

UC Berkeley

UC Berkeley Electronic Theses and Dissertations

Title

Data-driven Techniques for Improving Data Collection in Low-resource Environments

Permalink

<https://escholarship.org/uc/item/91w1q283>

Author

Chen, Kuang

Publication Date

2011

Peer reviewed|Thesis/dissertation

**Data-driven Techniques for Improving Data Collection in Low-resource
Environments**

by

Kuang Chen

A dissertation submitted in partial satisfaction
of the requirements for the degree of

Doctor of Philosophy

in

Computer Science

in the

GRADUATE DIVISION

of the

UNIVERSITY OF CALIFORNIA, BERKELEY

Committee in charge:

Professor Joseph M. Hellerstein, Chair
Professor Tapan S. Parikh
Professor Eric Brewer

Fall 2011

Data-driven Techniques for Improving Data Collection in Low-resource Environments

Copyright © 2011

by

Kuang Chen

Abstract

Data-driven Techniques for Improving Data Collection in Low-resource Environments

by

Kuang Chen

Doctor of Philosophy in Computer Science

University of California, Berkeley

Professor Joseph M. Hellerstein, Chair

Low-resource organizations worldwide work to improve health, education, infrastructure, and economic opportunity in disadvantaged communities. These organizations must collect data in order to inform service delivery and performance monitoring. In such settings, data collection can be laborious and expensive due to challenges in the physical and digital infrastructure, in capacity and retention of technical staff, and in poor performance incentives. Governments, donors, and non-governmental organizations (NGOs) large and small are demanding more accountability and transparency, resulting in increased data collection workloads. Despite continued emphasis and investment, countless data collection efforts continue to experience delayed and low-quality results. Existing tools and capabilities for data collection have not kept pace with increased reporting requirements.

This dissertation addresses data collection in low-resource settings by algorithmically shepherding human attention at three different scales: (1) by redirecting workers' attention at the moment of entry, (2) by reformulating the data collection instrument in its design and use, and (3) by reorganizing the flow and composition of data entry tasks within and between organizations. These three different granularities of intervention map to the three major parts of this dissertation.

First, the USHER system learns probabilistic models from previous form responses, supplementing the lack of expertise and quality control. The models are a principled foundation for data in forms, and are applied at every step of the data collection process: form design, form filling, and answer verification. Simulated experiments demonstrate that USHER can improve data quality and reduce quality-control effort considerably.

Next, a number of dynamic user-interface mechanisms improve accuracy and efficiency during the act of data entry, powered by USHER. Based on a cognitive model, these interface adaptations can be applied as interventions before, during, and after input. An evaluation with professional data entry clerks in rural Uganda reduced error by up to 78%.

Finally, the SHREDDR system transforms paper form images into structured data on-demand. SHREDDR reformulates data entry work-flows with pipeline and batching optimizations at the organizational level. It combines emergent techniques from computer vision, database systems, and machine learning, with newly-available infrastructure —

on-line workers and mobile connectivity — into a hosted data entry web-service. It is a framework for data digitization that can deliver USHER and other optimizations at scale. SHREDDR's impact on digitization efficiency and quality is illustrated in a one-million-value case study in Mali.

The main contributions of this dissertation are (1) a probabilistic foundation for data collection, which effectively guides form design, form filling, and value verification; (2) dynamic data entry interface adaptations, which significantly improve data entry accuracy and efficiency; and (3) the design and large-scale evaluation of a hosted-service architecture for data entry.

For Sophie and children who are not counted.

Contents

Contents	ii
List of Figures	vi
List of Tables	viii
Acknowledgements	ix
1 Introduction	1
1.1 Data in the First Mile	1
1.2 Challenges	3
1.2.1 Expertise, Training and Turnover	3
1.2.2 Telling Stories vs. Encoding Data	3
1.2.3 The Burden of Monitoring and Evaluation	4
1.2.4 Lack of Local Usage	4
1.3 Approach	5
1.4 Contributions	8
2 Usher	9
2.1 Introduction	9
2.2 Related Work	10
2.2.1 Data Cleaning	11
2.2.2 Data Entry Efficiency	11
2.2.3 Clinical Trials	11
2.2.4 Survey Design	12
2.3 System	12

2.3.1	Examples	13
2.3.2	Implementation	16
2.4	Learning a Model for Data Entry	17
2.5	Question Ordering	19
2.5.1	Reordering Questions during Data Entry	21
2.6	Question Re-asking	22
2.6.1	Error Model	23
2.6.2	Error Model Inference	25
2.6.3	Deciding when to Re-ask	25
2.7	Question Reformulation	26
2.7.1	Static Reformulation	27
2.7.2	Dynamic Reformulation	28
2.7.3	Reformulation for Re-asking	28
2.8	Evaluation	29
2.8.1	Data Sets and Experimental Setup	29
2.8.2	Ordering	30
2.9	Conclusion and Future Work	34
3	Adaptive Feedback	36
3.1	Introduction	36
3.2	Related Work	37
3.2.1	Managing Data Quality During Entry	37
3.2.2	Improving Data Entry Efficiency	37
3.2.3	Adaptive User Interfaces	38
3.3	A Data-driven Foundation For Adaptive Forms	38
3.4	Opportunities For Adaptation During Entry	38
3.5	Adaptive Feedback For Data Entry	41
3.5.1	Design Process	41
3.5.2	Feedback Mechanisms	43
3.5.3	Feedback Parameterization	44
3.6	User Study	46
3.6.1	Context and Participants	46

3.6.2	Forms and Data	46
3.6.3	An USHER Model for Patient Visits	47
3.6.4	System	47
3.6.5	Task	48
3.6.6	Procedure	48
3.7	Results	49
3.7.1	Accuracy	49
3.7.2	Correct vs. Incorrect feedback	50
3.7.3	Effort	52
3.8	Discussion	52
3.8.1	Every Error Counts	52
3.8.2	Room for improvement	53
3.8.3	Generalizability	53
3.8.4	Studying Rare Events	53
3.9	Conclusion and Future Work	54
4	Shreddr	55
4.1	Introduction	55
4.2	Related Work	56
4.2.1	Automatic document processing	56
4.2.2	Improving data entry	56
4.3	Existing Solutions	57
4.3.1	Replacing paper with direct-entry	57
4.3.2	Traditional double data entry	58
4.4	SHREDDR	58
4.5	Why “shred”?	60
4.5.1	Capture and encode separately	61
4.5.2	Paper independence	61
4.5.3	Create economies of scale	63
4.6	Case Study	63
4.6.1	Imaging and uploading	64
4.6.2	Alignment and shredding	64

4.6.3	Document definition	64
4.6.4	Digitization effort	65
4.6.5	Workers and errors	66
4.7	Data Quality	66
4.7.1	Double entry quality	67
4.7.2	Quality control	67
4.7.3	SHREDDR results	69
4.7.4	Quality comparison	71
4.7.5	Error independence	73
4.8	Efficiency	74
4.8.1	Shredded interfaces	74
4.9	Discussion	77
4.10	Conclusion and Future Work	78
5	Conclusion	79
5.1	Discussion and Future Work	80
5.1.1	Maximizing bits-per-decision	80
5.1.2	Bottom-up bootstrapping	80
5.1.3	Getting Started	80
	Bibliography	82

List of Figures

1.1	Photo of a hospital file room in Uganda, 2008	5
2.1	Learned structure for <i>patient</i> data.	14
2.2	Learned structure for <i>survey</i> data	15
2.3	Optimized question ordering.	16
2.4	USHER system diagram.	17
2.5	USHER error model.	22
2.6	Question reformulation example.	26
2.7	Ordering experiment results.	30
2.8	Re-asking experiment results.	32
2.9	Reformulation experiment results.	33
3.1	USHER system overview.	39
3.2	Data entry cognitive task model.	40
3.3	Study participants.	41
3.4	Alternative data entry widgets	42
3.5	Adaptive data entry widgets.	43
3.6	How many values to show?	45
3.7	Model predictiveness.	47
4.1	Document definition interface.	58
4.2	Makeshift copy stand.	59
4.3	Form image registration outcome.	60
4.4	Shredding.	61
4.5	Entry interface.	62
4.6	Quality control work-flow.	68

4.7	Accuracy results.	69
4.8	Difficult shred images	70
4.9	Entry of Ugandan last names	71
4.10	Verification interface: list-style.	74
4.11	Value-ordered verification interface: grid-style.	75
4.12	Duration by data type and question difficulty	76

List of Tables

3.1	Answer domain—feedback positions map.	45
3.2	“Adult Outpatient” form and dataset questions.	46
3.3	Error rates.	50
3.4	Error rates by type.	50
3.5	Error rate by feedback correctness.	51
3.6	Mean time per question.	52
4.1	Mali run statistics	65
4.2	Worker demographics	65
4.3	Constituting ideal double entry.	73
4.4	Error independence.	73

Acknowledgements

First and foremost, I am grateful for my wonderful advisors Joe and Tapan – they are kind, forgiving, but with high standards. I could not have imagined better mentors. Thank you to Maneesh Agrawala and Eric Brewer – my qualifying examination committee, who have been always available and generous with their thoughts.

Thank you to Harr Chen – I could not have done it without you.

Jamie Lockwood has been my cheerleader and champion – having faith in me when I was doubtful, and finding support for me to take the first steps in this research agenda. I am grateful for her kindness and encouragement.

Thank you to Neal Lesh for packing me off to Tanzania, and being there when I landed. I am grateful to the staff at the MDH project, and Sanjay Unni for showing me around Dar es Salaam. Thanks to Kurtis Heimerl, Yaw Anokwa, Brian DeRenzi and the rest of the Young Researchers in ICTD crew.

I am in debt to Christine Robson for her ample advice and patience, and extraordinary editorial ability.

Thank you to my prelim buddy Beth Trushkowsky, my break-time buddy Tyson Condie, and the rest of the guys in the Berkeley databases group – Peter Alvaro and Neil Conway.

Thanks to Phil Garland and Dave Goldberg at SurveyMonkey for supporting Usher and providing an example of a developing-world solution being fruitful at home.

Thank you to Melissa Ho Densmore for showing me the ropes in Mbarara, Uganda. She and Jessica Gottleib entrusted me with their PhD dissertation data, and were my patient and understanding test subjects. Your thoughtful feedback was invaluable.

I have warm memories and heartfelt gratitude for the staff and interns at the United Nations Development Programme and the Millennium Villages Project, Uganda; in particular: Assar Rwakaizi, Benson Tusingwire, Christine Nakachwa, Dieuwertje Heil, Elly Nankunda, Emmanuel Atuhairwe, Grace Duhimbazimana, Jessica Rijpstra, Jonathan Dick, Naomi Handa-Williams, Oliver Namatovu and Solomon Echel. Big thanks to Prabhjot Singh and Andrew Kanter for their support and advice.

Thank you to everyone who helped Shreddr take shape, especially Akshay Kannan and Andrea Spillmann.

To everyone at Captricity: Jeff Lin, Nick Jalbert, Lee Butterman, Trevor Smith, Jake Abernathy, Yori Yano, Logan Bowers, Nadja Haldermann, Jon Barron, and Rob Carroll – thank you for sharing this vision.

To my parents, Mingfei and Guang and Jiamin and Tetsu – thank you for everything.

To Alice, I joyfully have the rest of my life to thank you. To Sophie, your impending arrival helped push me over the finish line.

Chapter 1

Introduction

“To make people count, we first need to be able to count people.”

Dr. Jong-wook Lee, Director-General, World Health Organization

(2004-2006)

1.1 Data in the First Mile

Digital information is becoming “a form of infrastructure; no less important to modern life than roads, electrical grid or water systems” [10]. The potential for data to improve the efficacy and accountability of organizations is evident from the existence of billion-dollar industries serving business intelligence and data management needs.

Meeting the challenge of global development requires improving the economic opportunities, health, education, and infrastructure available to billions of people living in sub-standard and isolated conditions worldwide. Practitioners of global development are becoming increasingly *data-driven*, basing policies and actions on context-specific knowledge about local needs and conditions. Leading development organizations—with the help of research specialists like the Poverty Action Lab—undertake rigorous impact evaluations of development interventions, driven by the “belief in the power of scientific evidence to understand what really helps the poor.”¹

Unfortunately, the most under-developed communities are still beyond the reach of modern data infrastructure, constrained by limitations in physical and digital infrastructure, in capacity and retention of technical staff, and in performance incentives. As well,

¹<http://www.povertyactionlab.org>

funders' increasing emphasis on monitoring and evaluation (M&E) increases data collection workloads. Despite continued investment, data collection efforts struggle to keep up with demand.

Part of this problem has been the lack of Internet connectivity into under-served communities. Computer networking researchers and telecommunications policy experts often refer to it as “bridging the last mile.” However, much progress has been made connecting the last mile: mobile adoption has rapidly expanded network coverage; all but the most remote locations seem poised to be connected. Yet, connectivity alone does not ensure data availability. Even basic vital statistics are still largely unavailable—for example, only 24% of children born in East and Southern Africa are registered [78], rendering the remaining children invisible to decisions regarding resources and policy. For data scientists, this last mile is our “first mile”—where essential local data is being created, and the hard work of building modern data infrastructure is just beginning.

Without proper data infrastructure, practitioners, policy-makers, and researchers must rely on incomplete, inaccurate, and delayed information for making critical decisions. For example, the public health community warns of the “innovation pile-up” [23]: scientific advances such as new vaccines can sit idle, awaiting data about how to efficiently deliver new solutions and encourage adoption. Advances in database systems research suffer from a similar innovation pile-up. For want of data and knowledge about first-mile problems, some of our best ideas—including those that can be applied to solve those very first-mile data collection challenges—have been sidelined.

The tools designed for the developed-world usage cases tend to be inappropriate for low-resource work-flows. For example, data quality is a critical consideration in any data collection efforts, but techniques for automatic *data cleaning* tend to focus on data after it has been collected into a database [20, 12]. As well, automated methods of extracting data from paper forms [48] can be immensely beneficial to first-mile work-flows, but none of these methods have been widely adopted in the developing world due to cost and technical complexity.

The increasing availability of phones, computers, and Internet connectivity has the potential for streamlining data collection and management in first-mile environments. Several researchers have developed mobile-phone-based frameworks for data collection [7, 34, 60]. Mobile devices enable remote agents to directly enter information at the point of service, replacing data entry clerks and providing quick response times.

However, mobile direct entry tools often replace existing paper-based work-flows, and going “paperless” is not an option for many first-mile organizations [60]. Paper remains the time-tested and preferred data capture medium, usually for a combination of the following reasons:

- Resource limitations: lack of capital, stable electricity, IT-savvy workers, and system administrators
- Inertia: fear that even small changes to deeply ingrained paper-based work-flows can cause large ripple effects

- Regulation: compliance with paper record-keeping rules from the myriad oversight agencies
- Backup: need to safeguard against power or electronic system failure, and for having a primary-source reference [62])

The vast majority of first-mile organizations hire on-site data entry workers to transcribe their stacks of paper. To address the end-to-end challenge of building data infrastructure in first-mile communities, we must address the paper-to-structured-data digitization problem.

1.2 Challenges

My experiences volunteering and doing research in East Africa with public health and international development organizations provided the following perspectives about the nature of data digitizing bottlenecks in low-resource organizations. I saw that existing methods fail to meet a host of nuanced contextual challenges. This dissertation begins by organizing and analyzing the challenges beyond the lack of resources.

1.2.1 Expertise, Training and Turnover

In low-resource organizations, even office-based administrative staff lack expertise in critical areas like database and computer systems administration, form design and data entry, and usability and process engineering. This is especially true for small grassroots groups and the local field offices of international organizations, which are assigned the most critical and challenging task of actual service delivery. It is very expensive to provide IT training and expertise in remote and unappealing locations. For the same reason, it is difficult to recruit and retain high-quality talent. The best staff almost always leave to climb the career ladder, eventually ending up with a job in a major city, or even abroad. Turnover is very high, especially among young, English-speaking, computer-literate workers. As such, even organizations that invest heavily in training see limited returns.

1.2.2 Telling Stories vs. Encoding Data

Data entry is an encoding task, and doing it well requires domain knowledge, and sufficient literacy and numeracy. The field staff of low-resource organizations often have limited formal education. Previous empirical work has shown that uneducated users have difficulty organizing and accessing information as abstract concepts [72]. These characteristics have in turn been associated with the concept of “orality” [59]. According to this theory, oral

cultures are characterized by situational rather than abstract logic, preferring nuanced, qualitative narratives to quantitative data. Oral knowledge processes are also aggregative rather than analytic, favoring the assembly of complex and potentially-conflicting “stories,” rather than the documentation of experiences as individual “correct” measurements.

1.2.3 The Burden of Monitoring and Evaluation

Like enterprises in the developed world, monitoring organizations are becoming increasingly data-driven. As the World Bank reports, “Prioritizing for M&E has become a mantra that is widely accepted by governments and donors alike” [38]. On-the-ground organizations face, on one hand, growing data collection requirements, and on the other the mandate to minimize “administration” overhead—the budget that sustains data collection. Some international funders assume that if an on-the-ground organization is effective and worthy of aid, then their reporting requirements can be met with data already being collected [30]. This is wishful thinking and far from reality. Organizations active in local communities are often several rungs down on the sub-contracting or delegation ladder, and hence are disconnected from the rationale behind reporting requirements.

Specifically, generating the aggregated long-term data (months to years) that is useful for top-down evaluation and policy is very different from generating the fine-grain short-term data useful for making decisions at the local level. For example, a funder may be interested in a quarterly count of patients with malaria, while a health clinic wants to know which malaria patients from yesterday require follow-up. The latter information for improving service delivery is often skipped due to priority of reporting requirements.

Worse, because digitization workloads are uneven—for example, an immunization drive can cause a weeks-long data entry backlog in a rural health clinic—organizations need to over-provision worker capacity in order to have reliably fast turn-around times. Due to resource constraints, this is rarely possible. Instead, when reports are due, staff are pulled from other vital duties to work as expensive data entry clerks. I witnessed that doctors and nurses in many rural health clinics must stop providing medical services in order to spend days each month tabulating reports.

1.2.4 Lack of Local Usage

In an urban Tanzanian health clinic, I observed that patient visit data was recorded by hand twice, then digitally entered three times. A nurse first wrote in a paper register used for the clinic’s day-to-day operations, and next copied the information onto carbon-copy paper forms. The first copy, for the local ministry of health, was digitally entered on-site; the second copy, for medical research was shipped to a central office and separately entered twice-over into a different database.

A paradoxical challenge in paper data digitization is the ease of establishing “good-

enough” solutions, without regard to existing work-flows. The above clinic, like the many organizations worldwide, began digital data collection with simple tools—*ad hoc* spreadsheets or Microsoft Access databases. Such practices can be quite effective when there is the organizational will and resources to maintain these solutions and evolve them to accommodate changing requirements.



Figure 1.1. Photo of a hospital file room in Uganda, 2008

The Tanzanian clinic did not have such resources—they could not adapt the existing paper-only system, and thus had to duplicate transcription efforts into a tailored “point solution,” which minimally met the requirements of a single report. As a result, staff had no access their own digital data despite on-site data entry. Instead, staff relied on searching through paper forms on file-room shelves (Figure 1.1).

Data that local staff must collect but cannot use can be a strong disincentive to generate high-quality results: staff take lackadaisical attitudes towards data collection when the impact is unseen, and become resentful when they feel efforts are unnecessarily duplicated. As well, as the number of reports increase, so do the number of point solutions.

Local disuse can also lead to field staff emphasizing successes, and minimizing failures. After all, if digital data has no local impact, then reports can be biased towards what distant overseers want to see. Biases aggregate over time and geography, potentially obscuring serious local problems. Ironically, the typical reaction of the distant overseers to poor data quality is to increase the intensity of M&E—adding work to already-stretched local work-flows and reinforcing a vicious cycle.

1.3 Approach

Broadly speaking, meeting the above challenges requires improving three key factors—data accuracy, cost, and turn-around time—in under-staffed organizations with little domain or IT expertise, and poorly-trained workers who do not see the benefits of the increasing data collection workload (Section 1.2.4). However, each organization is subject

to its own combination of constraints, and places different relative priority on these factors. Technology interventions must carefully consider the specific requirements of each application domain and organization. The CAM system is an exemplary intervention in the domain of micro-finance. In his dissertation, Parikh described using mobile phones as digital imaging and data entry devices to aid in existing paper-based work-flows [60].

This dissertation extends the idea that solutions based on existing practices and nuanced understanding of local problems work well, and takes a *data-driven* and thus, adaptive, approach to augment existing paper-based work-flows. We focus our interventions on improving data quality and operational cost, trading off the immediate responsiveness that is possible with direct-entry on mobile-devices. Our systems and algorithms learn from data being collected to make custom augmentations to the modality, interaction, location and even choice of participants involved in local data collection processes. These augmentations are design to address some of the specific challenges in Section 1.2.

Our data-driven interventions algorithmically usher human attention at three scales: (1) during the moment of entry, we redirect workers' attention towards important data elements and likely errors; (2) spanning the design and usage of a data collection instrument, we reformulate the instrument's design and transcription process in terms of form layout, question wording, and answer verification logic; (3) at the organization level, we reorganize the flow and composition of data collection work.

First, we focus on data quality. The field of survey methodology has long been shepherding form designers with heuristics for crafting data collection instruments, and data managers with best practices for data quality [32]. As mentioned in Section 1.2.1, survey design expertise and sufficient resources for quality assurance are rare in resource-challenged settings. Often, poor quality data intensifies the mandate for more data collection (Section 1.2.3). To guide the design and usage of a data collection instrument, we take a machine-learning-based approach to supplement the lack of expertise and quality assurance. The foundation of this approach is a probabilistic representation of data collection forms induced from existing form values and their relationships between questions. The model represents data collection as an entropy-reduction process.

The USHER system trains and applies these probabilistic models in order to automatically derive process improvements in question flow, question wording, and re-confirmation.

- Since form layout and question selection is often ad hoc, USHER optimizes question ordering according to a probabilistic objective function that aims to maximize the information content of form answers as early as possible. Applied before entry, the model generates a static but entropy-optimal ordering, which focuses on important questions first.
- Applying its probabilistic model during data entry, USHER can evaluate the conditional distribution of answers to a form question. For difficult-to-answer questions, such as those with many extraneous choices, USHER can opportunistically *reformulate* them to be easier and more congruous with the available information.
- The model is consulted to predict which responses may be erroneous, so as to re-ask

those questions in order to verify their correctness. Intelligent re-asking approximates the benefits of double entry at a fraction of the cost.

Simulated experiments confirm that USHER can significantly improve input efficiency and accuracy.

During the act of data entry, a number of USHER-powered dynamic user-interface mechanisms can be used to improve accuracy and efficiency. The intuition is that the amount of effort, or *friction*, in the input interface should be proportion to answer likelihood. Based on a cognitive model that identifies the opportunity for applying friction, each act of entry is adaptive and a customized experience. Before and during entry, adaptations decrease the time and cognitive effort required for transferring an answer from paper, and reduce the possibility that the answer is lost from short-term memory in the transfer process. After entry, adaptations increase the time and cognitive effort spent to verify the answer’s correctness, with particular focus on important and likely-erroneous values. An evaluation with professional data entry clerks in rural Uganda showed the potential to reduce error by up to 78%.

At the organizational scale, we focus on re-architecting local data work-flows to introduce optimizations that can improve efficiency and lower cost. We created the SHREDDR system, which transforms paper form images into structured data as a hosted data entry web-service. SHREDDR leverages partitioning and batching optimizations by combining emergent techniques from computer vision, database systems, and machine learning, with newly-available infrastructure—on-line workers and mobile connectivity.

SHREDDR is a data entry assembly line, which first separates data capture and encoding into distinct steps. Front-line field workers are the best suited to capture information given their local contextual knowledge and familiarity with the community, whereas tasks involving data encoding require more literacy, training and knowledge about increasingly-specific data vocabularies and schemata (Section 1.2.2). The distinction makes it possible to move tasks involving data encoding to where the incentives and capabilities are more appropriate—away from the first-mile. By pooling the work from multiple organizations, we benefit from the economy of scale. A constant flow of work allows SHREDDR to further separate and reformulate data encoding tasks in additional assembly line steps. These steps are opportunities for existing and novel techniques to add value; for example: computer vision algorithms pre-process document images for legibility, and human workers on-line perform data entry and verification.

SHREDDR’s human data entry interfaces leverage batch compression. The inspiration came during USHER’s user study in an Ugandan health clinic. I noticed that a data entry clerk was mouthing numbers while inputting “yes/no” values. When asked, he pointed to the sequence of consecutive “yes” answers on the paper form, and explained that by memorizing the number of “yes” answers in the span, he could enter that many without looking back at the paper form. I recognized that he was performing on-the-fly data compression: specifically, he was unknowingly applying the equivalent of run-length encoding². Such a batch operation over many values at once can be engineered to occur as often as possible by

²http://en.wikipedia.org/wiki/Run-length_encoding

controlling information entropy. SHREDDR controls information entropy by decomposing filled-in paper images into constituent values, and then grouping these by value. Batches are then placed into batch operation interfaces for workers.

SHREDDR's algorithms for partitioning and batching data entry work, when combined with an elastic pool of on-line workers, can allow significant efficiency gains, as well as fast turn-around times. We describe a large-scale deployment in Mali, which digitized one million data values with higher quality, faster turn-around time, and lower cost compared to current practices.

1.4 Contributions

The main contributions of this research are as follows:

- A probabilistic foundation for data collection, which functions in the USHER system to improve form design, form filling, and value verification
- Design guidelines for dynamic data entry interface adaptations based on USHER's probabilistic machinery, and evaluation *in situ*
- The architecture of the SHREDDR data entry web-service, and its evaluation with a large-scale deployment

Chapter 2

Usher

2.1 Introduction

Current best practices for managing data quality during data entry come from the field of survey methodology, which offers principles that include manual question orderings and input constraints, and double entry of paper forms [32]. Although this has long been the *de facto* quality assurance standard in data collection and transformation, we believe this area merits reconsideration. For paper forms, we posit that a data-driven and more computationally sophisticated approach can significantly outperform these decades-old static methods in both accuracy and efficiency of data entry.

The problem of data quality is magnified in low-resource settings. First, many organizations lack expertise in form design: designers approach question and answer choice selection with a defensive, catch-all mindset, adding answer choices and questions that may not be necessary. Furthermore, they engage in ad hoc mapping of required data fields to data entry widgets by intuition [16, 52], often ignoring or specifying ill-fitting constraints.

Second, double data entry (DDE) is too costly. In some cases this means it is simply not performed, resulting in poor data quality. In other cases, particularly when DDE is mandated by an oversight agency, it results in delays and other unintended negative consequences. When DDE is imposed upon a busy field office, the effort can delay data availability for months—data is most often not used until both entry passes are finished and conflicts reconciled. Although the data eventually percolates up to national and international agencies, the clinic remains unable to benefit from the data they themselves digitized.

To address this spectrum of data quality challenges, we have developed USHER, an end-to-end system that can improve data quality and efficiency at the point of entry by learning *probabilistic models* from existing data, which stochastically relate the questions

of a data entry form. These models form a principled foundation on which we develop information-theoretic algorithms for form design, dynamic form adaptation during entry, and answer verification:

- When forms are designed or revised, USHER can optimize question *ordering* according to a probabilistic objective function that aims to maximize the information content of form answers as early as possible—we call this the *greedy information gain* principle. Applied before entry, the model generates a static but entropy-optimal ordering, which focuses on important questions first. During entry, it can be used to dynamically pick the next best question, based on answers so-far—this is appropriate in scenarios where question ordering can be flexible between instances.
- Applying its probabilistic model during data entry, USHER can evaluate the conditional distribution of answers to a form question, and make it easier for likely answers to be entered—we call this the *appropriate entry friction* principle. For difficult-to-answer questions, such as those with many extraneous choices, USHER can opportunistically *reformulate* them to be easier. In this way, USHER effectively allows for a principled, controlled tradeoff between data quality and form filling effort and time.
- Finally, the stochastic model is consulted to predict which responses may be erroneous, so as to *re-ask* those questions in order to verify their correctness—we call this the *contextualized error likelihood* principle. We consider re-asking questions both during the data entry process (integrated re-asking) and after data entry has been finished (post-hoc re-asking). In both cases, intelligent question re-asking approximates the benefits of double entry at a fraction of the cost.

The contributions of this chapter are three-fold:

1. We describe the design of USHER’s core: probabilistic models for arbitrary data entry forms.
2. We describe USHER’s application of these models to provide guidance along each step of the data entry lifecycle: reordering questions for greedy information gain, reformulating answers for appropriate entry friction, and re-asking questions according to contextualized error likelihood.
3. We present experiments showing that USHER has the potential to improve data quality at reduced cost. We study two representative data sets: direct electronic entry of survey results about political opinion, and transcription of paper-based patient intake forms from an HIV/AIDS clinic in Tanzania.

2.2 Related Work

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

2.2.1 Data Cleaning

In the database literature, data quality has typically been addressed under the rubric of *data cleaning* [12, 20]. Our work connects most directly to data cleaning via multivariate outlier detection; it is based in part on interface ideas first proposed by Hellerstein [36]. 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 at the time of data entry. Thus, it can catch errors when they are made and when ground-truth values may still be available for verification.

2.2.2 Data Entry Efficiency

Past research on improving data entry is mostly focused on adapting the data entry interface for user efficiency improvements. Several such projects have used learning techniques to automatically fill or predict a top-k set of likely values. For example, Ali and Meek [11] predicted values for combo-boxes in web forms and measured improvements in the speed of entry, Ecopod [83] generated type-ahead suggestions that were improved by geographic information, and Hermens et al. [37] automatically filled leave-of-absence forms using decision trees and measured predictive accuracy and time savings. In these approaches, learning techniques are used to predict form values based on past data, and each measures the time savings of particular data entry mechanisms and/or the proportion of values their model was able to correctly predict.

USHER’s focus is on improving data quality, and its probabilistic formalism are based on learning relationships within the underlying data that guide the user towards correct entries. In addition to predicting question values, we develop and exploit probabilistic models of user error, and target a broader set of interface adaptations for improving data quality, including question reordering, reformulation, and re-asking. Some of the enhancements we make for data quality could also be applied to improve the speed of entry.

2.2.3 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 [21, 42]. In particular, Kleinman takes a probabilistic approach toward choosing which forms to re-enter based on the individual performance of data entry staff [44]. 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 entry worker, 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 as-

assessment of the tradeoff between cost (of reconfirming more questions) versus quality (as predicted by the probabilistic model).

2.2.4 Survey Design

The survey design literature includes extensive work on form design techniques that can improve data quality [32, 56]. 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 many of these same constraints without requiring their explicit specification. When these constraints are violated during entry, USHER can then flag the relevant questions, or target them for re-asking.

It is important to note that USHER does not preclude the manual specification of constraints. This is critical, because previous research into the psychological phenomena of survey filling has yielded common constraints not inherently learnable from prior data [32]. 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.” USHER complements these human-specified constraints, accommodating them while leveraging any remaining flexibility to optimize question ordering in a data-driven manner.

2.3 System

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, resulting in a *Bayesian network*; and second, by estimating the parameters of that Bayesian network, which then allows us to generate predictions and error probabilities for the form.

After the model is built, USHER uses it to automatically *order* a form’s questions for *greedy information gain*. Section 2.5 describes both static and dynamic algorithms that employ criteria based on the magnitude of statistical information gain that is expected in answering a question, given the answers that have been provided so far. This is a key idea in our approach. By front-loading predictive potential, we increase the models’ capacity in several ways. First, from an information theoretic perspective, we improve our ability to do multivariate prediction and outlier detection for subsequent questions. As we discuss in more detail in Section 2.7, this predictive ability can be applied by reformulating error-prone form questions, parametrizing data entry widgets (type-ahead suggestions, default

values), assessing answers (outlier flags), and performing in-flight re-asking (also known as cross-validation in survey design parlance). Second, from a psychological perspective, front-loading information gain also addresses the human issues of user fatigue and limited attention span, which can result in increasing error rates over time and unanswered questions at the end of the form.

Our approach is driven by the same intuition underlying the practice of *curbstoning*, which was related to us in discussion with survey design experts [52]. Curbstoning is a way in which an unscrupulous door-to-door surveyor shirks work: he or she asks an interviewee only a few *important* questions, and then uses those responses to complete the remainder of a form while sitting on the curb outside the home. The constructive insight here is that a well-chosen subset of questions can often enable an experienced agent to intuitively predict the remaining answers. USHER’s question ordering algorithms formalize this intuition via the principle of greedy information gain, and use them (scrupulously) to improve data entry.

USHER’s learning algorithm relies on 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; this is no worse than standard practice. Subsequently, a training set can gradually be constructed by iteratively capturing data from designers and potential users in “learning runs.” It is a common approach to first fit to the available data, and then evolve a model as new data becomes available. This process of semi-automated form design can help institutionalize new forms before they are deployed in production.

USHER adapts to a form and dataset by crafting a custom model. Of course, as in many learning systems, the model learned may not translate across contexts. We do not claim that each learned model would or should fully generalize to different environments. Instead, each context-specific model is used to ensure data quality for a *particular* situation, where we expect relatively consistent patterns in input data characteristics. In the remainder of this section, we illustrate USHER’s functionality with examples. Further details, particularly regarding the probabilistic model, follow in the ensuing sections.

2.3.1 Examples

We present two running examples. First, the *patient* dataset comes from paper patient-registration forms transcribed by data entry workers at an HIV/AIDS program in Tanzania.¹ Second, the *survey* dataset comes from a phone survey of political opinion in the San Francisco Bay Area, entered by survey professionals directly into an electronic form.

In each example, a form designer begins by creating a simple specification of form questions and their prompts, response data types, and constraints. The training data set is made up of prior form responses. Using the learning algorithms we present in Section 2.4, USHER builds a Bayesian network of probabilistic relationships from the data, as shown

¹We have pruned out questions with identifying information about patients, as well as free-text comment fields.

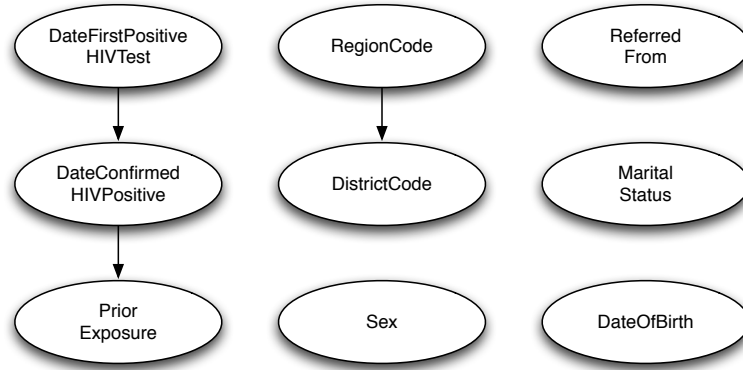


Figure 2.1. Bayesian network for the *patient* dataset, showing automatically inferred probabilistic relationships between form questions.

in Figures 2.1 and 2.2. In this graph, an edge captures a close stochastic dependency between two random variables (i.e., form questions). Two questions with no path between them in the graph are probabilistically independent. Figure 2.2 illustrates a denser graph, demonstrating that political survey responses tend to be highly correlated. Note that a standard joint distribution would show correlations among *all pairs* of questions; the sparsity of these examples reflects conditional independence patterns learned from the data. Encoding independence in a Bayesian network is a standard method in machine learning that clarifies the underlying structure, mitigates data over-fitting, and improves the efficiency of probabilistic inference.

The learned structure is subject to manual control: a designer can override any learned correlations that are believed to be spurious or that make the form more difficult to administer.

For the *patient* dataset, USHER generated the static ordering shown in Figure 2.3. We can see in Figure 2.3 that the structure learner predicted *RegionCode* to be correlated with *DistrictCode*. Our data set is collected mostly from clinics in a single region of Tanzania, so *RegionCode* provides little information. It is not surprising then, that USHER’s suggested ordering has *DistrictCode* early and *RegionCode* last—once we observe *DistrictCode*, *RegionCode* has very little additional expected conditional information gain. When it is time to input the *RegionCode*, if the user selects an incorrect value, the model can be more certain that it is unlikely. If the user stops early and does not fill in *RegionCode*, the model can infer the likely value with higher confidence. In general, static question orderings are appropriate as an offline process for paper forms where there is latitude for (re-)ordering questions, within designer-specified constraints.

During data entry, USHER uses its probabilistic machinery to drive dynamic updates to the form structure. One type of update is the dynamic selection of the best next question to ask among questions yet to be answered. This can be appropriate in several situations, including surveys that do not expect users to finish all questions, or direct-entry interfaces (e.g., mobile phones) where one question is asked at a time. We note that it is still im-

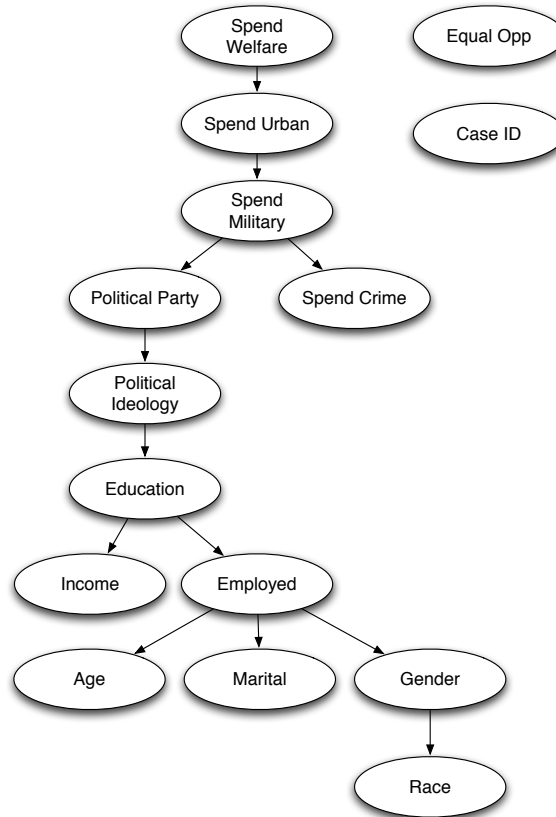


Figure 2.2. Bayesian network for the *survey* dataset. The probabilistic relationships are more dense. Some relationships are intuitive (Political Ideology - Political Party), others show patterns incidental to the dataset (race - gender).

portant to respect the form designer’s *a priori* specified question-grouping and -ordering constraints when a form is dynamically updated.

USHER is also used during data entry to provide dynamic feedback, by calculating the conditional distribution for the question in focus and using it to influence the way the question is presented. We tackle this via two techniques: question reformulation and widget decoration. For the former, we could for example choose to reformulate the question about RegionCode into a binary yes/no question based on the answer to DistrictCode, since DistrictCode is such a strong predictor of RegionCode. As we discuss in Section 2.7, the reduced selection space for responses in turn reduces the chances of a data entry worker selecting an incorrect response. For the latter, possibilities include using a “split” drop-down menu for RegionCode that features the most likely answers “above the line,” and after entry, coloring the chosen answer red if it is a conditional outlier. We discuss in Section ?? the design space and potential impact of data entry feedback that is more specific and context aware.

As a form is being filled, USHER calculates contextualized error probabilities for each question. These values are used for re-asking questions in two ways: during primary form entry and for reconfirming answers after an initial pass. For each form question, USHER

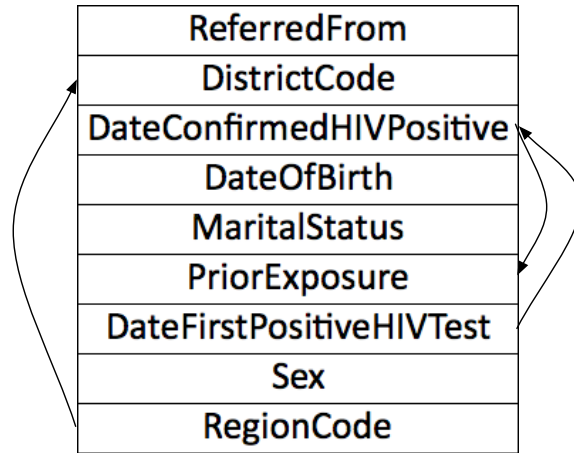


Figure 2.3. Example question layout generated by our ordering algorithm. The arrows reflect the probabilistic dependencies from Figure 2.1.

predicts how likely the response provided is erroneous, by examining whether it is likely to be a multivariate outlier, i.e., that it is unlikely with respect to the responses for other fields. In other words, an error probability is conditioned on all answered values provided by the data entry worker so far. If there are responses with error probabilities exceeding a pre-set threshold, USHER re-asks those questions ordered by techniques to be discussed in Section 2.6.

2.3.2 Implementation

We have implemented USHER as a web application (Figure 2.4). The UI loads a simple form specification file containing form question details and the location of the training data set. Form question details include question name, prompt, data type, widget type, and constraints. The server instantiates a model for each form. The system 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 [33], an open source Java package for structure learning of Bayesian networks. Our graphical model is implemented in two variants: the first model used for ordering is based on a modified version of JavaBayes [19], 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 [2], a Microsoft .NET Framework toolkit for Bayesian inference.

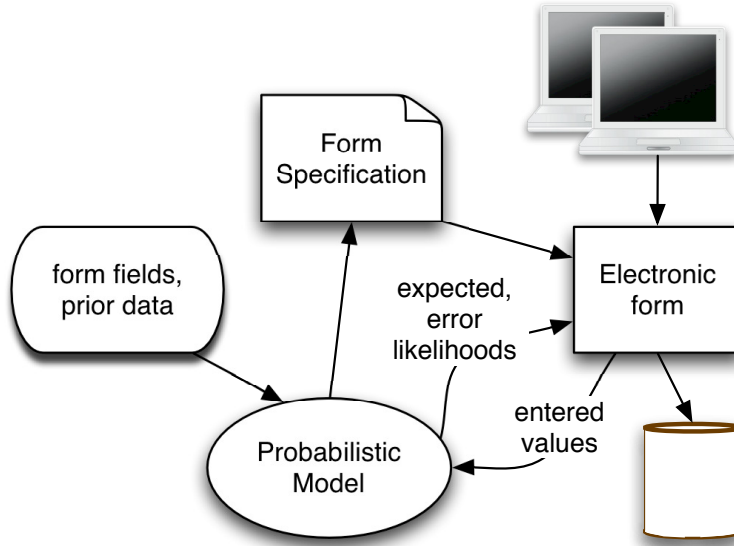


Figure 2.4. USHER components and data flow: (1) model a form and its data, (2) generate question ordering according to *greedy information gain*, (3) instantiate the form in a data entry interface, (4) during and immediately after data entry, provide dynamic re-ordering, feedback and re-confirmation according to *contextualized error likelihood*.

2.4 Learning a Model for Data Entry

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 a form’s 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 infer 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 $\mathbf{F} = \{F_1, \dots, F_n\}$ to denote a set of random variables representing the values of n questions comprising a data entry form. We assume that each question response takes on a finite set of discrete values; continuous values are discretized by dividing the data range into intervals and assigning each interval one value.² To learn 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 $\mathbf{G} \subseteq \mathbf{F}$ 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 independencies between

²Using richer distributions to model fields with continuous or ordinal answers (e.g., with Gaussian models) could provide additional improvement, and is left for future work.

the question random variables 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. USHER searches through the space of possible structures using simulated annealing, and chooses the best structure according to the Bayesian Dirichlet Equivalence criterion [35]. This criterion optimizes for a tradeoff between model expressiveness (using a richer dependency structure) and model parsimony (using a smaller number of parameters), thus identifying only the prominent, recurring probabilistic dependencies. Figures 2.1 and 2.2 show automatically learned structures for two data domains.³

In certain domains, form designers may already have strong common-sense notions of questions that should or should not depend on each other (e.g., education level and income are related, whereas gender and race are independent). As a postprocessing step, the form designer can manually tune the resulting model to incorporate such intuitions. In fact, the entire structure could be manually constructed in domains where an expert has comprehensive prior knowledge of the questions’ interdependencies. However, a casual form designer is unlikely to consider the complete space of question combinations when identifying correlations. In most settings, we believe an automatic approach to learning multivariate correlations would yield more effective inference.

Given a graphical structure of the questions, we can then estimate the *conditional probability tables* that parameterize 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 $\mathcal{P}(F_i)$ from the Bayesian network, is:

$$\begin{aligned} P(F_i = f_i \mid \{F_j = f_j : F_j \in \mathcal{P}(F_i)\}) \\ = \frac{N(F_i = f_i, \{F_j = f_j : F_j \in \mathcal{P}(F_i)\})}{N(\{F_j = f_j : F_j \in \mathcal{P}(F_i)\})}. \end{aligned} \tag{2.1}$$

In this notation, $P(F_i = f_i \mid \{F_j = f_j : F_j \in \mathcal{P}(F_i)\})$ refers to the conditional probability of question F_i taking value f_i , given that each question F_j in $\mathcal{P}(F_i)$ takes on value f_j . Here, $N(\mathbf{X})$ 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 $\mathcal{P}(F_i)$ 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 2.1 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

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

Input: Model \mathcal{G} with questions $\mathbf{F} = \{F_1, \dots, F_n\}$

Output: Ordering of questions $\mathbf{O} = (O_1, \dots, O_n)$

$\mathbf{O} \leftarrow \emptyset;$

while $|\mathbf{O}| < n$ **do**

$F \leftarrow \operatorname{argmax}_{F_i \notin \mathbf{O}} H(F_i \mid \mathbf{O});$
 $\mathbf{O} \leftarrow (\mathbf{O}, F);$

end

Algorithm 1: Static ordering algorithm for form layout.

estimates, interpolating each with a background uniform distribution. In particular, we revise our estimates to:

$$\begin{aligned} P(F_i = f_i \mid \{F_j = f_j : F_j \in \mathcal{P}(F_i)\}) \\ = (1 - \alpha) \frac{N(F_i = f_i, \{F_j = f_j : F_j \in \mathcal{P}(F_i)\})}{N(\{F_j = f_j : F_j \in \mathcal{P}(F_i)\})} + \frac{\alpha}{m}, \end{aligned} \quad (2.2)$$

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 [39].

Once the Bayesian network is constructed, we can infer distributions of the form $P(\mathbf{G} \mid \mathbf{G}' = \mathbf{g}')$ for arbitrary $\mathbf{G}, \mathbf{G}' \subseteq \mathbf{F}$ —that is, the *marginal* distributions over sets of random variables, optionally conditioned on observed values for other variables. Answering such queries is known as the *inference* task. There exist a variety of inference techniques. In our experiments, the Bayesian networks are small enough that exact techniques such as the *junction tree algorithm* [14] can be used. For larger models, faster approximate inference techniques like Loop Belief Propagation or Gibbs Sampling—common and effective approaches in Machine Learning—may be preferable.

2.5 Question Ordering

Having described the Bayesian network, we now turn to its applications in the USHER system. We first consider ways of automatically ordering the questions of a data entry form. 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. The key

idea behind our ordering algorithm is this *greedy information gain* — that is, to reduce the amount of *uncertainty* of a single form instance as quickly as possible. 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 from an entire form submission—is fixed. By reducing this uncertainty as early as possible, we can be more certain about the values of later questions. The benefits of greater certainty about later questions are two-fold. First, it allows us to more accurately provide data entry feedback for those questions. Second, we can more accurately predict missing values for incomplete form submissions.

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:

$$H(F_i) = - \sum_{f_i} P(f_i) \log P(f_i), \quad (2.3)$$

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 the 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 $\mathbf{G} = \{F_1, \dots, F_n\}$ have been previously encountered, with its *conditional entropy*:

$$\begin{aligned} H(F_i | \mathbf{G}) &= - \sum_{\mathbf{g}=(f_1, \dots, f_n)} \sum_{f_i} P(\mathbf{G} = \mathbf{g}, F_i = f_i) \log P(F_i = f_i | \mathbf{G} = \mathbf{g}), \end{aligned} \quad (2.4)$$

where the sum is over all possible question responses in the Cartesian product of F_1, \dots, 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 typically drop many terms from the conditioning in Equation 2.4 for faster computation.⁴

Our full *static ordering* algorithm based on greedy information gain is presented in Algorithm 1. We select the entire question ordering in a stepwise manner, starting with the first question. At the i th 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 the actual data entry session.

⁴Conditional entropy can also be expressed as the incremental difference in joint entropy due to F_i , that is, $H(F_i | \mathbf{G}) = H(F_i, \mathbf{G}) - H(\mathbf{G})$. Writing out the sum of entropies for an entire form using this expression yields a telescoping sum that reduces to the fixed value $H(\mathbf{F})$. Thus, this formulation confirms our previous intuition that no matter what ordering we select, the *total* amount of uncertainty is still the same.

In many scenarios the form designer would like 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:

$$\arg \max_{\mathbf{G}} H(\mathbf{G} \mid F_1, \dots, F_{i-1}), \quad (2.5)$$

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.

Form designers may also want to specify other kinds of constraints on form layout, such as a partial ordering over the questions that must be respected. The greedy approach can accommodate such constraints by restricting the choice of fields at every step to match the partial order.

2.5.1 Reordering Questions during Data Entry

In electronic form settings, we can take our ordering notion a step further and *dynamically reorder* questions in a form as an instance is being entered. This approach can be appropriate for scenarios when data entry workers input one or several values at a time, such as on a mobile device. We can apply the same greedy information gain criterion as in Algorithm 1, but update the calculations with the previous responses in the same form instance. Assuming questions $\mathbf{G} = \{F_1, \dots, F_\ell\}$ have already been filled in with values $\mathbf{g} = \{f_1, \dots, f_\ell\}$, the next question is selected by maximizing:

$$\begin{aligned} H(F_i \mid \mathbf{G} = \mathbf{g}) \\ = - \sum_{f_i} P(F_i = f_i \mid \mathbf{G} = \mathbf{g}) \log P(F_i = f_i \mid \mathbf{G} = \mathbf{g}). \end{aligned} \quad (2.6)$$

Notice that this objective is the same as Equation 2.4, except that it uses the actual responses entered for 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 and potential information gain of the latter. However, dynamic reordering may be confusing to data entry workers 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 a human factors issue that depends on both the application domain and the user interface employed.

2.6 Question Re-asking

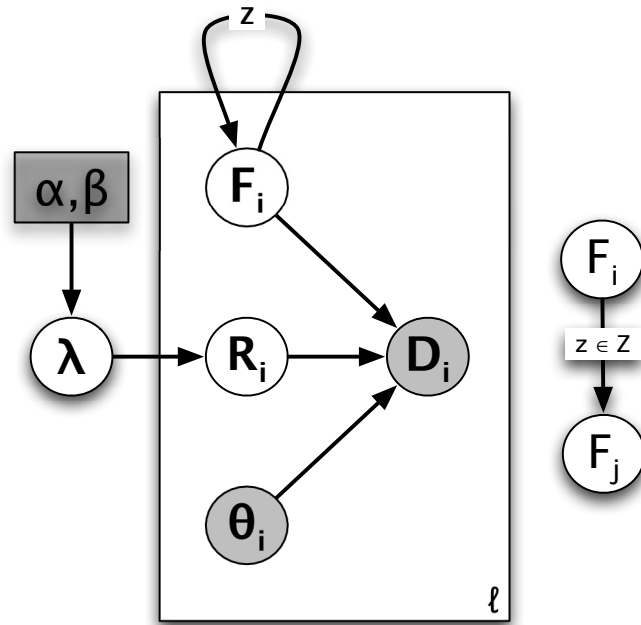


Figure 2.5. The error model. Observed variable D_i represents the actual input provided by the data entry worker for the i th question, while hidden variable F_i is the true value of that question. The rectangular plate around the center variables denotes that those variables are repeated for each of the ℓ form questions with responses that have already been input. The F variables are connected by edges $z \in Z$, representing the relationships discovered in the structure learning process; this is the same structure used for the question ordering component. Variable θ_i represents the “error” distribution, which in our current model is uniform over all possible values. Variable R_i is a hidden binary indicator variable specifying whether the entered data was erroneous; its probability λ_i is drawn from a Beta prior with fixed hyperparameters α and β . Shaded nodes denote observed variables, and clear nodes denote hidden variables.

The next application of USHER’s probabilistic model is for the purpose of identifying *errors* made during entry. Because this determination is made during and immediately after form submission, USHER can choose to *re-ask* questions during the same entry session. By focusing the re-asking effort only on questions that were likely to be misentered, USHER is likely to catch mistakes at a small incremental cost to the data entry worker. Our approach is a data-driven alternative to the expensive practice of double entry. Rather than re-asking *every* question, we focus re-asking effort only on question responses that are unlikely with respect to the other form responses.

Before exploring how USHER performs re-asking, we explain how it determines whether a question response is erroneous. USHER estimates *contextualized error likelihood* for each question response, i.e., a probability of error that is conditioned on every other previously entered field response. The intuition behind error detection is straightfor-

ward: questions whose responses are “unexpected” with respect to the rest of the known input responses are more likely to be incorrect. These error likelihoods are measured both during and after the entry of a single form instance.

2.6.1 Error Model

To formally model the notion of error, we extend our Bayesian network from Section 2.4 to a more sophisticated representation that ties together intended and actual question responses. We call the Bayesian network augmented with these additional random variables the *error model*. Specifically, we posit a network where each question is augmented with additional nodes to capture a probabilistic view of entry error. For question i , we have the following set of random and observed 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 entry worker, an *observed* variable.
- θ_i : the observed variable representing the parameters of the *probability distribution* of mistakes across possible answers, which is fixed per question.⁵ We call the distribution with parameters θ_i the *error distribution*. For the current version of our model, θ_i is set to yield a uniform distribution.
- R_i : a binary hidden variable specifying whether an error was made in this question. When $R_i = 0$ (i.e., when no error is made), then F_i takes the same value as D_i .

Additionally, we introduce a hidden variable λ , shared across all questions, specifying how likely errors are to occur for a typical question of that form instance. Intuitively, λ plays the role of a prior error rate, and is modeled as a hidden variable so that its value can be learned directly from the data.

Note that the relationships between field values discovered during structure learning are still part of the graph, so that the error predictions are contextualized in the answers of other related questions.

Within an individual question, the relationships between the newly introduced variables are shown in Figure 2.5. The diagram follows standard plate diagram notation [17]. In brief, the rectangle is a *plate* containing a group of variables specific to a single question i . This rectangle is replicated for each of ℓ form questions. The F variables in each question group are connected by edges $z \in Z$, representing the relationships discovered in the structure learning process; this is the same structure used for the question ordering component. The remaining edges represent direct probabilistic relationships between the variables that

⁵Note that in a hierarchical Bayesian formulation such as ours, random variables can represent not just specific values but also parameters of distributions. Here, θ_i is the parameters of the error distribution.

are described in greater detail below. Shaded nodes denote observed variables, and clear nodes denote hidden variables.

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 entry worker implicitly flips a coin for R_i when entering a response for question i , with probability of one equal to λ . Formally, this means R_i is drawn from a Bernoulli distribution with parameter λ :

$$R_i \mid \lambda \sim \text{Bernoulli}(\lambda) \quad (2.7)$$

The value of R_i affects how F_i and D_i are related, which is described in detail later in this section.

We also allow the model to learn the prior probability for the λ directly from the data. This value represents the probability of making a mistake on any arbitrary question. Note that λ is shared across all form questions. Learning a value for λ rather than fixing it allows the model to produce an overall probability of error for an entire form instance as well as for individual questions. The prior distribution for λ is a *Beta* distribution, which is a continuous distribution over the real numbers from zero to one:

$$\lambda \sim \text{Beta}(\alpha, \beta) \quad (2.8)$$

The Beta distribution takes two hyperparameters α and β , which we set to fixed constants (1, 19). The use of a Beta prior distribution for a Bernoulli random variable is standard practice in Bayesian modeling due to mathematical convenience and the interpretability of the hyperparameters as effective counts [13].

We now turn to true question value F_i and observed input D_i . First, $P(F_i \mid \dots)$ is still defined as in Section 2.4, maintaining as before the multivariate relationships between questions. Second, the user question response D_i is modeled as being drawn from either the true answer F_i or the error distribution θ_i , depending on whether a mistake is made according to R_i :

$$D_i \mid F_i, \theta_i, R_i \sim \begin{cases} \text{PointMass}(F_i) & \text{if } R_i = 0, \\ \text{Discrete}(\theta_i) & \text{otherwise,} \end{cases} \quad (2.9)$$

If $R_i = 0$, no error occurs and the data entry worker inputs the correct value for D_i , and thus $F_i = D_i$. Probabilistically, this means D_i 's probability is concentrated around F_i (i.e., a point mass at F_i). However, if $R_i = 1$, then the data entry worker makes a mistake, and instead chooses a response for the question from the error distribution. Again, this error distribution is a discrete distribution over possible question responses parameterized by the fixed parameters θ_i , which we set to be the uniform distribution in our current model.⁶

⁶A more precise error distribution would allow the model to be especially wary of common mistakes. However, learning such a distribution is itself a large undertaking involving carefully designed user studies with a variety of input widgets, form layouts, and other interface variations, and a post-hoc labeling of data for errors. This is another area for future work.

2.6.2 Error Model Inference

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

$$P(R_i \mid \mathbf{D} = \mathbf{d}), \quad (2.10)$$

where $\mathbf{D} = \{F_1, \dots, F_\ell\}$ are the fields that currently have responses, the values of which are $\mathbf{d} = \{f_1, \dots, f_\ell\}$ respectively. This probability represents a contextualized error likelihood due to its dependence on other field values through the Bayesian network.

Again, we can use standard Bayesian inference procedures to compute this probability. These procedures are black-box algorithms whose technical descriptions are beyond the scope of this thesis. We refer the reader to standard graphical model texts for an in-depth review of different techniques [14, 45]. In our implementation, we use the Infer.NET toolkit [2] with the Expectation Propagation algorithm [54] for this estimation.

2.6.3 Deciding when to Re-ask

Once we have inferred a probability of error for each question, we can choose to perform re-asking either during entry, where the error model is consulted after each response, or after entry, where the error model is consulted once with all form responses, or both.

The advantage of integrating re-asking into the primary entry process is that errors can be caught as they occur, when the particular question being entered is still recent in the data entry worker’s attention. The cognitive cost of re-entering a response at this point is lower than if that question were re-asked later. However, these error likelihood calculations are made on the basis of incomplete information—a question response may at first appear erroneous *per se*, but when viewed in the context of concordant responses from the same instance may appear less suspicious. In contrast, by batching re-asking at the end of a form instance, USHER can make a holistic judgment about the error likelihoods of each response, improving its ability to estimate error. This tradeoff reveals an underlying human-computer interaction question about how recency affects ease and accuracy of question response, a full exploration of which is beyond the scope of this thesis.

To decide whether to re-ask an individual question, we need to consider the tradeoff between improved data quality and the cost of additional time required for re-asking. USHER allows the form designer to set an error probability threshold for re-asking. When that threshold is exceeded for a question, USHER will re-ask that question, up to a predefined budget of re-asks for the entire form instance. If the response to the re-asked question differs from the original response, the question is flagged for further manual reconciliation, as in double entry. For in-flight re-asking, we must also choose either the original or re-asked response for this field for further predictions of error probabilities in the same form instance. We choose the value that has the lesser error probability, since in the absence of

further disambiguating information that value is more likely to be correct according to the model.

2.7 Question Reformulation

1. How did you come to the clinic?

- Ambulance
- Bicycle
- Bus (Daladala)
- Car taxi (Special hire)
- Foot
- Motorcycle taxi (Bodaboda)
- Private vehicle
- Other

2. How did you come to the clinic?

- Foot
- <another answer>

Figure 2.6. An example of a reformulated question.

Our next application of USHER’s probabilistic formalism is *question reformulation*. To reformulate a question means to simplify it in a way that reduces the chances of the data entry worker making an error. Figure 2.6 presents an example of a question as originally presented and a reformulated version thereof. Assuming that the correct response is *Foot*, the data entry worker would only need to select it out of two rather than eight choices, reducing the chance of making a mistake. Moreover, reformulation can enable streamlining of the input interface, improving data entry throughput.

In this work we consider a constrained range of reformulation types, emphasizing an exploration of the *decision* to reformulate rather than the interface details of the reformulation itself. Specifically, our target reformulations are binary questions confirming whether a particular response is the correct response, such as in the example. If the response to the

binary question is negative, then we ask again the original, non-reformulated question. We emphasize that the same basic data-driven approach described here can be applied to more complex types of reformulation, for instance, formulating to k values where $k < D$, the size of the original answer domain.

The benefits of question reformulation rely on the observation that different ways of presenting a question will result in different error rates. This assumption is borne out by research in the HCI literature. For example, in recent work Wobbrock et al. [82] showed that users' chances of clicking an incorrect location increases with how far and small the target location is. As a data entry worker is presented with more options to select from, they will tend to make more mistakes. We also note that it takes less time for a data entry worker to select from fewer rather than more choices, due to reduced cognitive load and shorter movement distances. These findings match our intuitions about question formulation—as complexity increases, response time and errors increase as well.

However, question reformulation comes with a price as well. Since reformulating a question reduces the number of possible responses presented to the data entry worker, it is impossible for the reformulated question to capture all possible legitimate responses. In the example above, if *Foot* was not the correct response, then the original question would have to be asked again to reach a conclusive answer. Thus, the decision to reformulate should rest on how *confident* we are about getting the reformulation “right”—in other words, the probability of the most likely answer as predicted by a probabilistic mechanism.

In light of the need for a probabilistic understanding of the responses, USHER's reformulation decisions are driven by the underlying Bayesian network. We consider reformulation in three separate contexts: *static reformulation*, which occurs during the form layout process; *dynamic reformulation*, which occurs while responses are being entered; and *post-entry reformulation*, which is applied in conjunction with re-asking to provide another form of cross-validation for question responses.

2.7.1 Static Reformulation

In the static case, we decide during the design of a form layout which questions to reformulate, if any, in conjunction with the question ordering prediction from Section 2.5. This form of reformulation simplifies questions that tend to have predominant responses across previous form instances. Static reformulation is primarily appropriate for situations when forms are printed on paper and question ordering is fixed. Standard practice in form design is to include *skip-logic*, a notation to skip the full-version of the question should the answer to the reformulated question be true. Alternatively, false responses to reformulated questions can be compiled and subsequently re-asked after the standard form instance is completed.

For each question, we decide whether to reformulate on the basis of the probability of its *expected response*. If that response exceeds a tunable threshold, then we choose to

reformulate the question into its binary form. Formally, we reformulate when

$$\max_j P(F_i = f_j) \geq T_i, \quad (2.11)$$

where T_i is the threshold for question i . In this work we consider values of T_i that are fixed for an entire form, though in general it could be adjusted on the basis of the original question’s complexity or susceptibility to erroneous responses. We note that this reformulation mechanism is directly applicable for questions with discrete answers, either categorical (e.g., *blood-type*) or ordinal (e.g., *age*); truly continuous values (e.g., *weight*) must be discretized before reformulation. However, continuous questions with large answer domain cardinalities are less likely to trigger reformulation, especially if their probability distributions are fairly uniform.

Setting the threshold T provides a mechanism for trading off improvements in data quality with the potential drawback of having to re-ask more questions. At one extreme, we can choose to never reformulate; at the other, if we set a low threshold we would provide simplified versions of every question, at the cost of doubling the number of questions asked in the worst-case.

2.7.2 Dynamic Reformulation

Paralleling the dynamic approach we developed for question ordering (Section 2.5.1), we can also decide to reformulate questions during form entry, making the decision based on previous responses. The advantage of dynamic reformulation is that it has the flexibility to change a question based on context—as a simple example, conditioned on the answer for an *age* question being 12, we may choose to reformulate a question about *occupation* into a binary *is-student* question. Dynamic reformulation is appropriate in many electronic, non-paper based work-flows. In this case, the reformulation decision is based on a *conditional expected response*; for question i we reformulate when

$$\max_j P(F_i = f_j \mid \mathbf{G} = \mathbf{g}) \geq T_i, \quad (2.12)$$

where previous questions $\mathbf{G} = \{F_1, \dots, F_\ell\}$ have already been filled in with values $\mathbf{g} = \{f_1, \dots, f_\ell\}$. Note the similarities in how the objective function is modified for both ordering and reformulation (compare equations 2.11 and 2.12 to equations 2.4 and 2.6).

2.7.3 Reformulation for Re-asking

Finally, another application of reformulation is for re-asking questions. As discussed in Section 2.6, the purpose of re-asking is to identify when a response may be in error, either during or after the primary entry of a form instance. One way of reducing the overhead associated with re-asking is to simplify the re-asked questions. Observe that a re-ask question does not have to elicit the true answer, but rather a corroborating answer. For example,

for the question *age*, a reformulated re-ask question could be the discretization bucket in which the age falls (e.g., *21–30*). From the traditional data quality assurance perspective, this technique enables dynamic cross-validation questions based on contextualized error likelihood.

The actual mechanics of the reformulation process are the same as before. Unlike the other applications of reformulation, however, here we have an answer for which we can compute error likelihood.

2.8 Evaluation

We evaluated the benefits of USHER by simulating two data entry scenarios to show how our system can improve data quality. We focused our evaluation on the quality of our model and its predictions. We set up experiments to measure our models’ ability to predict users’ intended answers, to catch artificially injected errors, and to reduce error using reformulated questions. We first describe the experimental data sets, and then present our simulation experiments and results.

2.8.1 Data Sets and Experimental Setup

We examine the benefits of USHER’s design using two data sets, previously described in Section 2.3. The *survey* data set comprises responses from a 1986 poll about race and politics in the San Francisco-Oakland metropolitan area [8]. The UC Berkeley Survey Research Center interviewed 1,113 persons by random-digit telephone dialing. The *patient* data set was collected from anonymized patient intake records at a rural HIV/AIDS clinic in Tanzania. In total we had fifteen questions for the survey and nine for the patient data. We 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, split 80% to 20%, respectively. For the survey, we had 891 training instances and 222 test; for patients, 1,320 training and 330 test. We performed structure learning and parameter estimation using the training set. As described in Section 2.4, this resulted in the graphical models shown in Figures 2.1 and 2.2. The test portion of each dataset was then used for the data entry scenarios presented below.

In our simulation experiments, we aim to verify hypotheses regarding three components of our system: first, that our data-driven question orderings ask the most uncertain questions first, improving our ability to predict missing responses; second, that our re-asking model is able to identify erroneous responses accurately, so that we can target those questions for verification; and third, that question reformulation is an effective mechanism for trading off between improved data quality and user effort.

2.8.2 Ordering

For the ordering experiment, we posit a scenario where the data entry worker is interrupted while entering a form submission, and thus is unable 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 pre-computed static ordering, USHER’s dynamic ordering (where the order can be adjusted 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 (mode) of the probability distribution over un-entered questions, given the known responses. Results are averaged over each instance in the test set.

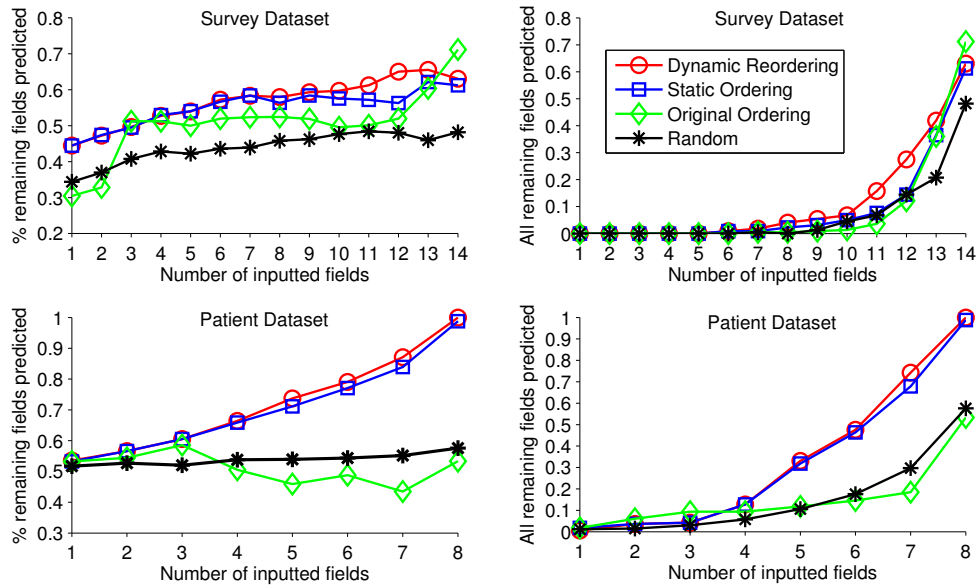


Figure 2.7. Results of the ordering simulation experiment. In each case, the x -axis measures how many questions are filled before the submission is truncated. In the charts on the left side, the y -axis plots the average proportion of remaining question whose responses are predicted correctly. In the charts on the right side, 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.

The left-hand graphs of Figure 2.7 measure the average number of correctly predicted unfilled questions, as a function of how many responses the data entry worker entered 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 2.7.

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 does not optimize for greedy information gain.

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.

Re-asking

For the re-asking experiment, our hypothetical scenario is one where the data entry worker enters a complete form instance, but with erroneous values for some question responses.⁷ Specifically, we assume that for each data value the data entry worker 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 2.8 plots the percentage of instances where we chose 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 is observed in the field [63], 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 able to achieve the same result of identifying all erroneous responses at a substantially reduced cost.

Reformulation

For the reformulation experiment, we simulate form filling with a background error rate and time cost in order to evaluate the impact of reformulated questions. During simulated entry, when a possible response a is at the mode position of the conditional probability distribution and has a likelihood greater than a threshold t , we ask whether the answer is

⁷Our experiments here do not include simulations with in-flight re-asking, as quantifying the benefit of re-asking during entry—question recency—is a substantial human factors research question that is left as future work.

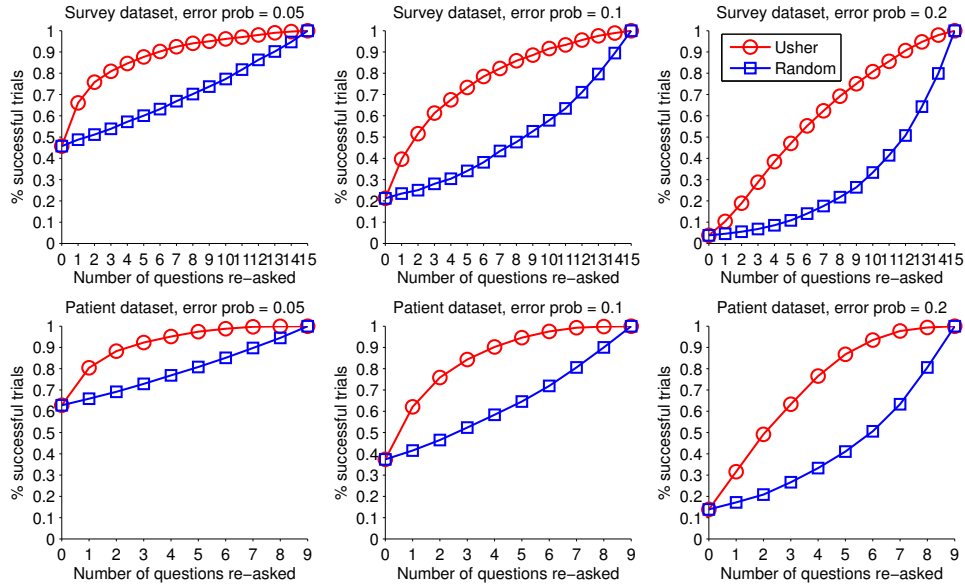


Figure 2.8. Results of the re-asking simulation experiment. In each case, the x -axis measures how many questions we are allowed to re-ask, and the y -axis measures whether we correctly identify all erroneous questions within that number of re-asks. The error probability indicates the rate at which we simulate errors in the original data. Results for the survey data are shown at top, and for the HIV/AIDS data at bottom.

a as a reformulated binary question. If a is not the true answer, we must re-ask the full question. Results are averaged over each instance in the test set.

Before discussing these results, we motivate the choice of error rates and cost functions that we employ in this experiment. As mentioned in Section 2.7, the intuition behind question reformulation is grounded in prior literature, specifically the notion that simpler questions enjoy both lower error rate and user effort. However, the downside with reformulation is that entry forms may cost more to complete, due to reformulated questions with negative responses.

In order to bootstrap this experiment, we need to derive a representative set of error rates and entry costs that vary with the complexity of a question. Previous work [82, 26] has shown that both entry time and error rate increase as a function of interface complexity. In particular, Fitts' Law [26] describes the time complexity of interface usage via an *index of complexity* (ID), measured in *bits* as $\log(A/W + 1)$. This is a logarithmic function of the ratio between target size W and target distance A . Mapping this to some typical data entry widgets such as radio-buttons and drop-down menus, where W is fixed and A increases linearly with the number of selections, we model time cost as $\log(D)$ where D is the domain cardinality of the answer. In other words, time cost grows with how many bits it takes to encode the answer. For our experiments, we set the endpoints at 2 seconds for $D = 2$ up to 4 seconds for $D = 128$.

We also increase error probabilities logarithmically as a function of domain cardinality

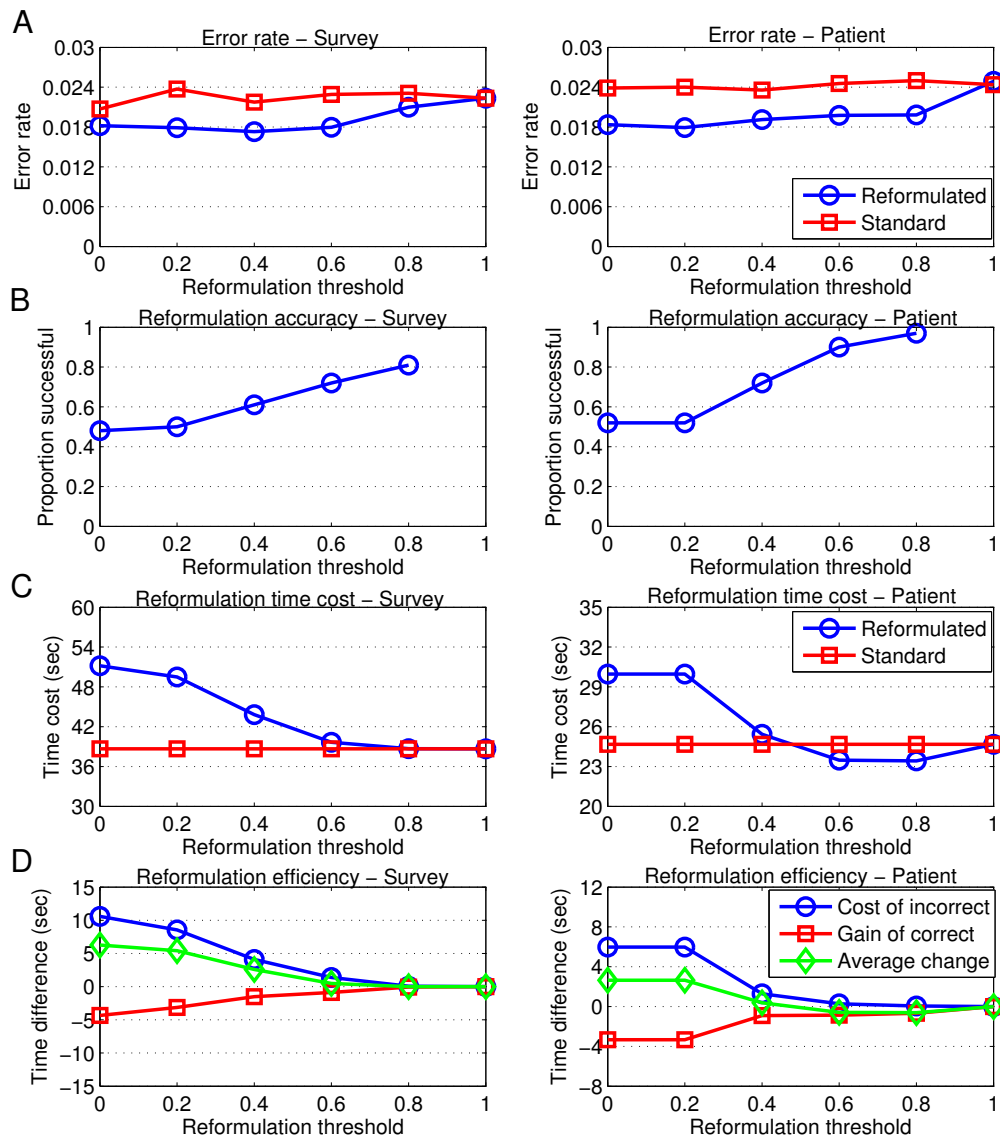


Figure 2.9. Results of the question reformulation experiment. In each chart, the x -axis shows the reformulation thresholds; when the threshold = 1, no question is reformulated. *A* shows the overall error rate between reformulated and standard entry. *B* shows the likelihood that a reformulated answer was, in fact, the correct answer. *C* shows the impact of reformulation on user effort, measured as time—average number of seconds per form. *D* shows the gain/loss in effort due to when reformulation is correct vs incorrect.

D , relying on the intuition that error will also tend to increase as complexity increases [82]. Our error rates vary from 1% for $D = 2$ to 5% for $D = 128$.

We do not claim the generalizability of these specific numbers, which are derived from a set of strong assumptions. Rather, the values we have selected are representative for studying the general trends of the tradeoff between data quality and cost that re-asking enables, and are in line with typical values observed in the field [63]. Furthermore, we attempted this experiment with other error and cost parameters and found similar results.

The results of question reformulation can be found in Figure 2.9. In the pair of graphs entitled A, we measure the error rate over reformulation thresholds for each dataset. Our results confirm the hypothesis that the greater the number of additional reformulated questions we ask, the lower the error rate. In the pair of graphs entitled B, we observe that as the selectivity (threshold) of reformulation goes up, the likelihood that we pick the correct answer in reformulation also rises. Observe that reformulation accuracy is greater than 80% and 95% for the *survey* and *patient* datasets, respectively, at a threshold of 0.8. In the pair of graphs entitled C, we see an unexpected result: entry with reformulation features a time cost that quickly converges with, and in the case of the *patient* dataset, dips below that of standard entry, at thresholds beyond 0.6. Finally, in the pair of graphs entitled D, we summarize the time cost incurred by additional questions versus the time savings of the simpler reformulated questions. Of course, given our assumptions, we cannot make a strong conclusion about the cost of question reformulation. Rather, the important takeaway is that the decrease in effort won by correct reformulations can help to offset the increase due to incorrect reformulations.

2.9 Conclusion and Future Work

In this chapter, we have shown that a probabilistic approach can be used to design intelligent data entry forms that result in high data quality. USHER leverages data-driven insights to automate multiple steps in the data entry pipeline. Before entry, we find an ordering of form fields that promotes rapid information capture, driven by the greedy information gain principle, and can statically reformulate questions to promote more accurate responses. During entry, we dynamically adapt the form based on entered values, facilitating re-asking, reformulation, and real-time interface feedback in the spirit of providing appropriate entry friction. After entry, we automatically identify possibly-erroneous inputs, guided by contextualized error likelihoods, and re-ask those questions, possibly reformulated, to verify their correctness. Our simulated empirical evaluations demonstrate the data quality benefits of each of these components: question ordering, reformulation and re-asking.

The USHER system we have presented combines several disparate approaches to improving data quality for data entry. The three major components of the system—ordering, re-asking, and reformulation—can all be applied under various guises before, during, and after data entry. This suggests a principled roadmap for future research in data entry. For

example, one combination we have not explored here is re-asking before entry. At first glance this may appear strange, but in fact that is essentially the role that cross-validation questions in paper forms serve—pre-emptive, reformulated, re-asked questions. Translating such static cross-validation questions to dynamic forms is a potential direction of future work.

This work can be extended by enriching the underlying probabilistic formalism. Our current probabilistic approach assumes that every question is discrete and takes on a series of unrelated values. Relaxing these assumptions would make for a potentially more accurate predictive model for many domains. Additionally, it is worthwhile 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. On that front, an interesting line of future work would be to learn a model of data entry errors and adapt our system to catch them.

We can adapt this approach to the related problem of conducting on-line surveys. To do so, 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. An intriguing possibility is to use USHER’s predictive model to detect problematic user behaviors, including detecting user fatigue and *satisficing*, where a respondent does just enough to satisfy form requirements, but nothing more [32]. 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.

Chapter 3

Adaptive Feedback

3.1 Introduction

In this chapter, we use USHER’s predictive ability to design a number of intelligent user interface adaptations that can directly improve data entry accuracy and efficiency during the act of entry. 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% – +6%.

The specific adaptations we tested include: 1) setting defaults corresponding to highly-likely answers, 2) dynamically re-ordering 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 chapter is organized as follows. In the next section, we discuss related work, and after that we recall the relevant parts of USHER. In the section after, we provide a cognitive model of data entry based on our contextual inquiry, and after that we use the model to motivate a number of specific intelligent user interface adaptations. Finally, we present the experimental setup for our evaluation, and describe the results.

3.2 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.

3.2.1 Managing Data Quality During Entry

A best-practice for data quality during entry is to use pre-determined constraints to reject or warn the user when they enter illegal or unlikely values [76]. 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* [36] 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 [36]. 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.

3.2.2 Improving Data Entry Efficiency

Cockburn *et al.* modeled the performance of menu interfaces as control mechanisms [18]. They offer insight on the transition from novice to expert behavior. We focus on a wider class of input interfaces for expert entry.

Recall that we referenced several approaches that use learning techniques to automatically fill or predict a top-k set of likely values [11, 37, 83], in the previous chapter. In addition, there have been several efforts to improve data input with modeling and interface adaptation [46, 70, 77, 79, 80]. Most of these have only provided simulation results of predictive accuracy and improvements in efficiency.

The work by Warren *et al* is relevant to our domain. They worked with medical doctors entering diagnosis information into an electronic medical record [77, 79, 80]. They trained models on drugs and diagnoses to automatically populate a “hotlist” of potential drug choices for the provider. Their models rely only on bi-variate relationships between a drug and a single diagnosis.

Again, 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.

3.2.3 Adaptive User Interfaces

The literature on adaptive user interfaces discusses feedback mechanisms' behavioral predictability, cognitive complexity, cost of being wrong, predictive accuracy, and users' ability to opt-out of an adaptation [29]. 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 click-able area for more likely options [25, 58, 73, 81].

3.3 A Data-driven Foundation For Adaptive Forms

USHER serves as a data-driven foundation for adapting data entry forms. Here, we summarize the aspects relevant to this chapter.

Recall the goal of an USHER model: given a subset of answers for a form, accurately predict values for the unanswered questions. As shown in Figure 3.1, 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 based on prior data. Learning these relationships for an arbitrary form is the first step for building this model. In Figure 3.1, we can see the Bayesian network generated for the data set that we used for the evaluation described later in this chapter. 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. Using this model, we can then infer a probability distribution for unanswered form questions given answered ones. Improving the predictive accuracy of the model allows for more effective feedback during entry.

3.4 Opportunities For Adaptation During Entry

Figure 3.2 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.

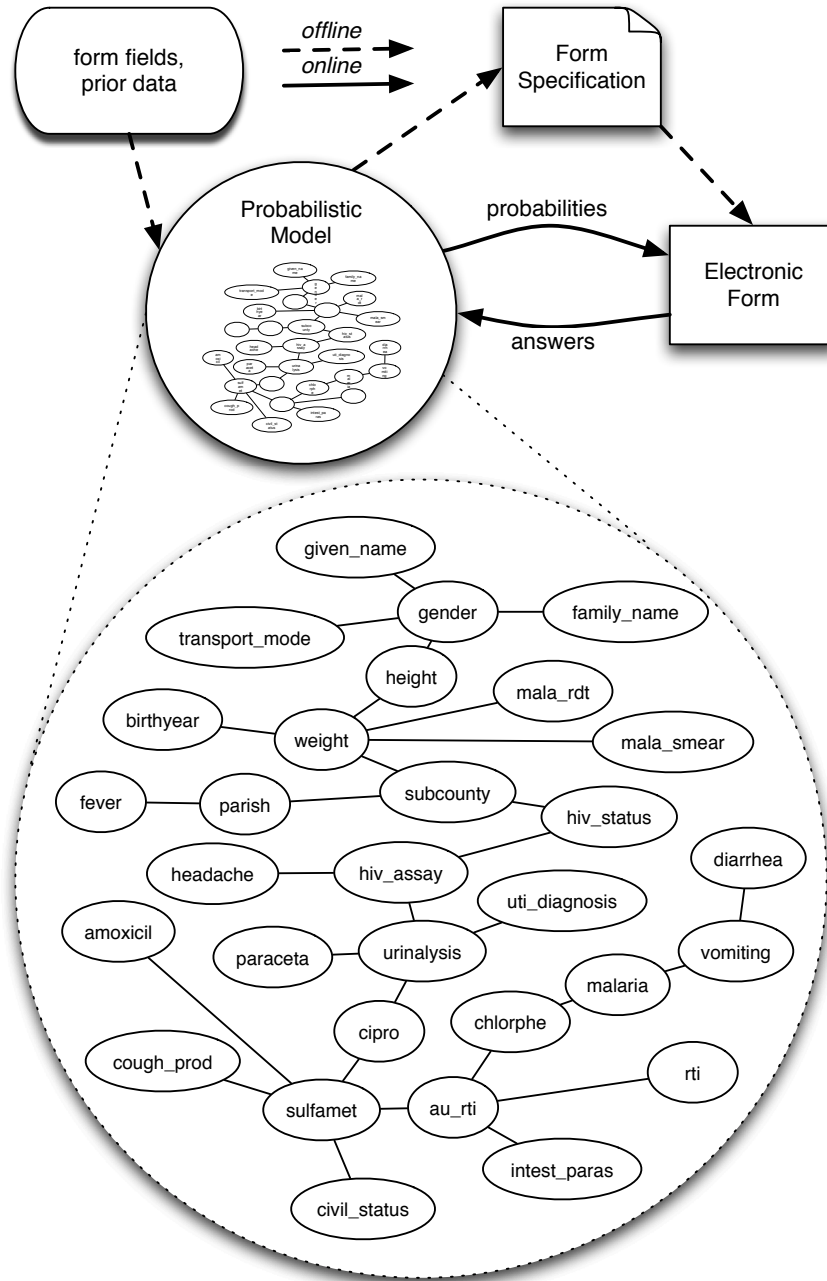


Figure 3.1. USHER components, data flow and probabilistic model: Above are USHER system components and data artifacts, with arrows showing data flow; zoom circle shows the Bayesian network learned from data.

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

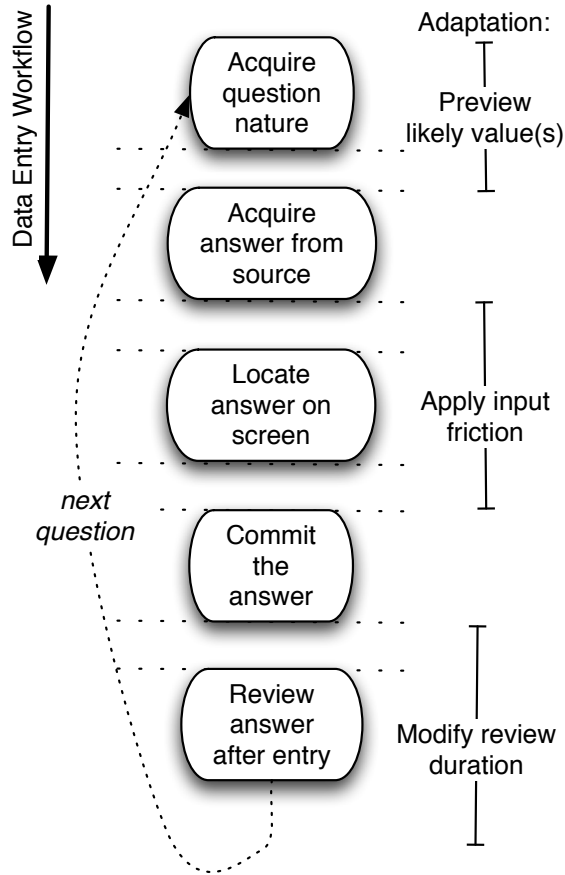


Figure 3.2. A cognitive task model for data entry. The vertical spans, on the right, show opportunities for dynamic adaptation.

- 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 chapter, 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 the 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.

3.5 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.

3.5.1 Design Process



Figure 3.3. Christine and Solomon, two professional data entry workers, and 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.

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



Figure 3.4. Alternative designs for radio button feedback: (1) radio buttons with bar-chart overlay. (2) radio buttons with scaled labels.

- 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 3.4). 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 [24].
- 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 3.4. 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 [29].
- Feedback should not depend on timed onset. Timed interaction techniques like ephemeral adaptation [25] 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 [28].

3.5.2 Feedback Mechanisms

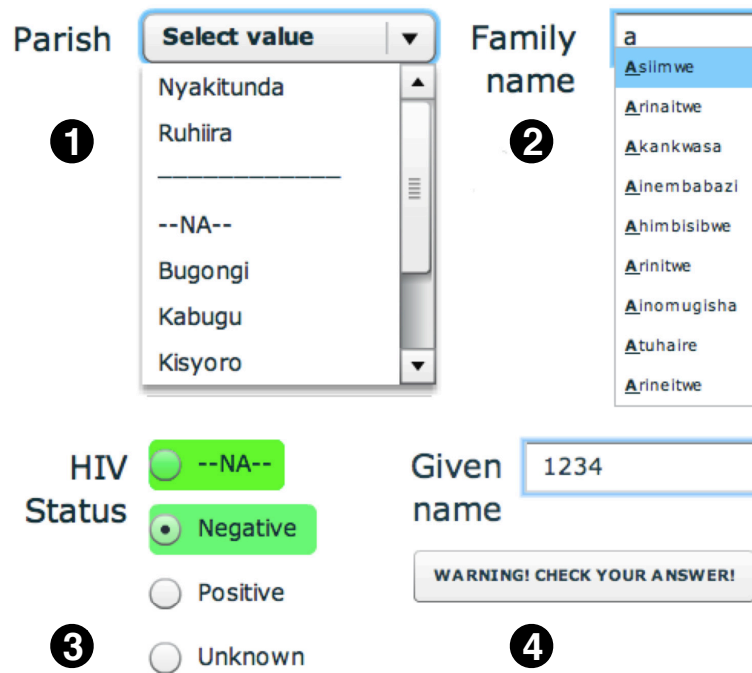


Figure 3.5. (1) drop down split-menu promotes the most likely items (2) text field ranks autocomplete suggestions by likelihood (3) radio buttons highlights promote most likely labels (4) warning message appears when an answer is a multivariate outlier.

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 3.5.

- *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 3.5(2), “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 3.5(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 3.4), we decided to simply scale the opacity of the highlights according to a \log_2 -scale [55].

- *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.

3.5.3 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 ± 2) of distinct items [53, 55]. When this number is exceeded, the potential for user error increases [50]. 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.

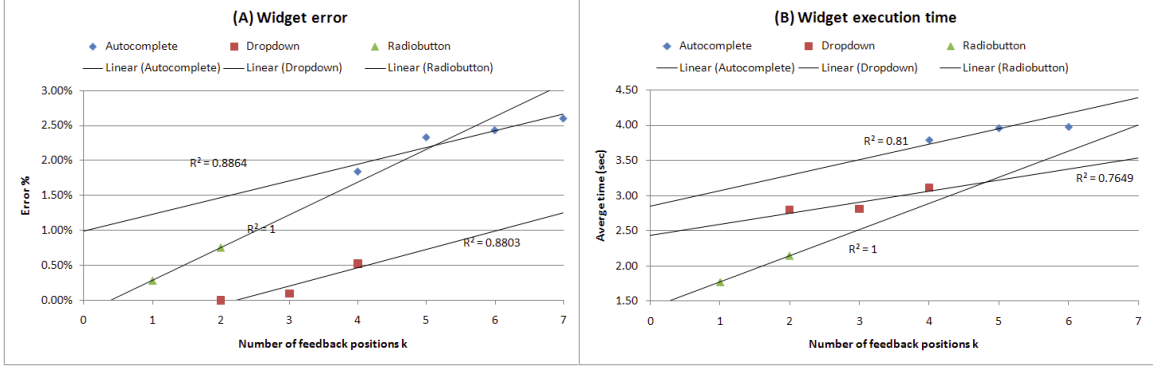


Figure 3.6. (A) linear interpolations of error rate for each widget-type. (B) linear interpolations of average time spent per question (seconds) for each widget-type.

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:

$$k = \min(7, \text{ceiling}(\log_b(D)), b > 1$$

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

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

3.6 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.

3.6.1 Context and Participants

To conduct this research, we collaborated with a health facility in a village called Ruhira, in rural Uganda. Ruhira is actively supported by the Millennium Villages Project [4] (MVP). MVP conducts multi-pronged interventions in health, agriculture, education and infrastructure to reduce poverty in Sub-Saharan Africa. The MVP health team in Ruhira (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 when obtaining the insights described in prior sections. Prior to the study, each of them became proficient with our electronic forms interface.

3.6.2 Forms and Data

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

Table 3.2. “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 3.1. More details about the data set can be found in Table 3.2.

3.6.3 An USHER Model for Patient Visits

The zoom circle in Figure 3.1 graphically depicts the USHER model resulting from structure learning on this *patient visit* dataset. The edges denote correlations between pairs of variables.

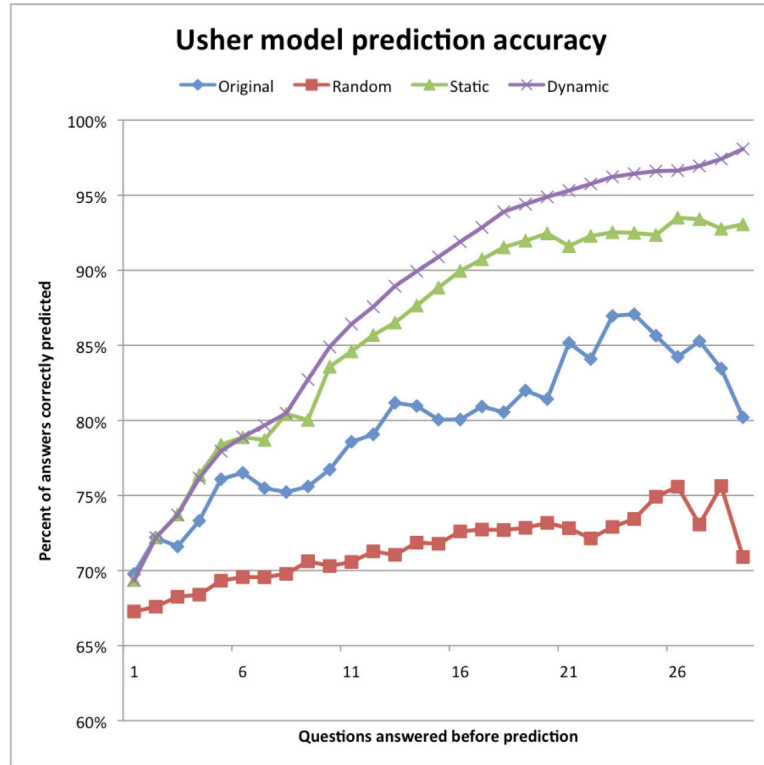


Figure 3.7. Results of the ordering experiment: x-axis measures the number of questions “entered”; y-axis plots the percentage of remaining answers correctly predicted.

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. As in the previous chapter, the x-axis in Figure 3.7 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.

3.6.4 System

We built a web application using Java Servlets and Adobe Flex (see Figure 3.1). 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.

3.6.5 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.

3.6.6 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*, and *warnings*. We added a control variation, *plain* with 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 than 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.

3.7 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.

3.7.1 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 3.3) and by each widget type (Table 3.4). Each error rate shown was estimated by its least square mean.

Table 3.3 shows the error rates of each feedback mechanism, and compares 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.

Breaking down the results by widget type leads to more statistically significant findings. Table 3.4 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 %.

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

Table 3.3. Mean error rates for each feedback variation (across all widget types) and comparisons to the *plain* control-variation.

Widget type	Feedback Type	Error rate	vs. <i>plain</i>	Adj. p-value	# wrong / # total
radio button	<i>plain</i>	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	<i>plain</i>	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	<i>plain</i>	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 3.4. Mean error rates for each widget type with break down by feedback type; as well, comparisons to the *plain* control-variation.

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 3.2 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.

3.7.2 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 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

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

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 3.5.

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

Table 3.5. 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 than 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, they 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.

²One reason not to provide more frequent feedback is for maintaining the user’s trust.

3.7.3 Effort

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

Table 3.6. 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 3.6. 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 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.

3.8 Discussion

3.8.1 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 [9]. 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 million³ 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*.

³350 million \times $\frac{1}{2}$ visit clinic \times 1.04% error rate

3.8.2 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 [21, 42]. 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 [67]. 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.

3.8.3 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 mobile entry settings, we can also use USHER *dynamic* orderings to further improve the predictive accuracy of the system.

3.8.4 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.

3.9 Conclusion and Future Work

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 Ruhira, Uganda. Our results show that these mechanisms can significantly improve data entry accuracy and efficiency.

A next step is to develop a version of USHER 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 [1, 3, 5] can benefit from this approach.

Chapter 4

Shreddr

4.1 Introduction

The SHREDDR system transforms paper form images into structured data on-demand. It combines batch processing and compression techniques from database systems, automatic document processing with computer vision, and value verification with crowd-sourced workers.

The approach redesigns data entry as an always-on software service in order to address the critical bottlenecks in existing solutions. First, SHREDDR extracts a paper form's schema and data locations via a web interface. Next, it uses computer vision to align then break up images into constituent image *shreds*. Then, the system reformulates the grouping of shreds according to an entropy-based metric, and places the shred groups into *batched* worker interfaces. Lastly, the system assigns work to distributed on-line workers whom iteratively refine shred estimates into final values.

SHREDDR is novel in several ways.

1. It allows approximate automation to simplify a large portion of data entry tasks.
2. Working with shreds in the cloud, and a crowd, gives us latitude to control latency and quality at a per-question granularity. For instance, time-sensitive answers can be prioritized, and important answers can be double- or triple-checked.
3. Shredded worker interfaces use cognitive and physical interface *compression* to enable fast, batched data entry and verification.

In the following sections, we compare and contrast “shredding” to existing approaches, and follow with a description of the SHREDDR system and a case study. Next, we illustrate

SHREDDR’s design principles and quantitative results for data quality and effort. Finally, we summarize our findings.

4.2 Related Work

The most relevant prior work to SHREDDR includes research on document recognition and various other systems for improving data digitization.

4.2.1 Automatic document processing

One of the first applications of computer technology was for the digitization of paper forms for large enterprises. Platforms like EMC’s Captiva ¹, and IBM’s DataCap ² handle millions of forms, ranging from invoices to insurance claims and financial statements for well-resourced organizations. Completely automated end-to-end solutions exist, but are financially unattainable for low-resource organizations.

The document processing industry views paper digitization as consisting of three main challenges: (1) recognizing the page being examined, (2) reading values from the page and (3) associating values together across pages [69]. The field of document recognition has provided reliable solutions for a number of these challenges, including: automatic recognition of handwritten numbers, short text with a small number of possible values, and machine produced text. Researchers now focus on issues like reading text in many languages and automatic segmentation and reading of challenging document images. The field, like many enterprise-scale systems, has given relatively little attention to the user-centric requirements and contextual constraints [69] that would make their tools useful in low-resource contexts. Addressing these issues is SHREDDR’s primary focus.

4.2.2 Improving data entry

Lorie *et al.* proposed an automated data entry system that integrates operator intervention with OCR, and describes a multi-staged verification process that improves the results [48]. This approach focuses on optimizing human attention for correcting OCR in an enterprise environment. Their iterative results improvement model is similar to some aspects of the SHREDDR’s pipeline. Several academic evaluations of automatic forms processing systems have been performed on flight coupons [84, 49]. In contrast to these OCR-then-correct approaches, SHREDDR is (1) an on-line service that (2) handles arbitrary legacy forms, with format agnostic input.

¹<http://www.emc.com/products/family2/captiva-family.htm>

²<http://www.datacap.com/solutions/applications/forms-processing>

There have been novel mobile data collection solutions specifically designed for low-resource organizations. Systems like CAM and Open Data Kit provide *in situ* data entry directly on mobile devices [34, 62]. In contrast with SHREDDR, these approaches conflate rather than separate capturing versus encoding data.

DigitalSlate is an elegant approach that uses pen and paper on top of a customized tablet device, providing immediate electronic feedback to handwritten input [66]. In this approach, the number of these novel and dedicated tablets required grows with the number of agents or points of service delivery.

Automatic interpretation of bubble forms was evaluated for numeric data entry [75]. SHREDDR handles all form data types.

4.3 Existing Solutions

In this section, we describe the prevailing standard approaches: direct digital entry, and double entry from paper forms. Examination of their strengths and weaknesses motivates our ensuing discussion about SHREDDR.

4.3.1 Replacing paper with direct-entry

Direct digital entry means digitally capturing data as close to the source as possible, often using a mobile device. Direct entry focuses on improving efficiency and feedback. The thinking goes: if a remote agent can enter values directly into a device, then it saves the work of hiring data entry clerks and transferring paper. Several groups have demonstrated potential efficiency improvements, on-device feedback to the worker, and faster turn-around time through digitization at the point of collection [22, 34, 71].

However, in many situations paper cannot be replaced; recall the discussion in Chapter 1. Organizations may desire to retain paper documents for reference [62]. More importantly, adoption of new technology and associated work-flow changes can incur significant cost, including training, hardware, and software. For some settings, the initial and on-going costs of such an approach is not financially viable over a reasonable time frame [65]. Low literacy users have also shown preference for traditional analog information capture mediums such as voice or paper [64]. A study of students doing math problems showed that writing modalities less familiar than pen and paper increased cognitive load and reduced performance [40].

4.3.2 Traditional double data entry

Double data entry (DDE) refers to keying the same values multiple times by different workers. The two (or more) sets of entries are batch compared after completion to determine the values that need correction. This comparison-and-fix process is most often performed with software.

Data quality from data entry paper forms can vary widely, ranging from poor to excellent. A study of paper data collection in South Africa demonstrated completion rates of only about 50% and accuracy rates of 12.8%. [51]; paper-based clinical trials (for pharmaceutical development) achieve error rates of 0.15% [68], relying on well-honed double entry practices.

The user's skill level is critical for accurate and efficient data entry. Factors like experience, job engagement level and subject matter expertise of the worker are significant factors in data quality. Trusted workers with high accuracy can produce results close to that of DDE [41].

4.4 SHREDDR

In this section, we describe how the SHREDDR system digitizes paper forms.

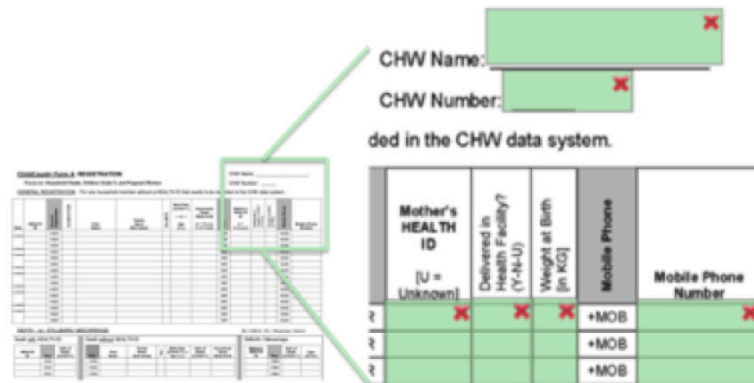


Figure 4.1. A form image template on the left; highlighted field areas of interest on the right.

1. First, a user uploads a sample of a form ("template") to SHREDDR's servers. Using SHREDDR's on-line tools, users highlight data fields of interest using a web interface (Figure 4.1). The user draws boxes of relevant fields, and for each field, provides us with a few pieces of meta-data, including: name, data-type, unit, significant digits, and potential values. It has been shown that this process itself can be sourced to Amazon's Mechanical Turk (MTurk) [47].

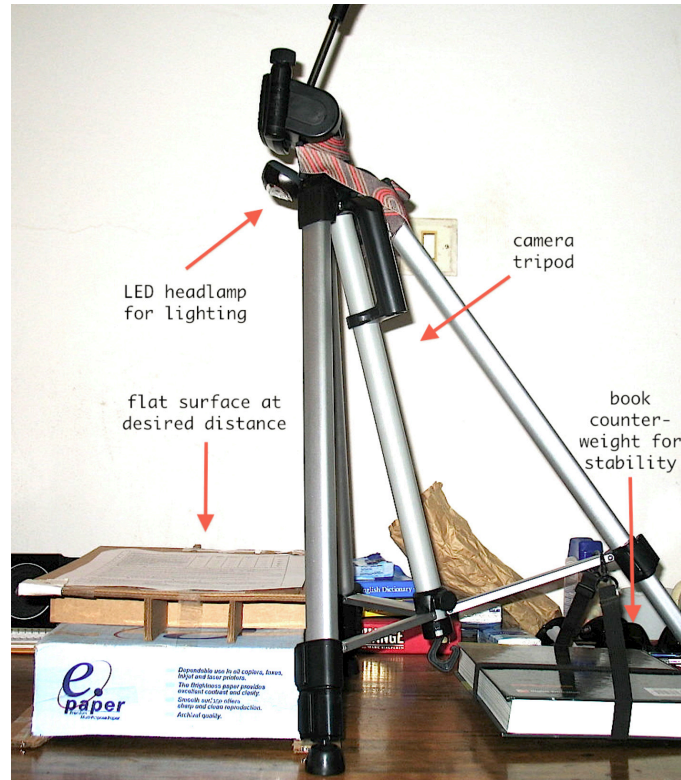
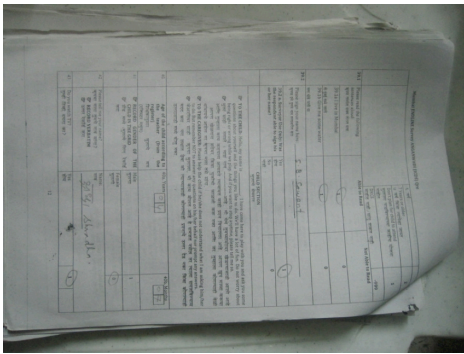


Figure 4.2. Our makeshift copy-stand used to digitize a country-wide survey in Mali.

2. Next, the user generates images of the forms (“instances”) needing processing. This can be done using a multitude of methods (e.g. scanner, low-cost digital camera, camera phone, etc.; see Figure 4.2).
3. These images are then uploaded to our website, either individually or using a batch submission interface (for example: a file or image service, email, website upload.) Images are aligned and auto corrected for rotation, lighting, and perspective transformation using computer vision algorithms (Figure 4.3)
4. The system then “shreds” apart the form images into fragments according to the document definition (Figure 4.4).
5. Shreds are batched together by field (for example: all the numeric age fields) into groups of 30 - 100, depending on field difficulty, and the type of question. We prompt crowd workers using specific instructions like: “Type in this handwriting” (entry), “Verify our guesses” (verify) and “Correct the spelling” (spell-check). An example entry interface is found in Figure 4.5 Each shred is tracked by a *decision plan*—a state machine that decides, as estimates come in from workers, when to confirm an answer (depending on the desired accuracy), or what next question to ask.
6. Gold standard values and decision plans’ logic govern data quality heuristically, based on data type (details of SHREDDR quality control process in Section 4.7).



A.

B.

C.

Figure 4.3. A. The original image, B. The template form, C. the results of image registration.

Asynchronous daemon processes advance the state of decision plans as estimates arrive from workers.

7. Finalized shred decisions, history of decision processes, along with shred and original page images are securely stored in SHREDDR's database. The end user can browse and edit, filter by problematic values (such as those marked for user review), and download in CSV format.

4.5 Why “shred”?

SHREDDR decomposes the data entry work-flow into many small steps, and shred apart paper documents into many small images. Our approach is to data production what to manufacturing assembly lines are to, say, cars. Pipelined digitization steps each require a specific type of human or algorithmic attention. Automation does the “heavy lifting”—the straightforward, anyone-can-do-it tasks, and specializes workers to focus on particularly problematic values that require expertise or local troubleshooting.

In this section, we examine the benefits of SHREDDR, which (1) allow users to control the trade-off between cost, data quality and turn-around time, (2) handle arbitrary paper forms, old and new, and (3) make minimal impact on an organization's capital investment in technology, training, and staffing.



Figure 4.4. Form image on the left, “shredded” fragments on the right.

4.5.1 Capture and encode separately

The first principle behind pipelined data entry is to separate capturing and encoding data into many distinct steps (recall Chapter 1). The *capturing* step records analog signals from the real world into a persistent digital state, and *encoding* steps iteratively transcribe, clean, translate and annotate the recorded signals according to pre-defined schemata. Field workers of low-resource organizations are often well-suited to capture local information, given their contextual familiarity. However, they may be ill-suited to encode high quality information, which requires more literacy, training and domain / technology expertise.

Adoption: From the perspective of an organization, subscribing to a data entry service is an incremental change to an existing paper-based work flows. Wherever *in situ* data entry needs to occur, a “data-imager” armed with a single device for imaging and upload will suffice. SHREDDR’s elastic worker pool can turn-around structured data on the order of many hours or few days. These characteristics make it easier for organizational champions to muster the political will to adopt this technology.

Robustness: Imaging devices are resilient to infrastructural unreliability. Intermittent power failures may grow the paper stack to-be-scanned, and Internet failures may grow the upload queue, but these failures do not disrupt the flow of paper. SHREDDR is also robust to data schema change. The data manager can simply update the document definition on-line.


4.5.2 Paper independence


The second principle behind pipelined data entry is to separate the usage of paper from the usage of data. The main idea, like the concept of *data independence* from the data management literature, is to uncouple the organization and layout of data on paper from

Type number

Please enter the numbers written.
 (Anything spelled out, like 'twenty two', should be written as digits, '22'.)
 Numbers are here:

23a.	
24a.	

1  nothing written here
▲ can't read this

2  nothing written here
▲ can't read this

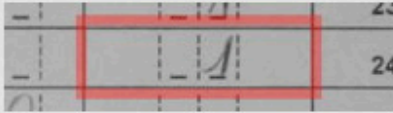
3  nothing written here
▲ can't read this

Figure 4.5. Entry interface.

its usages. In other words, fields on a page must be accessible without the page, and the person who captured the value cannot be required to know all its intended usages. This decoupling of value from page has significant benefits:

Access control: Shredding allows different users to see different shred image subsets, for example: (1) research staff can double-check single values, or (2) a subject matter expert can look at just at-risk or “user review”-flagged values.

Because workers only see a single field at a time without knowing what type of document it comes from, we have the potential to safely digitize sensitive data, including personally identifiable information by presenting to on-line workers in a way that does not allow an individual to decipher the original content by combining shreds over time. Shredding allows automatic redaction of values from images before they are shown to workers.

Multiple worker pools: Shredding allows granular control over sending work to mul-

multiple different providers. For example, regulations may require sensitive data to be digitized in-country, or values flagged as difficult by MTurk workers can be sent to specialists at an organization's headquarters. Thinking more creatively, medically related shreds would be ideal as a re-CAPTCHA³-like mechanism on particular types of websites. We may even be able to rely on "cognitive surplus" in the developed world, in the form of volunteer workers earning social kudos or other incentives [74].

Lazy evaluation: With linked images to shreds available, it may not be worthwhile to transcribe high-cost responses such as open response text fields, or data captured just-in-case, since it can be gleaned from the image when relevant and on-demand.

Furthermore, because answer quality for fields can be improved on-demand (with more estimates and verifications), an organization may choose to start with only computer-vision estimates and request higher confidence values later when needed.

4.5.3 Create economies of scale

Aggregating data entry work from many sources together allows us to achieve economies-of-scale across documents and organizations. By hosting a central service, we can (1) invest deeply in new data digitization technologies, e.g. improve automatic value estimation via computer vision and other probabilistic means, (2) have enough volume to engage with many different labor pools, and (3) specialize work assignments according to workers' skills and incentives.

4.6 Case Study

SHREDDR digitized the paper forms of a paper survey of Malian citizens across 576 villages in 95 rural towns with the goal of evaluating the political impacts of a randomized information intervention: a village-level civics course. Ten survey enumerator teams filled out 36819 pages of three surveys with 460 fields and 13 distinct pages. Their work occurred over three months, ending in August 2011.

The project had planned to hire two or three data entry clerks, rent office space, purchase computers and provide training and supervision in-country. By their own estimation of three surveys per hour rate of data entry, it would have required two data clerks seven–eight months to finish the job at single-entry quality. This rough estimate totaled 2600 estimate hours of human work—assuming five working days per week, and eight working hours per day.

Using SHREDDR, one data staff member was trained to collect paper forms at a centrally

³<http://en.wikipedia.org/wiki/ReCAPTCHA>

located office. He photographed the pages with a point-and-shoot digital camera (Canon Powershot A620) and tripod (as seen in Figure 4.2).

4.6.1 Imaging and uploading

For imaging and uploading, the data manager explained: “I was able to explain the process in 10 minutes to [a data clerk] and leave him unsupervised.” Striking a balance between legibility and upload size, each image was saved in JPG format at 1200 by 1600 resolution and averaged 350KB each. Images were transferred via SD card swapping to Picasa image management desktop software⁴ on a netbook laptop computer. In Picasa, images were batch-renamed according to a previously agreed-upon convention, and exported to a shared DropBox folder. DropBox⁵ is a commercial software service for synchronizing files over the Internet—it provided us delay tolerant file syncing over a 3G mobile data connection. Each battery charge lasted about 6 hours. If there was no power, back up batteries were needed or the upload process was stalled.

The data manager estimates that he processed roughly 150 pages (or 22 surveys) per hour, and split his time as follows: 15% organizing and preparing paper forms for imaging, 60% photographing, and 25% organizing and uploading the digital files. At this rate of image creation and upload: 150 pages per hour * 350KB per page, they need only a 15KB/sec upload link to keep up with the demand. This rate is achievable with a GPRS cellular connection.

4.6.2 Alignment and shredding

We batch loaded the uploaded images into SHREDDR and first performed image alignment. As part of alignment, SHREDDR’s computer vision based algorithms attempt to detect when pages may be swapped, missing or mislabeled. A generic solution to this problem, referred to as *doctype classification* by the document processing research community, is still considered a “holy grail.” We side-step the generic problem, for the batch upload case, by specifying a file naming convention.

4.6.3 Document definition

Our document definition tool has been successfully used by several well-educated and computer-savvy survey researchers. In the Mali case, the principal investigator (PI) of the survey project marked up each of 13 template pages after a a single in-person tutorial

⁴<http://picasa.google.com>

⁵<http://dropbox.com>

Datatype	#Shreds	%	Time
Text	186324	18.7%	18.8s
Integer	180144	18.1%	10.1s
Select-one	551542	55.4%	11.0s
Select-many	77284	7.8%	12.7s
Total	995294	100.0 %	12.5s

Table 4.1. This table shows the number of shreds (and percentage of total), and mean amount of worker time spend per value for the case study dataset.

Country/Territory	Visits	Pages/Visit
India	7565	16.7
United States	1939	9.4
United Kingdom	108	32.4
Chennai	1799	21.6
Bangalore	1385	14.9
Total	10443	

Table 4.2. Worker demographics

session. Document definition requires sufficient domain expertise to know which data fields are relevant and what the potential answers could be.

4.6.4 Digitization effort

This dataset has about 1 million values (shred images). The number of shreds (and percentage of total) per data type can be found in Figure 4.1. Figure 4.1 also shows the cumulative amount of worker attention (from 3 up to 12 workers). Quality will be explored in detail later in Sections 4.7.

MTurk workers cumulatively spent 2923 hours on this dataset. Using SHREDDR, each shred was seen by at least 3 MTurk workers. We can see that SHREDDR has a clear advantage in cost. Still, we believe the total effort can be much lower.

During data entry, the maximum throughput reached roughly 3000 assignments per hour. Throughput was constrained by both by an early system slowdown (bug), as well as, later, reaching an arbitrary maximum limit of tasks live on MTurk. With starts and stops, the run took more than 4 days. At the observed peak rate (which we do not believe is a true peak), we could have completed the dataset in under 24 hours (71169 total assignments were submitted).

4.6.5 Workers and errors

3672 unique workers saw our tasks on MTurk over 10,443 visits; each performing on average 28 minutes of work. 98% of those workers used a downloaded browser (Chrome and Firefox), indicating some technical sophistication. Most workers hailed from India. Among Indian workers, almost half had IP addresses registered in Chennai or Bangalore (Table 4.2), both being significant outsourcing hubs.

4.7 Data Quality

Errors can happen at each step of the data collection pipeline. Here, we briefly introduce some common techniques of managing data quality and discuss how SHREDDR contributes to these methods.

Incorrect or missing measurements during initial capture: These issues fall under the topic of instrument and protocol design. To address these errors, an organization may take action such as:

1. Planting respondents who give known answers among the population
2. Repeating measurements over the same populations
3. Adding questions to the data collection instrument to cross-validate important answers
4. Finding outliers in the answers, and prioritizing the most unlikely answers for review

These methods help establish invariants, which if violated, indicate quality degradation. SHREDDR helps in doing so by providing data incrementally, as collection takes place, so data managers can make corrections in-flight.

Corruption to digital files or physical paper during transit or storage: These issues are a matter of backup and security. An organization must implement procedures for internal safekeeping unless it is outsourced to a trusted third party. In our experience, the longer paper stacks sit idle, without digitization or curation, the more likely it is lost, shuffled, and lose usefulness. By capturing whole document images, then transmitting, encrypting and storing them in the “cloud”, SHREDDR is a straightforward means to safeguard against these issues.

Mis-keying during transcription: These errors are the primary focus of our discussion below. At a high level, an organization can fix mis-keying with the following approaches:

- Use gold standards to measure error rates and root out systematic errors early
- Repeat entry and verification to fix random errors

- Leverage dynamic interfaces to guide high quality entry (Chapter 3)

Below, and in the following sections, we will describe each of these methods and how they fit together in SHREDDR’s digitization pipeline.

4.7.1 Double entry quality

To better understand how SHREDDR manages quality control, we first take a deeper look at the way double entry works. In full-DDE, when all values are double entered, the approach is quite effective in catching random errors. We propose a simplified model of the expected DDE error rate: errors in the data that were flagged and incorrectly reconciled and errors that were not flagged; more specifically:

$$p(dde) = p(de_conflict) * p(judge_error) + p(mutual_mistakes) \quad (4.1)$$

The first term models the likelihood that two independent measurements disagreed, and a third attempt to arbitrate was also mistaken. The second term models the likelihood that the two independent measurements made the same mistake on the same answer.

However, errors can quite easily be systematic in nature; for example, both DDE workers misunderstand the entry protocol and skip an entry that should be have the value as “not applicable”. In this case, DDE cannot recover these errors.

Often, due to resource and time limits, partial-DDE only checks a portion of values, as a quality-control measure, to ensure that the single-entry error rate is at a reasonable level. In this case, the expected error rate decreases as follows:

$$p(partial_dde) = p(single_pass_error) - portion_double_entered * (p(dde) - p(single_pass_error)) \quad (4.2)$$

This quality check is quite useful for many reasons, including reporting accuracy.

4.7.2 Quality control

Here, we detail SHREDDR’s approach and analyze results of the Malian dataset.

Gold standards: SHREDDR programmatically creates *gold standard* values, from a randomly selected subset of shreds, to act as the first line in quality control. This has been

shown to be an effective technique for managing quality in crowd-sourced systems [43, 57]. For gold standards, we are very conservative about error and do not mind a high rate of failure: gold shreds are randomly assigned, with entry prompts, to three to five different MTurk workers. A gold shred is accepted if all answers exactly match (after canonicalization to case-insensitive Unicode).

For the remaining shreds, entry batches are injected with a small number of correct gold standards; verify batches are injected with correct and incorrect gold standards. If an answer set from a worker fails a majority of gold standard values, it is rejected.

Gold standards have the added benefit that we use them as ground truth data to compare against regularly-transcribed values, and achieve the same effect as partial-DDE (Equation (4.2)). However, we must first make up for the gold standards that were attempted-but-failed, in order to have an unbiased, random sample.

Decision plans: SHREDDR currently uses heuristic decision plans for catching mistakes. For all data types, but particularly numbers and multiple choice answers, we use *entry + majority-verification*(E3V) decision plans. We have also created other decision plans, for example: for handwritten text answers, we may use a *double entry + spelling-correction*.

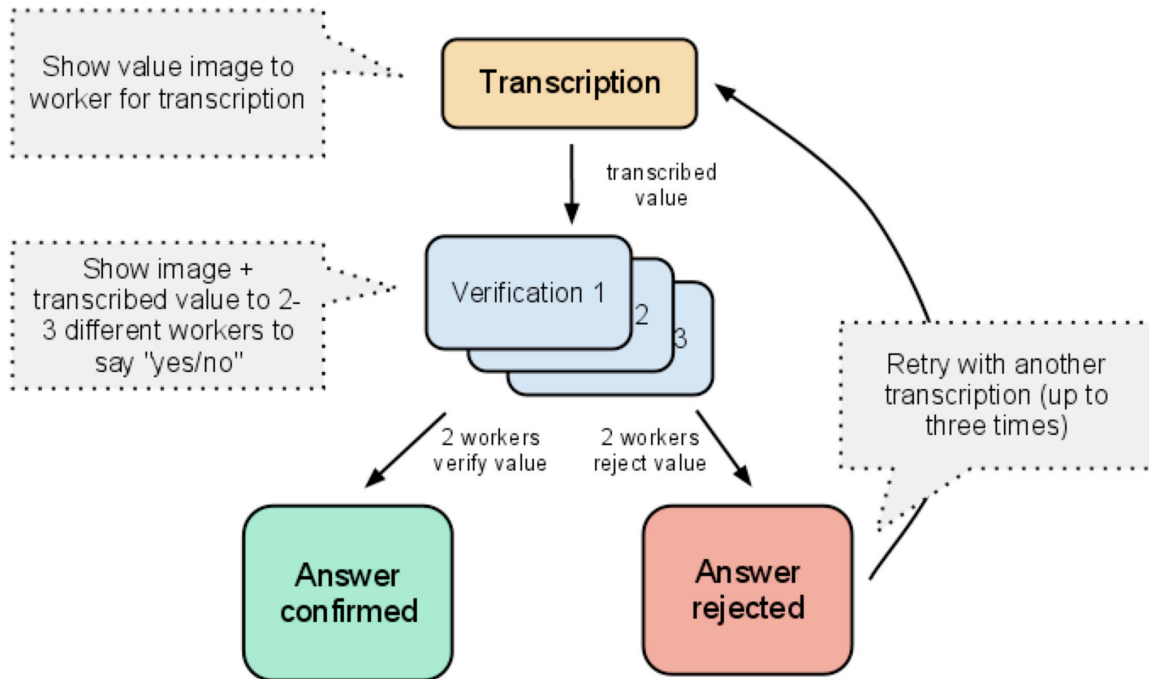


Figure 4.6. Quality control work-flow in SHREDDR (E3V).

E3V: The decision process, shown in Figure 4.6, starts with a single *entry* estimate for each shred. The estimate is verified by a majority vote over two (or if necessary, three) *verify* tasks. If the vote fails, repeat the loop; if rejected three times, mark as needing user review.

DES: This decision process starts with DDE (Equation (4.1)): source two entry estimates from different workers, if they match (after canonicalization), accept the answer. If the answers are close (by the metric described below), we ask a third worker to choose the one with better spelling and accept that answer. If the answers are not close or the shred is flagged “can’t read this”, ask for a third estimate. If the 3rd estimate answer does not match either the 1st or 2nd, mark as needing user review.

Automatic word and mark recognition algorithms can be used to make initial entry estimates. These improvement promise to drastically reduce the amount of human attention required for good estimates. Automation of straight-forward work will allow human workers to focus on more difficult (and higher paying) tasks.

4.7.3 SHREDDR results

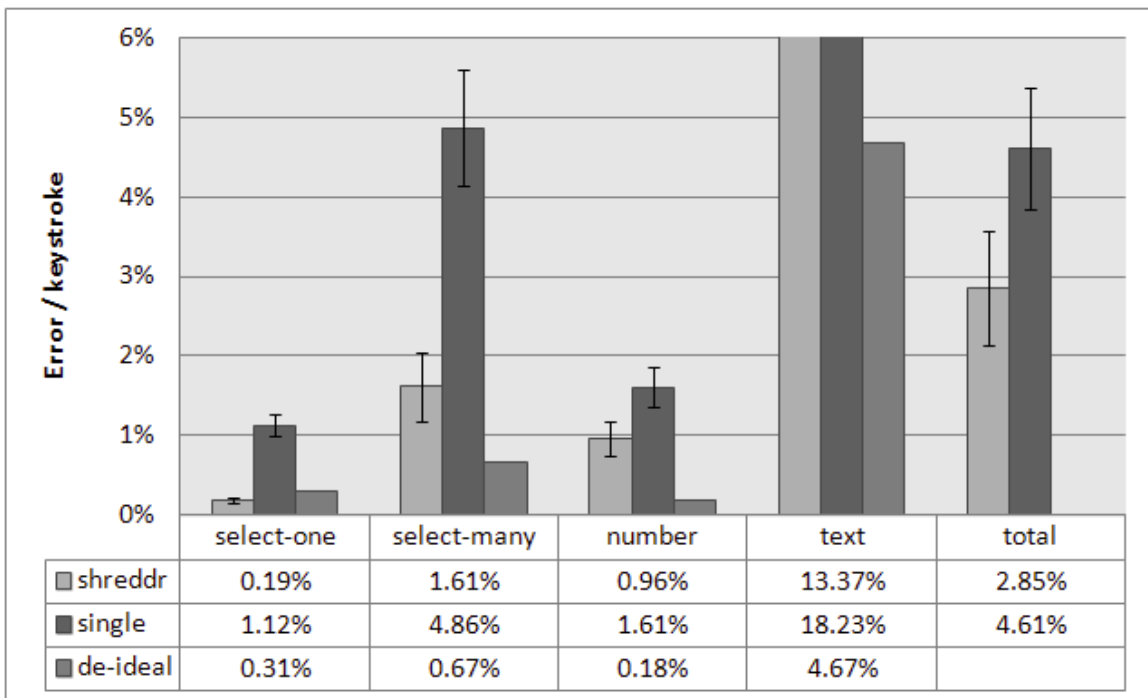


Figure 4.7. Accuracy of SHREDDR versus single pass entry and that of idealized double entry.

We processed all shreds of the case study using the E3V decision plan. In order to measure quality, we sampled a random subset of 23,255 shreds, and manually curated a “ground truth” baseline based on the gold standards approach describe above.

The industry-standard metric for comparing data quality is errors-per-keystroke. For

text or number entry, it means the number of characters typed. For multiple choice fields, it is the number of items selected.



Figure 4.8. Examples of difficult shreds. In A and B, the number script is unfamiliar to many workers; in C, French and cursive handwriting proved difficult; in D, the markings require some deduction to determine what is the intended answer.

Figure 4.8 features some example shreds that workers got wrong or were marked “I can’t read this”: in A and B, the number script is unfamiliar to many workers; in C, French and cursive handwriting was difficult; in D, the marking require some deduction to determine what is the intended answer. Notably, these examples show that there exists some baseline rate of ambiguous shreds that must be “caught” and escalated back to the user (and potentially the data’s source).

We observed an overall errors-per-keystroke rate of 2.85% (label `shreddr` in Figure 4.7). SHREDDR performed well on select-one multiple choice fields: 0.19%, and numbers: 0.96%, as well as select-many multiple choice fields: 1.61%.

The text entry error rate of 13.37% required more exploration. The responses being in Malian French was the chief cause of error. More specifically, (1) MTurk workers’ unfamiliarity with the language prevented them from correcting ambiguous writing to actual words and phrases; (2) the handwriting was often in French cursive script which featured unfamiliar serifs and conventions; (3) many MTurk workers did not know how to type accented characters (such as “é”).

It is entirely possible to transcribe difficult text and achieve high quality results, with both poorly printed text , as well as ambiguous English handwriting on MTurk . We corroborated these results in a separate quality assessment, which removed handwriting ambiguity and the need for word-level semantic understanding. We used unfamiliar, but unambiguous and English-language keyboard type-able, words, asking MTurk workers to enter Ugandan last names rendered in a handwriting-style font (shown in Figure 4.9). The experiment offered tasks with 25 names to type for \$0.02 USD, and then doubled the number of names per task to 50, 100, and so on, until workers were unwilling to perform the task. We measured the errors-per-keystroke rate of all completed tasks to be 1.28% in a single-pass.

Character-separation can be an important quality control mechanism. We noted that when the space to write a short response is placed in a character-segmented box, like this:

Name : | - | - | - |

the results were significantly better than those for open-ended response fields.

To best address this issue, we simply need to find French-proficient workers. More broadly, we need to specialize a subset of the workforce to a subset of the work. On MTurk

Please transcribe the following words.

Requester: Shreddr

Qualifications Required: None

Mukandutiye	
Bibyumumeisho	
Tibihwa	
Aryatwihayo	
Kansasira	

Figure 4.9. Example interface showing Ugandan last names rendered in a cursive font.

specifically, we can require workers have French knowledge through a “qualification”, and beyond MTurk, we can engage with other worker pools.

4.7.4 Quality comparison

Data collection scenarios differ significantly across different contexts. Recall in Section 4.3, that data entry quality can range up to many orders of magnitude, making it very difficult to make “fair” comparisons. A poorly designed instrument, under-trained enumerators or data entry workers can each make significant impact on data quality. As such, for our comparison, we try to hold contextual factors constant and compare the outcomes of SHREDDR digitization, to that of hypothetical single and double entry scenarios.

In Figure 4.7, the variables `shreddr` refers to SHREDDR’s finalized results; `single` refers to the same work that was assign to MTurk workers using the same interfaces and infrastructure as SHREDDR, except SHREDDR’s decision plan quality control mechanisms; `de-ideal` is the hypothetical best case for double entry quality (details about how it is computer to follow.)

Vs. single pass: We see that SHREDDR outperforms single entry across every data type. For categorical fields, SHREDDR is an order of magnitude more accurate; in select-one accuracy, SHREDDR is the range of the lowest reported error rates from the clinical trials literature [41, 68]. Number fields saw 40% fewer errors than single pass entry, and for text fields, the advantage was a smaller 27% fewer errors. The relatively smaller improvement for text, given the above discussion about French language transcription, is indicative of a

baseline error rate, due to systematic error rather than something repeated measurements can fix.

Vs. ideal double entry: Recall Equation (4.1):

$$p(dde) = p(de_conflict) * p(judge_error) + p(mutual_mistakes)$$

To calculate *de-ideal*, we make two simplifying assumptions: (1) the data accuracy of reconciliation is equal to that of single entry; and (2) in general, mistakes are independently and uniformly distributed. The latter implies that we can model the likelihood of a worker choosing a particular wrong answer as a random drawing from the *domain* of possible choices, repeated for each decision (a *keystroke*). This means:

$$\begin{aligned} p(judge_error) &= p(single_pass_error) \\ p(mutual_mistakes) &= p(single_pass_error) \\ &\quad * \frac{1}{domain_size^{keystrokes}} \end{aligned}$$

So, our model becomes:

$$\begin{aligned} p(dde) &= \\ & p(de_conflict) * p(single_pass_error) \\ & + p(single_pass_error) * \frac{1}{domain_size^{keystrokes}} \end{aligned} \tag{4.3}$$

We parameterize the model with a combination of empirical measurements and derivations, summarized Table 4.3. $p(de_conflict)$ are the empirical % values that were flagged by two independent single entry passes; *average keystrokes* are the empirical averages representing the number of categorical values chosen or the length of characters typed; *domain size* is the total number of categorical values available or 10 and 26 for number and text fields, respectively. For example, for select-one fields: $3.39\% * 1.01\% + 1.01\% * (1/4.1^1) = 0.31\%$.

We were pleased to see that SHREDDR’s select-one error rates were below what would be expected from idealized double entry (select-many, number and text types are roughly 2.5, 5.5 and 3 times the ideal rate, respectively). Recall the difficult shreds shown in Figure 4.8: we must keep in mind that the ideal double entry error rate does not account for the baseline error rate due to ambiguity, that is, values that we simply cannot say for sure what is the correct value.

This comparison validates SHREDDR’s ability to surpass standard practice in terms of quality, as well as pointing out areas where we must improve.

	select-1	select-many	number	text
de_conflict	3.39%	8.96%	5.50%	25.60%
keystrokes	1.00	1.57	1.26	2.75
domain	4.13	6.92	10	26

Table 4.3. $p(\text{de_conflict})$ is the empirical % of values that were flagged by two independent single entry passes; average keystrokes empirical averages representing the number of categorical values chosen or the length of characters typed; domain size is the total number of categorical values available or 10 and 26 for number and text fields, respectively.

	1 worker	1 or 2 workers	3 workers
P(disabled)	7.9%	0.76%	0.04%
% answered by	1.3%	16%	84%

Table 4.4. Probabilities of a gold standard value being answered by, and being disabled, by number of unique workers

4.7.5 Error independence

The ability to assign shreds to workers gives us independence from the original paper form. Paper independence also implies error independence—that errors are not systematic and correlated to other errors. The key idea is that a single worker may tend to make the same type of mistake for same type of question over and over again, perhaps due to systematic biases, such as misunderstanding task instructions. Consequently, digitization approaches with relatively few workers has a higher likelihood of encountering systematically-confused workers agreeing on wrong answers. In contrast, SHREDDR distributes the work across thousands of workers. As such, for a given question, the probability that a random pair of workers will repeatedly make the same systematic error is less. This is not to say that MTurk workers are necessarily better individually, but with N workers, their systematic confusion will cancel out as N grows.

We can test this hypothesis by measuring the effects of worker distribution in SHREDDR’s generation of gold standard values, described in Section 4.7. After gold standard values are created, they are used to police other work. We keep a running rate of worker disagrees vs. agreement percentage, and automatically disable the gold standard if the rate rises above a conservative threshold T . A reject essentially means the value is wrong or ambiguous.

We examined the distribution of workers who participated in generating our gold standard data (Table 4.4). Recall that over three thousand different workers provided input to our case study dataset. Still, there was a 1.3% chance that a single worker answered all 3 gold standard estimates (separately), and a 16% chance that a single worker provided at least 2 of 3. These worker-distribution scenarios, per shred, is akin to that of DDE or direct entry.

When a gold standard was created with input from three different workers, its likelihood of being disabled is 0.04%; this increases to 0.76% if fewer than 2 workers provided the

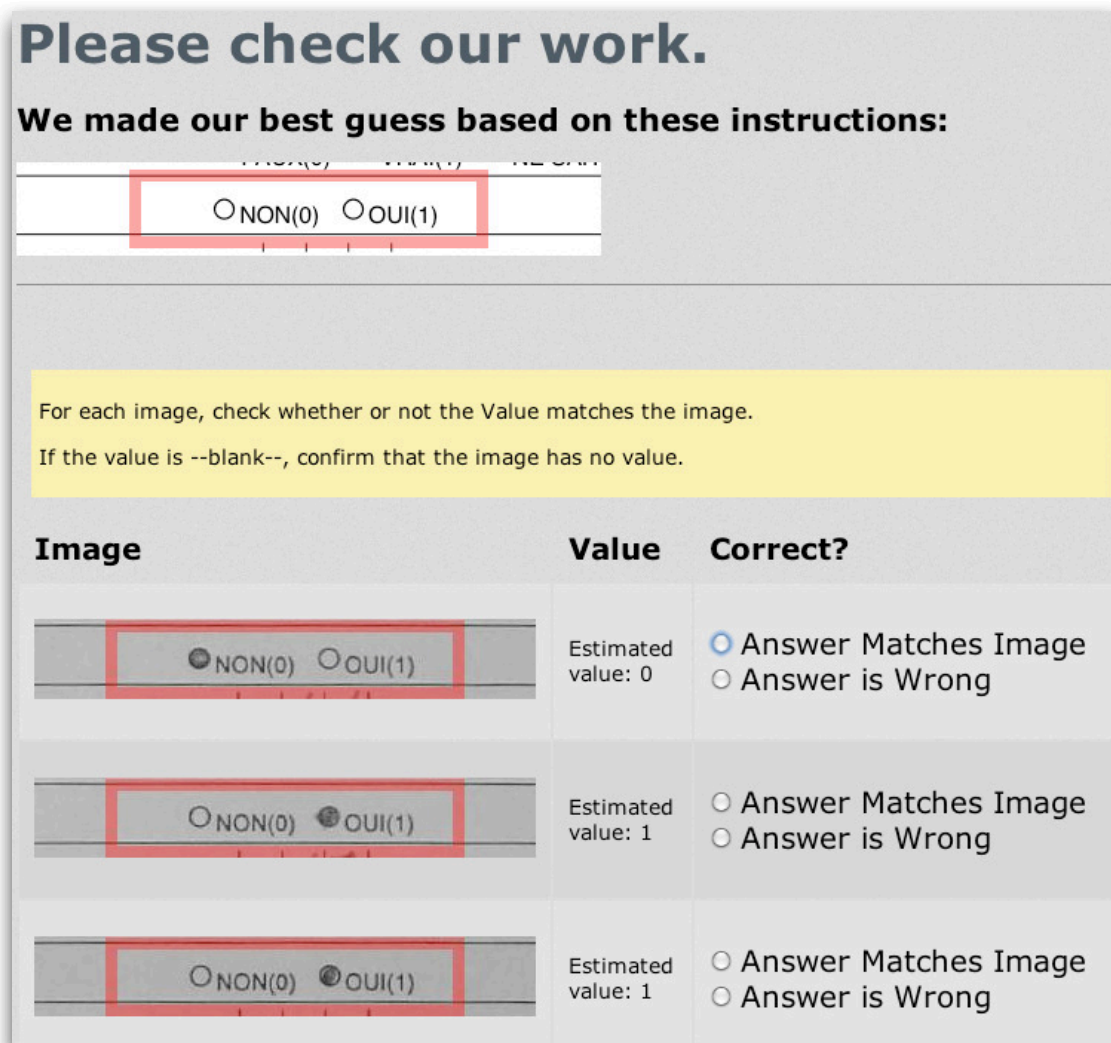
input, and increase much higher to 7.9% if only 1 worker provided all 3 estimates. In other words, a value that was agreed on by three different workers is exponentially more likely to be correct than if fewer (but still independent) workers provided the opinion. It stands to reason that SHREDDR can catch a much greater number of systematic errors with its wider worker distribution.

4.8 Efficiency

4.8.1 Shredded interfaces

Please check our work.

We made our best guess based on these instructions:



For each image, check whether or not the Value matches the image.
If the value is --blank--, confirm that the image has no value.


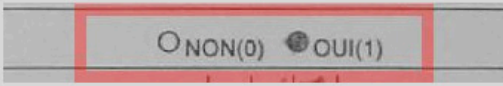
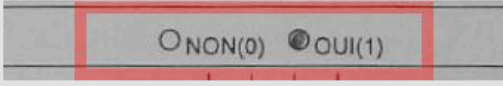
Image	Value	Correct?
	Estimated value: 0	<input checked="" type="radio"/> Answer Matches Image <input type="radio"/> Answer is Wrong
	Estimated value: 1	<input type="radio"/> Answer Matches Image <input type="radio"/> Answer is Wrong
	Estimated value: 1	<input type="radio"/> Answer Matches Image <input type="radio"/> Answer is Wrong

Figure 4.10. Verification interface: list-style.

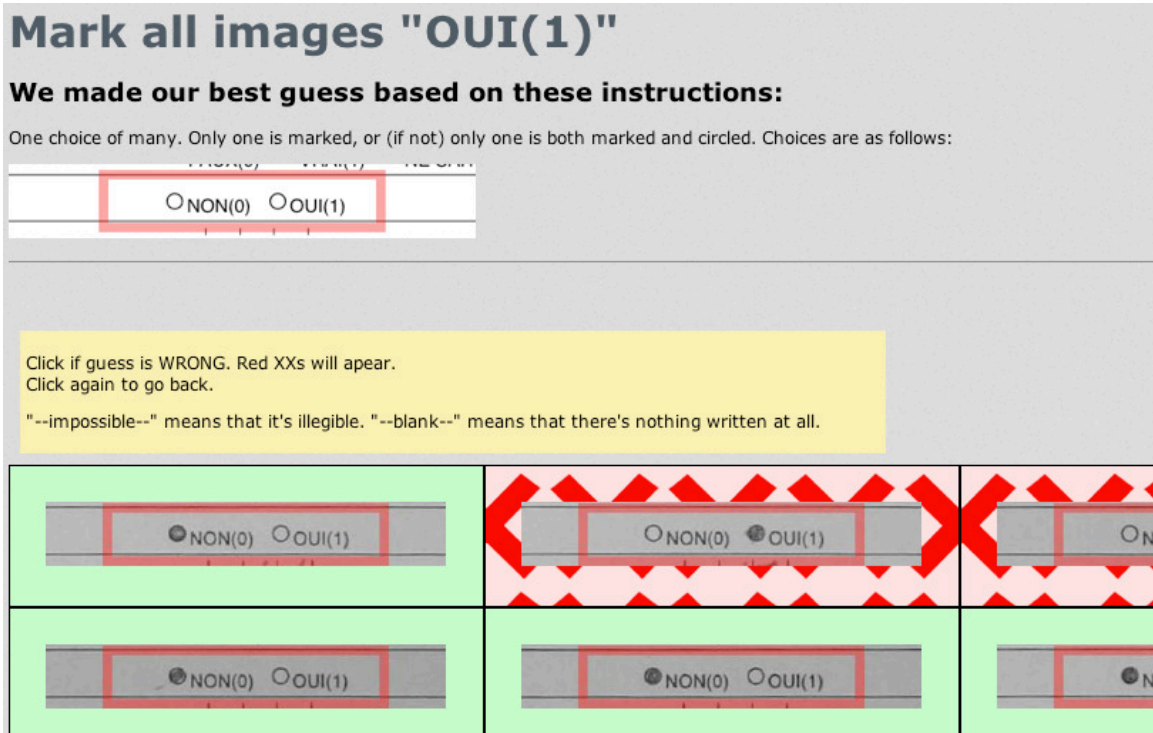


Figure 4.11. Value-ordered verification interface: grid-style.

Recall that the idea for SHREDDR began with a conversation about a run of “yes” values during data entry (Chapter 1). Repeated runs of the same value allow the worker to *compress* the input into one operation, like run-length-encoding enables vector operations over stored data. Data compression algorithms like run-length-encoding reduce the consumption of storage and and network bandwidth. In the same way, we can engineer the data entry task to reduce the consumption of human cognitive bandwidth.

Shredding filled-in documents create the opportunity to create worker optimized interfaces. The freedom to work with images representing individual values is key. Both physical and cognitive operations can benefit. We use information entropy, the metric for data compress-ability, combined with Fitts’ Law [27], the metric for physical efficiency, to measure the effectiveness of a compressed data entry interface

Order by field: Figure 4.5, as mentioned before, is an example of our entry interface. In it, we show only values from a single field—in this case, a 2-digit number. This interface is advantageous in terms required physical movements: the value image is co-located with the entry field, rather than, say, on a desk; as well as entropy: the worker only has to think about numbers, rather than switch mental context between different value domains. This also allows them to center their hands on the numeric keypad, if one is available.

Reformulation and order by value: Figures 4.10 and 4.11 show examples of our verification interfaces: the first lists our best estimate alongside the image and prompts the worker to confirm whether an answer is correct, the second pre-sorts our best estimates

by value, and lays out only those that we believe to be a particular value, and prompts the worker to click those that do not match. The list interface reduces the active domain size to a binary decision. The value-ordered grid interface reduces the task to that of pattern matching.

Efficiency analysis:

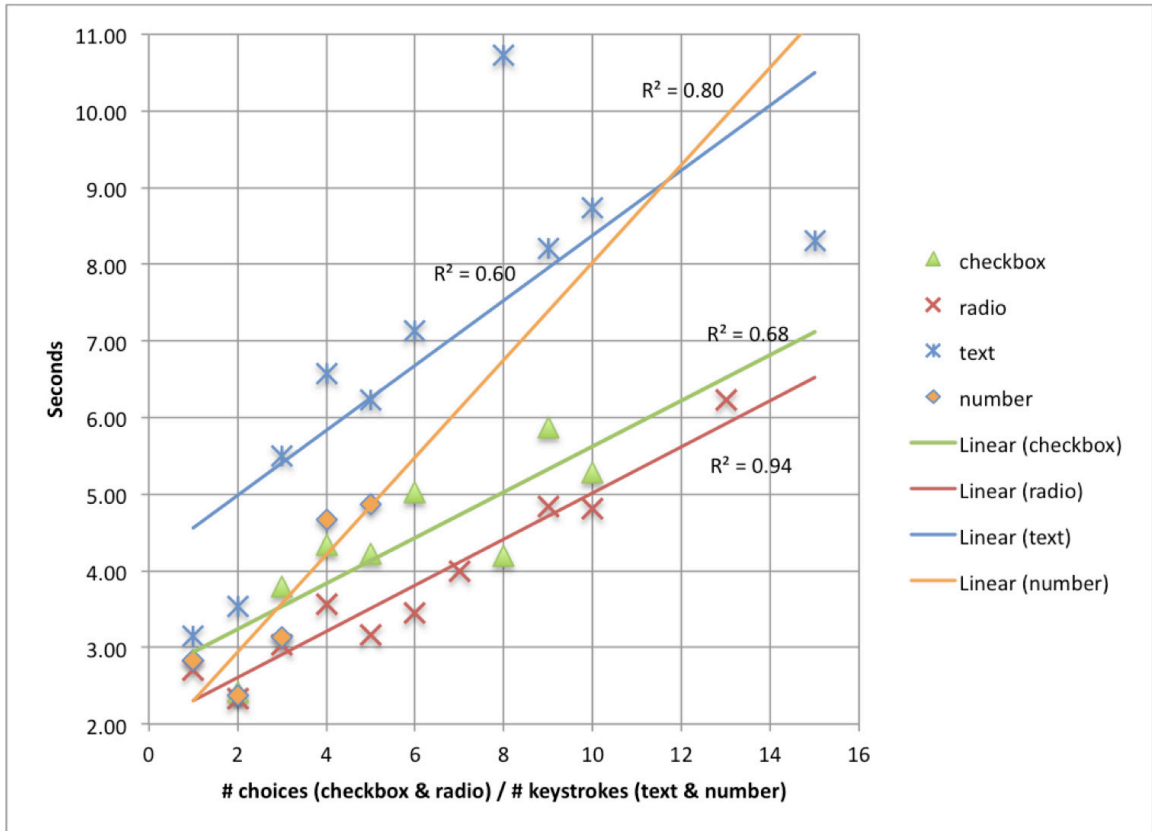


Figure 4.12. Durations in seconds of how long a MTurk worker spent per value, by data type and question difficulty

Figure 4.12 shows durations in seconds of how long a MTurk worker spent per value, by data type and question difficulty. We present question difficulty as the number of choices to select from for checkbox and radiobuttons, and the number of keystrokes for handwritten text or numbers.

We can see that for the same number of characters, integers take less time than text, providing more evidence that a smaller domain size (10 digits) is faster than a larger one (all letters). The durations of radiobuttons are consistently below that of checkboxes, because checkboxes can have multiple answers and thus a longer amount of time.

By the same argument, we can also gain by turning an entry task into a verification—recall the *question reformulation* technique from Chapter 2. Our earlier results show that for best guess estimates, reformulated questions can both be faster and more accurate.

4.9 Discussion

Scan and upload: The data manager reported that his biggest pain was removing staples from surveys and having to manually transfer and rename batches of images. Since then, we have improved our image alignment algorithm to handle stapled-page image distortion (see Figure 4.3). A mobile phone camera-based application can help make easier the process of organizing, imaging, and uploading.

Re-extraction: It is inevitable that mistakes are made in document definition. The implication is that workers are given bad instructions, and data quality suffers. For example, one of a question’s multi-select options was missing. Workers had to mark answers without the missing item, which led to some confusion. We fixed the mistake, and “re-extracted” the field as another run.

The re-extraction feature is a mechanism for handling data schema and extraction policy changes. It is notable that SHREDDR handles such changes with little additional overhead.

Escalation pipeline: Each response value lies on a continuum of difficulty between those that unspecialized workers can correctly transcribe (roughly speaking, via pattern matching), and those that are truly ambiguous. We call the process that each shred goes through an *escalation pipeline*.

DDE clearly points to a set of answers needing escalation. Its escalation pipeline is typically a one-step procedure: after independent entries are compared, all conflicts go to the data manager. Sometimes, an additional step may occur whereby the ambiguous answers go to original data capturer, if available.

Anecdotally, we have seen that the cost of this error reconciliation pass often goes unaccounted for when planning data collection projects. Assuming a random distribution of errors in full DDE, if a page has on average 20 fields, and the error rate is 5%, it is likely that all pages must be reviewed by the arbiter. Often, the arbiter is a data manager or higher-up who creates a latency bottleneck in the whole process—as a result, data often sits waiting for a 3rd “expert” pass.

Expert reconciliation is unnecessary for all conflicts. For example, random errors that are straightforward can be handled by just another measurement. Local experts with domain knowledge should only see the truly ambiguous values. Toward this goal, we propose a more refined and granular pipeline for data entry. Each step goes to increasingly specialized workers to solve parts of the problem, and to decrease the number of the values escalated beyond. To tease out types of errors, we escalate first to data entry specialists in particular aspects (such as a cursive script in a language), then to subject matter specialists (such as medical terms or regions of Uganda).

As we learn more about particular fields and shreds, we tag potentially difficult shreds in various ways, including:

1. Better-instructions-required
2. Need context from page

3. Need context from other fields
4. Domain-expert-required
5. Truly ambiguous

Shredded entry and traditional entry provide different types of context: shredded entry allows a worker to see across a value's domain (for example, other villages in a region). Traditional name-age-phone ordered entry provides semantic and correlational context, enabling a worker who is not speeding through the task to reflect on semantic incongruities (for example, whether a 5-year weighs 100 kilograms). Recall in Chapter 3 that our workers also often say they do not think about the meaning behind what they are transcribing. As such, the domain context is much more useful than the correlational context. As for the latter, if we use USHER, we can automatically find semantic incongruities as well.

4.10 Conclusion and Future Work

We have presented a new way to digitize paper documents into structured data. SHREDDR's primary design goal was to address the constraints faced by grassroots organizations and field offices of international agencies while streamlining data entry. This is achieved with a novel combination of emergent technologies and techniques, including computer vision, data compression, crowd-sourcing, cloud computing.

We presented a case study of the system in action, in which we digitized one million values from a large-scale citizen survey conducted in Mali. Our results indicate that this is an opportunity for improvement, particularly for dealing with specialized languages and in identifying and escalating difficult values to appropriate expert workers. Other data types (select-one) showed excellent results that put us in the same league as clinical trial practitioners.

Opportunities for future work include: develop algorithms that group together specialized and more difficult (higher paying) work; model confidence of estimates per value, and expanding support to in-house, in-country and volunteer workers.

Chapter 5

Conclusion

This dissertation addresses first-mile data collection challenges with data-driven techniques. The multi-prong approach covers optimizing individual acts of data entry to aggregating work across multiple organizations.

USHER relies on previous data to automatically improve digitization of arbitrary forms. Re-ordering, re-formulating and re-asking are techniques that reduce the amount of local expertise and training required to operate data collection work-flows. Furthermore, USHER models enable a number of dynamic user-interface mechanisms that improve accuracy and efficiency during the act of data entry. Based on a cognitive model, these interface adaptations can be applied as interventions before, during, and after input. We see both experimentally and empirically that USHER can increase the quality and efficiency of data entry.

SHREDDR allows field workers to capture information using existing and familiar paper forms, which are then transferred as images out of challenged settings, and iteratively digitized, using a combination of automated and human-assisted techniques. SHREDDR's approach segments the work-flow to leverage pipeline and task parallelism. The key mechanisms are to separate data capture from encoding, to shred images to create opportunities for vector operations in the interface, and to leverage on-line workers to achieve sufficient turn-around times at-scale.

5.1 Discussion and Future Work

5.1.1 Maximizing bits-per-decision

In data systems research, streamlining dataflows is about finding bottlenecks—mismatches in impedance. The orders of magnitude time-difference between the rates of human and computer decision-making indicate that we should optimize around the amount of latency human computation introduces into a computing system. As data digitization becomes more and more automated, we must ask, “What diminishing roles do humans have in a mostly digital system”? Our belief is that we should treat human computation as input for significant events, and use it during computational fix-points or to break ties, or to answer very hard problems. It follows that human-computer hybrid data flows must reformulate the human task to maximize the *bits-per-decision* as much as possible.

5.1.2 Bottom-up bootstrapping

The work of optimizing data entry work-flows all point towards the goal of an organization using and benefiting from data. An important long-term goal is to help local people (and their global supporters) surface locally-important outliers and trends that may otherwise get lost at higher levels of reporting (and aggregation).

Generating locally-actionable insights from data has a chicken-and-egg relationship with an organization’s ability and will to maintain its local data work-flows. Sometimes, immediate local benefit can be bootstrapped. While in a rural Ugandan village, I developed a simple tool that allowed clinicians to view data visualizations of health trends in their community. The key idea was leveraging “found data” from the intermediate results of fulfilling external data collection requirements. The tool was simple: an Excel workbook with macros that tapped into data collected from community health workers (CHWs). I created a workbook tab of visualizations featuring PivotCharts like “Patients under 5 years old with malaria by village”, and taught the village doctor create his own PivotCharts. The village doctor was delighted with his new-found ability to monitor CHWs through visualizations. I saw that the ability to see and benefit from CHW-collected data immediately improved the incentives and feedback loops for CHW data collection. This simple tool’s adoption is a hint at what an orchestrated and purposeful intervention can do in creating a virtuous cycle between better data and improved service delivery and capacity.

5.1.3 Getting Started

There are a number of first-mile data challenges that can be directly addressed by re-organizing and optimizing *local data work-flows*. Data scientists are well-positioned to

make significant contributions in this area, but have been focusing on backend infrastructure and algorithms because their customers – the people with the means to acquire and manage data – demand it. In order for us to create appropriate solutions, we must shift our attention to those potential customers who have trouble acquiring data in the first place. We must work on optimizing data work-flows in the context of limited human, organizational and technical resources.

The notion of “too much data”: William Gibson observed that “The future is already here, it is just unevenly distributed” [31]. This insight applies to data as well. Data scientists often talk about the data deluge occurring in the developed world, but there is ironically far too little data available about conditions in the developing world—data that is relevant to some of the most important challenges and opportunities of the 21st century. While we are very comfortable with issues like scale and privacy in data-rich environments, we are less familiar with circumstances where even the most basic improvements in data availability can enable significant progress in meeting local needs.

The infatuation with “big data”: Researchers take more interest in problems that center on large data volumes. But, because low-resource organizations use tools like Microsoft Access, they tend to fly under our radar. However, their multitude of “little data”, each different by culture and environment, also presents an interesting scale problem: the challenge of wide-scale in *contextual diversity*, rather than large-scale in volume.

The myth of expertise: We often assume that competent staff is on hand to implement, administer and use computer systems. This thinking is reasonable for many office-based, developed world environments, but if we want to extend the reach of our systems to more people and organizations, we must go further in terms of making our solutions more appropriate for a broader range of skill levels and familiarity with technology.

The most direct route to engaging with global problems is pragmatic. Several early researchers in this emerging field have highlighted a simple formula for achieving success [15, 61]: go to the problem, find a good partner organization, and solve their real problems in an empirically demonstrable, and hopefully broadly generalizable, way.

Bibliography

- [1] “CommCare,” <http://dimagi.com/commcare>.
- [2] “Infer.NET,” Microsoft Research Cambridge, <http://research.microsoft.com/en-us/um/cambridge/projects/infernet>.
- [3] “JavaRosa,” <http://www.javarosa.org>.
- [4] The millennium villages project. [Online]. Available: <http://www.millenniumvillages.org>
- [5] “Open Data Kit,” <http://opendatakit.org>.
- [6] “OpenMRS,” <http://openmrs.org>.
- [7] “OpenRosa,” <http://openrosa.org/>.
- [8] “Survey documentation and analysis,” U. C. Berkeley, <http://sda.berkeley.edu>.
- [9] “Human Development Report,” United Nations Development Program, 2007, <http://hdr.undp.org/en/reports/global/hdr2007-2008>.
- [10] “The open society: Governments are letting in the light,” *The Economist: A special report on managing information.*, Feb. 2010.
- [11] A. Ali and C. Meek, “Predictive models of form filling,” Microsoft Research, Tech. Rep. MSR-TR-2009-1, Jan. 2009.
- [12] C. Batini and M. Scannapieco, *Data Quality: Concepts, Methodologies and Techniques*. Springer, 2006.
- [13] J. M. Bernardo and A. F. Smith, *Bayesian Theory*. Wiley Series in Probability and Statistics, 2000.
- [14] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.
- [15] E. A. Brewer. (2007) VLDB Keynote Address: Technology for Developing Regions.
- [16] J. V. D. Broeck, M. Mackay, N. Mpontshane, A. K. K. Luabeya, M. Chhagan, and M. L. Bennis, “Maintaining data integrity in a rural clinical trial,” *Controlled Clinical Trials*, 2007.

- [17] W. L. Buntine, "Operations for learning with graphical models," *Journal of Artificial Intelligence Research*, vol. 2, pp. 159–225, 1994.
- [18] 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.
- [19] F. G. Cozman, "JavaBayes - Bayesian Networks in Java." [Online]. Available: <http://www.cs.cmu.edu/~javabayes>
- [20] T. Dasu and T. Johnson, *Exploratory Data Mining and Data Cleaning*. Wiley Series in Probability and Statistics, 2003.
- [21] S. Day, P. Fayers, and D. Harvey, "Double data entry: what value, what price?" *Controlled Clinical Trials*, 1998.
- [22] B. DeRenzi, N. Lesh, T. Parikh, C. Sims, W. Maokla, M. Chemba, Y. Hamisi, D. Sellenberg, M. Mitchell, and G. Borriello, "E-imci: improving pediatric health care in low-income countries," in *Proc. SIGCHI*, 2008.
- [23] C. J. Elias, "Can we ensure health is within reach for everyone?" *The Lancet*, vol. 368, pp. S40–S41, 2006.
- [24] M. W. Eysenck, *Principles of cognitive psychology*. Psychology Press, 1993.
- [25] 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.
- [26] P. M. Fitts, "The information capacity of the human motor system in controlling the amplitude of movement." *J. of Exp. Psychology*, vol. 47, no. 6, 1954.
- [27] P. M. Fitts, "The information capacity of the human motor system in controlling the amplitude of movement." *J. of Exp. Psychology*, vol. 47, no. 6, 1954.
- [28] 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.
- [29] 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.
- [30] B. Gates. (2009) ICTD Keynote Address. [Online]. Available: {http://ictd2009.org/event_footage/videos/ICTD2009_Bill_Gates_Keynote.mp4}
- [31] W. Gibson, "The science in science fiction," *Talk of the Nation*, 1999.
- [32] R. M. Groves, F. J. Fowler, M. P. Couper, J. M. Lepkowski, E. Singer, and R. Tourangeau, *Survey Methodology*. Wiley-Interscience, 2004.

- [33] A. Hartemink, “Banjo: Bayesian network inference with java objects,” <http://www.cs.duke.edu/~amink/software/banjo>.
- [34] C. Hartung, Y. Anokwa, W. Brunette, A. Lerer, C. Tseng, and G. Borriello, “Open data kit: Building information services for developing regions,” in *Proc. ICTD*, 2010.
- [35] D. Heckerman, D. Geiger, and D. M. Chickering, “Learning bayesian networks: The combination of knowledge and statistical data,” *Machine Learning*, vol. 20, no. 3, pp. 197–243, 1995.
- [36] J. M. Hellerstein, “Quantitative data cleaning for large databases,” United Nations Economic Commission for Europe (UNECE), 2008.
- [37] L. A. Hermens and J. C. Schlimmer, “A machine-learning apprentice for the completion of repetitive forms,” *IEEE Expert: Intelligent Systems and Their Applications*, vol. 9, no. 1, 1994.
- [38] I. E. G. (IEG), *Monitoring and Evaluation: Some Tools, Methods and Approaches*. Washington, DC: World Bank, 2004.
- [39] 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.
- [40] W. Johnson, H. Jelinek, L. Klotz Jr, R. Rao, and S. Card, “Bridging the paper and electronic worlds: the paper user interface,” in *Proceedings of the INTERACT’93 and CHI’93 conference on Human factors in computing systems*. ACM, 1993, pp. 507–512.
- [41] C. Jørgensen and B. Karlsmose, “Validation of automated forms processing: A comparison of teleform with manual data entry,” *Computers in biology and medicine*, vol. 28, no. 6, pp. 659–667, 1998.
- [42] D. W. King and R. Lashley, “A quantifiable alternative to double data entry,” *Controlled Clinical Trials*, 2000.
- [43] A. Kittur, E. Chi, and B. Suh, “Crowdsourcing user studies with mechanical turk,” in *Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*. ACM, 2008, pp. 453–456.
- [44] K. Kleinman, “Adaptive double data entry: a probabilistic tool for choosing which forms to reenter,” *Controlled Clinical Trials*, 2001.
- [45] D. Koller and N. Friedman, *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.
- [46] D. Lee and C. Tsatsoulis, “Intelligent data entry assistant for xml using ensemble learning,” in *Proceedings of ACM IUI*, 2005.

- [47] G. Little and Y. Sun, “Human ocr: Insights from a complex human computation process,” 2011.
- [48] R. Lorie, V. Riyaz, and T. Truong, “A system for automated data entry from forms,” in *Pattern Recognition, 1996., Proceedings of the 13th International Conference on*, vol. 3. IEEE, 1996, pp. 686–690.
- [49] J. Mao, R. Lorie, and K. Mohiuddin, “A system for automatically reading iata flight coupons,” in *icdar*. Published by the IEEE Computer Society, 1997, p. 153.
- [50] J. Martin, *Design of Man-Computer Dialogues*. Prentice-Hall, Inc., 1973.
- [51] K. Mate, B. Bennett, W. Mphatswe, P. Barker, and N. Rollins, “Challenges for routine health system data management in a large public programme to prevent mother-to-child hiv transmission in south africa,” *PLoS One*, vol. 4, no. 5, p. e5483, 2009.
- [52] R. McCarthy and T. Piazza, “Personal interview,” University of California at Berkeley Survey Research Center, 2009.
- [53] G. A. Miller, “The magical number seven, plus or minus two: Some limits on our capacity for information processing,” *Psychological Review*, vol. 63, no. 2, 1956.
- [54] T. P. Minka, “Expectation propagation for approximate bayesian inference,” in *Proceedings of the Conference in Uncertainty in Artificial Intelligence*, 2001.
- [55] K. Mullet and D. Sano, *Designing Visual Interfaces: Communication Oriented Techniques*. Prentice Hall, 1995.
- [56] K. L. Norman, “Online survey design guide,” http://lap.umd.edu/survey_design.
- [57] D. Oleson, A. Sorokin, G. Laughlin, V. Hester, J. Le, and L. Biewald, “Programmatic gold: Targeted and scalable quality assurance in crowdsourcing,” 2011.
- [58] C. Olston and E. H. Chi, “Scenttrails: Integrating browsing and searching on the web,” *ACM TOCHI*, vol. 10, no. 3, 2003.
- [59] W. J. Ong, *Orality and Literacy: The Technologizing of the Word*. Routledge, 2002.
- [60] T. S. Parikh, “Designing an architecture for delivering mobile information services to the rural developing world,” Ph.D. dissertation, University of Washington, Seattle, WA, USA, 2007.
- [61] T. S. Parikh, K. Ghosh, and A. Chavan, “Design studies for a financial management system for micro-credit groups in rural India,” in *Proc. Conference on Universal Usability*, 2003.
- [62] T. Parikh, P. Javid, *et al.*, “Mobile phones and paper documents: evaluating a new approach for capturing microfinance data in rural india,” in *Proceedings of the SIGCHI conference on Human Factors in computing systems*. ACM, 2006, pp. 551–560.

- [63] S. Patnaik, E. Brunskill, and W. Thies, “Evaluating the accuracy of data collection on mobile phones: A study of forms, sms, and voice,” in *ICTD*, 2009.
- [64] S. Patnaik, E. Brunskill, and W. Thies, “Evaluating the accuracy of data collection on mobile phones: A study of forms, sms, and voice,” in *ICTD*, 2009.
- [65] A. Ratan and M. Gogineni, “Cost realism in deploying technologies for development,” 2008.
- [66] A. Ratan, S. Chakraborty, K. Toyama, P. Chitnis, K. Ooi, M. Phiong, and M. Koenig, “Managing microfinance with paper, pen and digital slate,” *Proc. Info. & Comm. Tech. and Development (ICTD 2010)*, London, IEEE, 2010.
- [67] R. A. Reynolds-Haertle and R. McBride, “Single vs. double data entry in cast,” *Controlled Clinical Trials*, vol. 13, no. 6, 1992.
- [68] R. A. Reynolds-Haertle and R. McBride, “Single vs. double data entry in cast,” *Controlled Clinical Trials*, vol. 13, no. 6, 1992.
- [69] E. Saund, “Scientific challenges underlying production document processing,” in *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, vol. 7874, 2011, p. 1.
- [70] J. C. Schlimmer and P. C. Wells, “Quantitative results comparing three intelligent interfaces for information capture,” *Journal of Artificial Intelligence Research*, vol. 5, 1996.
- [71] Y. Schwartzman and T. Parikh, “Using cam-equipped mobile phones for procurement and quality control at a rural coffee cooperative,” *MobEA V: Mobile Web in the Developing World*, 2007.
- [72] S. Scribner and M. Cole, *The Psychology of Literacy*. Harvard University Press, 1981.
- [73] A. Sears and B. Shneiderman, “Split menus: effectively using selection frequency to organize menus,” *ACM Trans. Comput.-Hum. Interact.*, vol. 1, no. 1, pp. 27–51, 1994.
- [74] C. Shirky, *Cognitive Surplus: Creativity and Generosity in a Connected Age*. Penguin, 2010.
- [75] G. Singh, L. Findlater, K. Toyama, S. Helmer, R. Gandhi, and R. Balakrishnan, “Numeric paper forms for ngos,” in *Information and Communication Technologies and Development (ICTD)*, 2009 International Conference on. IEEE, 2009, pp. 406–416.
- [76] S. L. Smith and J. N. Mosier, “Guidelines for designing user interface software,” 1986.
- [77] 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.

- [78] UNICEF, “The state of the world’s children 2008: child survival,” 2008.
- [79] J. Warren and P. Bolton, “Intelligent split menus for data entry: a simulation study in general practice medicine,” *J Amer Med Inform Assoc*, 1999.
- [80] J. Warren, A. Davidovic, S. Spenceley, and P. Bolton, “Mediface: anticipative data entry interface for general practitioners,” in *Proceedings of OzCHI*, 1998.
- [81] W. Willett, “Scented widgets: Improving navigation cues with embedded visualizations,” *IEEE Transactions on Visualization and Computer Graphics*, 2007.
- [82] J. O. Wobbrock, E. Cutrell, S. Harada, and I. S. MacKenzie, “An error model for pointing based on fitts’ law,” in *Proceeding CHI*, 2008.
- [83] Y. Yu, J. A. Stamberger, A. Manoharan, and A. Paepcke, “Ecopod: a mobile tool for community based biodiversity collection building,” in *JCDL*, 2006.
- [84] S. Zhao and Z. Wang, “A high accuracy rate commercial flight coupon recognition system,” in *Document Analysis and Recognition, 2003. Proceedings. Seventh International Conference on.* IEEE, 2003, pp. 82–86.