is becoming increasingly important for decision-making and resource allocation. We look forward to exploring more such applications in future research.

## ACKNOWLEDGMENTS

The authors would first like to acknowledge the staff and interns at UNDP/MVP Uganda; in particular: Assar Rwakaizi, Benson Tusingwire, Christine Nakachwa, Elly Nankunda, Emmanuel Atuhairwe, Grace Duhimbazimana, Oliver Namatovu and Solomon Echel. We would like to thank our anonymous reviewers, Alice Lin, Andrew Kanter, Bjoern Hartmann, Christine Robson, Guang Chen, Harr Chen, Jamie Lockwood, Jonathan Dick, Kurtis Heimerl, Maneesh Agrawala, Peter Alvaro and Prabhjot Singh for their support and advice. This work was funded in part by the Yahoo Labs Technology for Good Fellowship.

## REFERENCES

1. The Millennium Villages Project. `http://www.millenniumvillages.org`.
2. 2007 Human Development Report. United Nations Development Program. `http://hdr.undp.org/en/reports/global/hdr2007-2008`.
3. CommCare. `http://dimagi.com/commcare`.

4. JavaRosa. `http://www.javarosa.org`.

5. Open Data Kit. `http://opendatakit.org`.

6. OpenMRS. `http://openmrs.org`.

7. The open society: Governments are letting in the light. The Economist: A special report on managing information., Feb. 2010.

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

9. C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2007.

10. K. Chen, H. Chen, N. Conway, T. S. Parikh, and J. M. Hellerstein. Usher: Improving data quality with dynamic forms. In *Proceedings of the International Conference on Data Engineering*, 2010.

11. A. Cockburn, C. Gutwin, and S. Greenberg. A predictive model of menu performance. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, 2007.

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

13. M. W. Eysenck. *Principles of cognitive psychology*. Psychology Press, 1993.

14. L. Findlater, K. Moffat, J. McGrenere, and J. Dawson. Ephemeral adaptation: The use of gradual onset to improve menu selection performance. In *Proceedings of CHI*, 2009.

15. K. Z. Gajos, M. Czerwinski, D. Tan, and D. S. Weld. Exploring the design space for adaptive graphical user interfaces. In *Proceedings of AVI*, 2006.

16. K. Z. Gajos, K. Everette, D. Tan, M. Czerwinski, and D. S. Weld. Predictability and accuracy in adaptive user interfaces. In *Proceedings of CHI*, 2008.

17. R. M. Groves, F. J. Fowler, M. P. Couper, J. M. Lepkowski, E. Singer, and R. Tourangeau. *Survey Methodology*. Wiley-Interscience, 2004.

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

19. L. A. Hermens and J. C. Schlimmer. A machine-learning apprentice for the completion of repetitive forms. *IEEE Expert: Intelligent Systems and Their Applications*, 9(1), 1994.

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

21. D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.

22. J. A. Krosnick. The threat of satisficing in surveys: the shortcuts respondents take in answering questions. Survey Methods Centre Newsletter, 2000.

23. D. Lee and C. Tsatsoulis. Intelligent data entry assistant for xml using ensemble learning. In *Proceedings of ACM IUI*, 2005.

24. J. Martin. *Design of Man-Computer Dialogues*. Prentice-Hall, Inc., 1973.

25. G. A. Miller. The magical number seven, plus or minus two: Some limits on our capacity for information processing. *Psychological Review*, 63(2), 1956.

26. K. Mullet and D. Sano. *Designing Visual Interfaces: Communication Oriented Techniques*. Prentice Hall, 1995.

27. C. Olston and E. H. Chi. Scenttrails: Integrating browsing and searching on the web. *ACM TOCHI*, 10(3), 2003.

28. T. S. Parikh. *Designing an architecture for delivering mobile information services to the rural developing world*. PhD thesis, University of Washington, Seattle, WA, USA, 2007.

29. R. A. Reynolds-Haertle and R. McBride. Single vs. double data entry in cast. *Controlled Clinical Trials*, 13(6), 1992.

30. J. C. Schlimmer and P. C. Wells. Quantitative results comparing three intelligent interfaces for information capture. *Journal of Artificial Intelligence Research*, 5, 1996.

31. A. Sears and B. Shneiderman. Split menus: effectively using selection frequency to organize menus. *ACM Trans. Comput.-Hum. Interact.*, 1(1):27–51, 1994.

32. S. L. Smith and J. N. Mosier. Guidelines for designing user interface software, 1986.

33. S. E. Spenceley, J. R. Warren, S. K. Mudali, and I. D. Kirkwood. Intelligent data entry for physicians by machine learning of an anticipative task model. In *Proceedings of OzCHI*, 1996.

34. J. Warren and P. Bolton. Intelligent split menus for data entry: a simulation study in general practice medicine. *J Amer Med Inform Assoc*, 1999.

35. J. Warren, A. Davidovic, S. Spenceley, and P. Bolton. Mediface: anticipative data entry interface for general practitioners. In *Proceedings of OzCHI*, 1998.

36. W. Willett. Scented widgets: Improving navigation cues with embedded visualizations. *IEEE Transactions on Visualization and Computer Graphics*, 2007.

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