Collaborative Development of Information Integration Systems

AnHai Doan, Robert McCann, Warren Shen
University of Illinois
{anhai, rlmccann, whshen}@cs.uiuc.edu

Abstract

Developing information integration systems today is largely done by hand, in a very labor intensive and error prone process. In this paper we describe MOBS, a conceptually new solution to this problem. MOBS employs *mass collaboration*. It asks the users of an information integration system to "pay" for using the system by answering simple questions, then uses those answers to further develop the system. This way an enormous burden of system development is lifted from the system builders and spread "thinly" over a multitude of users. We present both simulation and real-world experiments which suggest that MOBS can build systems accurately and efficiently. Finally we discuss the potential applications of MOBS beyond the information integration context.

1 Introduction

The rapid growth of distributed data has generated much interest in building information integration systems (or "II systems" for short). An II system provides users with a *global query interface* to a multitude of data sources, thus freeing them from manually querying each individual source. Figure 1 shows an II system over several bookstore sources. Given a user query formulated in the global query interface, the system uses a set of *semantic mappings* to translate it into queries over *source query interfaces*. Those queries are then executed and the combined results are returned to the user.

Much research has been conducted on information integration, in both the AI and database communities (e.g., (Garcia-Molina *et al.* 1997; Levy, Rajaraman, & Ordille 1996; Friedman & Weld 1997; Lambrecht, Kambhampati, & Gnanaprakasam 1999; Knoblock *et al.* 1998; Arens, Hsu, & Knoblock 1996; Chen *et al.* 2000)). Significant progress has been made, but despite this progress, today building II systems is still carried out largely by hand, in an extremely labor intensive and error prone process. The advent of languages and mediums for exchanging semi-structured data, such as XML, OWL, and the Semantic Web, will further accelerate the needs for II systems and exacerbate the above problem. Thus, it has now become critical to develop techniques for efficiently developing II systems.

In this paper we describe the MOBS approach to address the above problem. The basic idea underlying our approach is to ask the *users* of an II system to *"pay"* for using it by answering relatively simple questions, then use those answers to further build the system. This way an enormous burden of system development is lifted from the system builders and spread "thinly" over a mass of users. MOBS therefore is shorthand for "Mass Collaboration to Build Systems".

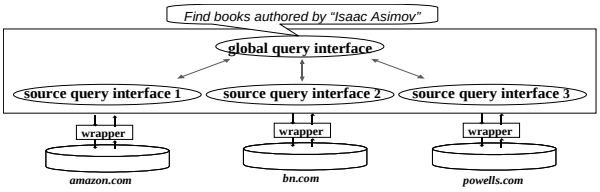

Figure 1: An information integration system in the book domain.

Mass collaboration is not new. It has been employed successfully in open-source software (e.g., *Linux*), product reviews (e.g., *amazon.com*), collaborative filtering (Resnick *et al.* 1994), tech support websites (Ramakrishnan 2001), marking up online images (e.g., *espgame.org*), trust management (Agrawal, Domingos, & Richardson 2003), and building collective knowledge bases (Richardson & Domingos 2003). However, as far as we know, no work has considered applying it to the information integration context.

In this paper, we propose to apply mass collaboration to build II systems. We believe the approach could be applicable to a broad variety of settings, including enterprise intranets, scientific domains, and the Web. For example, within an organization, the employees can collaboratively build systems that integrate organizational data. Bioinformatists can collaboratively build II systems over online bioinformatics sources. Several volunteers in a particular Web domain can deploy a system that integrates sources in that domain, by constructing an initial system shell, putting it on the Web, and asking the community to collectively maintain and expand it. This way, the system can be developed at very little cost to any particular entity, but at great benefits for the entire community.

As described, mass collaboration has the potential to dramatically reduce the cost of building II systems and spread their deployment in many domains. But it also raises many challenges.

- The first challenge is how to adapt information integration tasks (such as query interface recognition and matching) to mass collaboration. In the current framework we propose to decompose these tasks into Boolean statements whose truth values can be verified quickly by users.
- The second challenge is how to entice users to provide answers to the above statements. Virtually all prior works in mass collaboration rely on volunteers. However, this is problematic because it does not provide any guarantees on how quickly or in which quantity we can obtain user answers for the mass collaboration process. We propose several solutions which provide such guarantee, under some reasonable and natural conditions. The key idea

behind our solution is to leverage the user population of other well-established or "monopoly" applications in the information integration environment.

- The third challenge is how to handle malicious users and successfully combine multiple potentially noisy user answers into correct answers for the Boolean statements. In this paper we provide a simple but empirically effective solution to this problem. (In a forthcoming paper we describe a probabilistic framework based on dynamic Bayesian networks to address this problem. The framework subsumes the current solution and provides theoretical guarantees for most common mass collaboration cases.)
- The fourth, and final, challenge is how to evaluate our framework. Traditionally it has been very difficult to evaluate mass collaboration research, since such research requires the deployment and evaluation of some real-world applications that involve potentially a large number of people. We believe this has been a key bottleneck for mass collaboration research to take off. In this paper we propose a solution to this problem. Since the only real-world applications that most academic researchers could control reside within their Computer Science department, we propose to start with those applications, and with students and users in the departments.

The rest of the paper is organized as follows. First we develop a general MOBS solution to building II systems (Section 2). Then we apply it to construct and deploy an II system in the book domain (Section 3). Next, we present experiments with both simulation and the above deployed system (Section 4). The experiments show that MOBS can build systems accurately and efficiently. In particular, it took a population of 132 users 1.6 answers per user per day, for 12 days, to rapidly build a functional II system over 10 real-world bookstore sources.

Due to space limitation, this paper only briefly touches on various aspects of our work. For more detail, please see the full technical report (McCann *et al.* 2005).

2 The MOBS Solution

In this section we describe the basic ideas behind MOBS. We start with technical challenges such as handling malicious users and combining user feedback. Then we discuss incentive models for users to give feedback.

2.1 Technical Framework

We describe MOBS using a scenario of recognizing query interfaces. Suppose we have built a small II system over three bookstore sources, and now are considering adding a fourth source, which resides at the URL *book.com*.

The HTML page of an URL such as *book.com* often contains many *input forms*. Some of these forms are indeed query interfaces into bookstores, but many others are not. They can be interfaces to book review sites, or forms asking for email addresses, zipcodes, and buyer information.

Hence, given each input form F of page *book.com*, a crucial problem is to recognize if F is indeed a bookstore query interface (Chang *et al.* 2003). In what follow we describe the MOBS solution to this problem.

Decomposing the Problem: To apply MOBS to any task P, we begin by translating P into a series of simple "yes"/"no" questions, such that knowing the answers of the questions amounts to solving the problem. In this specific case, the problem of recognizing form F can be translated into a single question "is F a query interface to a bookstore?". In Section 3 we describe the attribute-matching task which needs to be translated into multiple questions.

For tasks that we cannot yet fully decompose, even decomposing just a part of it may already be very helpful. For example, many wrapper construction techniques require the system admin to manually label a set of data fragments to provide training data (Kushmerick, Weld, & Doorenbos 1997). This labeling process can be converted into a series of statements (e.g., "(217) *is an area code.*"), and be carried out by mass collaboration of users, thus relieving the system admin of this burden.

Soliciting User Answers: Next, we solicit user answers to the question as follows. When a user queries the current system (e.g., with "find all books written by Isaac Asimov"), we make the user "pay" for the query result, by displaying the question together with form F highlighted. Figure 2 shows such a question. Since form F is designed for *human consumption*, the user should be able to quickly answer "yes" or "no". Only after the user has answered do we display the results of his or her query. Note that form F has not yet been used to produce results for the query of this user (as it may still have to be verified by other users). In Section 2.2 we discuss other ways to solicit user answers.

Assessing Users' Reliability: To handle malicious or ignorant users, we compute a *weight* for each user U that measures the quality of U's feedback, by occasionally asking *evaluation questions*: those whose answers we already know. For example, we can show users the forms that we encountered during the construction of the current system, or forms from a set-aside training set, and ask if they are indeed bookstore query interfaces. We set the weight of U to be zero if U has answered fewer than k evaluation questions (where k is a given threshold); otherwise we set it to be the fraction of U's answers to the evaluation questions that is correct.

For any single user we randomly mix the evaluation questions with teaching questions (i.e., those used to get feedback on unverified query interfaces) to keep the identities of evaluation questions secret. Once a user weight has been computed we stop evaluating the user (we are exploring the option of continual evaluation). We call a user *trustworthy* if his or her weight is above a threshold w (currently set at 0.65) and *untrustworthy* otherwise.

Combining User Answers: As users query the current system, the question Q "is form F a bookstore query interface?" receives a steady stream of "yes"/"no" answers. We monitor this stream, and terminate it (i.e., no longer solicit answers for Q) as soon as a stopping criterion is reached. Suppose Q has received a total of n answers from trustworthy users. A possible stopping criterion is (a) the number of majority answers (either "yes" or "no") reaches $n * 0.65$ and n exceeds t_1, or (b) n exceeds t_2, where $t_1 < t_2$ are given.

Figure 2: A sample question that MOBS asks users in the interface recognition task.

Condition (b) ensures that asking questions about form F eventually terminates. Once stopped, we assign the majority answer as the final answer for Q.

In (McCann *et al.* 2005) we describe a probabilistic framework based on dynamic Bayesian networks to address the problem of assessing users' reliability and combining their answers. The framework subsumes the current solution and provides theoretical guarantees for most common mass collaboration cases.

2.2 Incentive Models

We now discuss how to entice users to provide feedback. First, we discuss two models of prior work, and show how they can be adapted to our context. Then we propose two novel models that address some important limitations of prior models.

Volunteer/Delayed-Gratification Model: The vast majority of works and real-world applications in mass collaboration rely on users' volunteering to provide feedback. Example include Bibserv (Agrawal, Domingos, & Richardson 2003), *openmind.org*, *amazon.com*, and *epinions.com*. Such volunteering is typically motivated by some (often delayed) benefits or self-gratification. To entice volunteering, many applications also institute a mechanism of distributing credits to users. This model clearly also applies to our II context. For example, the employees of an organization may understand the long-term benefits of providing feedback and are willing to help build systems over organizational data.

Instant Gratification Model: Some recent works propose to provide users with *instant* benefits as soon as they volunteer some feedback, on the ground that such immediate gratification may motivate users to volunteer more feedback. The instant benefits could be value-added services (McDowell *et al.* 2003) or game play experience, which helps implicitly generate user feedback (e.g., *espgame.org*).

This model can be easily adapted to our II context. For example, an II system in the book domain would produce the usual list of books in response to a user query, but would also indicate that *even more details* about the books in the list are available, if the user is interested. To get to those details, however, the system needs the user to answer a simple question. Thus, the user has a strong incentive to supply some simple feedback, which provides instant gratification in terms of more details about the answers.

Limitations of Prior Incentive Models: The above two models rely on users' being *willing to provide feedback*. This is problematic because it provides no guarantee on how quickly we can build II systems. To address this problem, we propose to also consider two models in which users are *forced to participate*, hence guaranteeing a steady stream of user answers for the mass collaboration process.

Monopoly/Better-Service Model: If the application under consideration provides a monopoly service that its users need, or a service in much better ways than its competitors, then it can leverage that advantage to "force" the users to "pay" for using its service, where the "payment" is a bit of user knowledge. For example, consider an internal web service that the employees of a company must access daily. Since this internal web service is in a monopoly situation, it can use MOBS to collect "payments" from the employees, then leverage these "payments" to improve its services.

As another example, imagine that *Google* asks each of its users to answer 1-2 relatively simple questions per day. Suppose the questions are asked at a predictable time (preferably only at the start of a user day or session). Suppose also that each user has the ability to *defer* question answering to a more appropriate time (in effect owing the system the answers). Under these conditions, it is possible that many users still use *Google*, not defecting it to another search engine service. *Google* can then leverage users' answers to mark up Web pages quickly, thus greatly facilitate integrating Web data, and improve its services.

Helper Application Model: An II system can also leverage the users of *another application* – which we call a *helper application* – to provide feedback. For example, within a company, an II system can leverage the users of an internal web service (as described above). For this model to work, we believe the helper application must be providing a monopoly or better service. The following example illustrate this model:

Example 2.1 Many companies must constantly integrate and monitor data on the websites of their competitors. This is especially crucial for online stores such as *amazon.com*, *bestbuy.com*, etc. This task is clearly very labor intensive and time critical, hence companies can use MOBS to enlist their employees in order to reduce cost and improve timeliness. In this context, helper applications (which are monopoly services) can be online internal company services that employees must use, such as payroll systems, bulletin boards, internal newsgroups, etc.□

Helper applications are in a sense similar to the initial capital invested in the real world in a business, until the business grows to the extent that it can sustain itself.

We note that many Web services that are traditionally free have also started to charge users "payment". The news magazine *salon.com* for example asks users to watch and click on an ad before allowing them to read its articles. The peer-to-peer community has also taken steps to eliminate "freeloaders" from file-sharing services. In several schemes (e.g., (Yang & Garcia-Molina 2003; Golle *et al.* 2001)), users are forced to "pay" for their own downloads by cooperating to satisfy the requests of other peers (e.g., con-

tribute new files, or relay existing files). Likewise, some systems (e.g., (Kamvar, Schlosser, & Garcia-Molina 2003)) give preferential treatment to cooperative peers, in a sense allowing individuals to "pay" for better service.

Summary: From the above discussion, it appears that incentive models can be considered based on (a) forced participation vs. not forced participation, (b) delayed gratification vs. instant gratification, (c) monopoly vs. not monopoly, and (d) same application vs. different application[1]. It would be an interesting future research direction to explore and evaluate models along these dimensions.

2.3 Starting MOBS Applications

We now discuss how a system can get started with MOBS and can use the above incentive models. If the system is already well-established, with a stable user population (such as the II system ENTREZ ($ncbi.nlm.nih.gov/Entrez$) in bioinformatics, or even *CiteSeer* and *Google*), and we simply want to expand its capabilities or coverage, then the system can use any combination of the four incentive models described above, depending on its specific situation.

In contrast, if we want to build an II system from scratch, then it cannot use the monopoly/better service model, but possibly the remaining three ones. The best solution is probably to start with the volunteer/delayed gratification model and the helper application model to build up the system. Once the system has obtained a certain amount of coverage and capabilities, the instant gratification model as described earlier can be quite useful. Finally, when the system has reached a position in which it provides the best or only service of a kind, it can also leverage the monopoly/better service model to solicit user feedback.

3 Applying MOBS to Build II Systems

We now describe applying the general MOBS solution in Section 2 to build an II system over online bookstores, using the helper application model. We choose the book domain because it is easy to understand and is well-known in the II literature (Chang *et al.* 2003).

Recognizing Query Interfaces: We started with a simple global query interface that has four attributes: $title, author, genre$, and $isbn$. Next, we selected 30 book-related URLs from a Deep Web source repository (Chang *et al.* 2003) (another option is to use a focused crawler to discover such URLs). These URLs have a total of 55 input forms, out of which only 19 are bookstore query interfaces. Our first task was then to recognize the bookstore query interfaces.

As the baseline solution, we trained a learning method on another set of 24 book-related URLs (where the forms were manually labeled), then applied it to classify the 55 above forms. We experimented with SVM, Naive Bayes, TF/IDF methods (details omitted due to space reasons), and found TF/IDF to be the best, achieving 70% precision and 89% recall (note that we have manually annotated all 55 forms for evaluation purposes).

[1]We thank an anonymous reviewer for suggesting these dimensions.

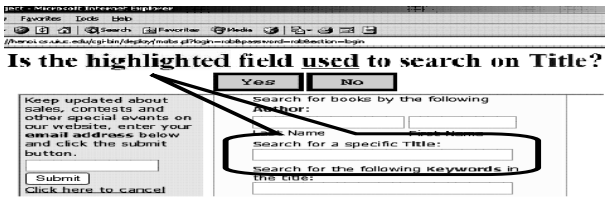

Figure 3: A sample question that MOBS asks users during the attribute matching task.

In the next step, we applied MOBS on the result of the learning method, to further improve accuracy. Since we started the book II system from scratch, we considered using a helper application (as discussed in Section 2.3). The challenge was to find a helper application under our control.

Our solution was to leverage the homepage of a course that one of us was teaching. We placed MOBS on that course homepage, so that when a student accessed a page on the homepage, he or she had to answer a simple question "is form F a bookstore query interface?" (as discussed in Section 2). However, we limited the load of each student to four questions per day. (Most students ended up answering on average fewer than two questions per day.)

We limited the questions to only the 24 forms that TF/IDF had classified as positive. Within 5 days, the students had classified all 24 forms with perfect accuracy, and identified 17 of them to be bookstore query interfaces.

Matching Attributes of Interfaces: For each of the 17 identified bookstore interfaces, our next step was to match its attributes (i.e., input fields) with those of the global interface. Consider matching an attribute a of form F. First we decompose this problem into four questions "$a = title$", "$a = author$", "$a = genre$", and "$a = isbn$", representing all possible matching scenarios.

Next, we used the same MOBS mechanism on the course homepage to solicit answers to the above questions. Figure 3 shows a sample question. If the answer for a question converges to "yes", we stop and return the match encoded by that question. If all four questions converge to "no", then we match a to the special attribute $none$.

We ran this experiment for 7 days, until the semester ended in December 2003. At that time, the students had successfully matched 10 out of 17 interfaces, with 97% precision and recall (again, here we have manually annotated all interfaces for evaluation purposes, see the experiment section).

Translating User Queries: Our next task was to find out how to translate a user query, leveraging the above matches. Specifically, suppose a user has filled in the field $author$ of the global interface with "Isaac Asimov", how should we fill in corresponding fields at a source query interface F?

Currently we use a simple but effective rule-based method for filling such fields. We examined 30 bookstore interfaces to create a rule list. Examples of rules applying to the above case include "if $author$ matches a single field of F then fill that field with "Isaac Asimov"", and "if $author$ matches two fields of F, then fill them with "Isaac" and "Asimov", respectively". We found that since bookstore interfaces tend

P1 – Uniform [0,1]	P2 – Uniform [0.3,0.7]	P3 – Uniform [0.5,0.9]
P4 – Bell [0,1]	P5 – Bell [0.3,0.7]	P6 – Bell [0.5,0.9]
P7 – Bimodal {0.2,0.8}		
P8 – 90% Uniform [0,0.4], 10% {0.8}		
P9 – 10% {0.1}, 50% Uniform [0.5,0.7], 40% Uniform [0.8,1]		
P10 – 10% {0.3}, 90% Uniform [0.7,1]		

Figure 4: The ten user populations for simulations with MOBS. "Uniform [0,1]" means randomly assigning a reliability score from [0,1] to each user. "Bell [0,1]" means the scores follow a bell distribution over [0,1]. "Bimodal {0.2,0.8}" means 50% of users receive score 0.2 and the remaining 50% score 0.8. Finally, "10% {0.1}, 50% uniform [0.5,0.7], 40% Uniform [0.8,1]" means 10% of the users have score 0.1, 50% receive score uniformly over [0.5,0.7], and the remaining 40% uniformly over [0.8,1].

to be similar, a simple set of rules cover the overwhelming majority of translation cases, an observation made also by some recent work (Chang *et al.* 2003).

Given an interface F and its semantic matches into the global interface (as created by MOBS), the II system consults the above rule list to translate the user query. The system consults the builder if it cannot find an applicable rule.

Putting It All Together: Atfer 12 days of deploying MOBS in December 2003, we now have an II system over 10 real-world bookstore sources. The system is *functional* on the Web and can already perform useful querying (the system URL is withheld due to blind review). Most importantly, the workload of each user was minimal (about 16 total answers per user, or 1.6 per source), and the workload of us as the system builders was also minimal. This workload consists of collecting the URLs of the sources and creating the translation rule list, a total of less than a day of work, which does not have to be repeated. This workload will be greatly amortized in Spring 2004, when we plan to leverage another course homepage to expand the above system to cover hundreds more bookstore sources. We also plan to apply MOBS to remaining II problems, such as wrapper construction and the query translation problem itself.

4 Empirical Evaluation

We now describe extensive experiments with MOBS, starting with simulation, then with the above II system in the book domain.

4.1 Experiments with Simulation

We generated ten synthetic user populations $P_1 - P_{10}$. Each population has 500 users, and each user has a reliability score. A *reliability score* of 0.6 means that on average the user answers 6 questions correctly out of 10. The populations have their reliability scores generated according to *uniform, bell, bimodal, and mixed* distributions, with more detail in Figure 4. The ten populations thus represent a broad range of user populations that we expect to commonly occur in practice.

For each population we simulated its interaction with the II system described in Section 3. For example, given population P_1 (Figure 4), we simulate its handling the interface recognition and matching tasks (Section 3). For the interface recognition task, for example, we simulated each user in P_1

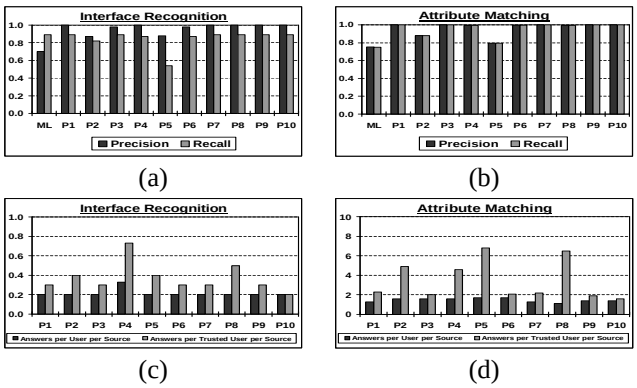

Figure 5: Accuracy (top) and average workload (bottom) over a broad range of populations.

accessing the II system and providing answers on whether a particular interface is a bookstore query interface.

Once the simulation for a task has completed, we compute the *accuracy* of MOBS on that task, using the standard IR notion of precision and recall (Salton & McGill 1983). For instance, for interface recognition we compute precision to be the fraction of interfaces that MOBS correctly identifies over the total number of interfaces for which MOBS makes a prediction.

High Accuracy: The top row of Figures 5 shows the accuracy of MOBS for the above two tasks. For instance, for interface recognition, Figure 5.a shows the precision (black bar) and recall (gray bar) for the learning method (denoted as "ML", see Section 3), and then MOBS for all ten simulated populations.

The figure shows that MOBS significantly outperforms the learning method, and more importantly, reaches the highest possible precision (100%) and recall (89%) in all but two populations, P_2 and P_5. MOBS only reaches 89% recall (same as that of the learning method) because we currently apply MOBS only to the interfaces classified as positive by the learning method, as explained in Section 3.

The populations P_2 and P_5 are quite "unreliable" in that they contain only a few "good" users, at the highest reliability score of 0.7 (see Figure 4). Hence, the convergence criterion used for the experiments in Figure 5 is sufficient for the other eight "more reliable" populations (to achieve the maximum accuracy) but is inadequate for P_2 and P_5. When run with tighter convergence criteria, these two populations achieve near perfect accuracy, while incurring only a small increase of less than 2 answers in the workload of each user.

Figures 5.b shows similar results for MOBS' accuracy on attribute matching.

Low Workload: The second row of Figure 5 shows the average workload (i.e., the number of answers) per user per task. Figure 5.c for example shows the workload for interface recognition. In this figure, the two bars for each population show the average number of answers per user, and per *trusted* user, respectively. The latter is interesting because most of the work is in effect spread over the trusted users, so we would like to know if their workload stays reasonable.

Name	Helper Application	Duration & Status	Current Progress	Precision	Recall	Avg User Workload
Interface Recognition	DB course website, 132 undergrad students	5 days, Completed	Completed 24/24 interfaces, Found 17 bookstore interfaces	1.0 (0.7 ML)	0.89 (0.89 ML)	7.4 answers
Attribute Matching	DB course website, 132 undergrad students	7 days, Stopped	Completed 10/17 interfaces, Matched 65 total attributes	0.97 (0.63 ML)	0.97 (0.63 ML)	12.5 answers

Table 1: MOBS results with the deployed II system in the book domain.

Figure 5.c shows that for the 500 users, each user answers on average only 0.2 question in order to classify a form. The workload for trusted user is only slighly higher, from 0.2 to 0.74. Figure 5.d shows that each user must answer at most 6.8 questions to match the attributes of an interface. In contrast, for each source interface, a single system builder would have to match on average 9 attributes with 4 global-interface attributes, in effect performing a workload of answering 36 questions. These results therefore demonstrate that MOBS indeed can spread the construction burden thinly over a mass of users, such that each of them has to do only minimal work.

Effects of Population Size: We have also experimented with population size varying from 50 to 10,000 users, over numerous population topologies. We observed that matching accuracy remains stable across varying sizes. In all cases the time it took to manage the mass collaboration mechanism was negligible. We further observed that, as expected, the number of answers required per user to reach convergence decreases linearly as the population size increases. This suggests that our approach can scale up to very large populations in all important performance aspects, and that in a large population the already minimal workload per user will be significantly reduced further.

4.2 Real-World Deployment

We have described building an II system in the book domain (Section 3). In what follows we describe our deployment results and experience.

System Performance: Table 1 describes MOBS's performance on the two tasks of interface recognition and attribute matching. The results show that MOBS achieves very high accuracy in practice. On interface recognition, it obtains the highest possible 100% precision and 89% recall (MOBS has a recall ceiling due to the fact that it is currently applied only to forms that are classified as positive by learning techniques). The value (0.7 ML) indicates that the learning technique we used achieves 70% precision in this case.

On interface matching, MOBS achieves 97% precision and recall, misclassifying only 2 out of 65 attributes. These two attributes are "keyword", presenting an ambiguous case to the users. MOBS also significantly outperformed the employed learning technique, as the columns "precision" and "recall" show.

Finally, the last column of Table 1 shows that MOBS incurred very little workload on each user (e.g., only 1.25 answers per user per interface, on the interface matching task).

User Performance: The deployment clearly shows that users can handle the cognitive load of answering questions in these applications. It also shows that most users have a minimal participation. In the deployment, we gave users the options of answering at most 4 questions per day, or answering an unlimited number of questions. Most users selected the first option, and indeed answered on average less than 2 questions per day. But several users were more active and answered on average 4-7 questions per day. Indeed, we observed that user participation follows the similar Zipfian distribution, though most users made meaningful contributions to the task at hand.

MOBS was able to detect 11-23 untrusted users per each task, and ignore their feedback. The accuracy of untrusted users's answers is low at 35-48%, while that of trusted users is 76-81%. We also observed that the accuracy of trusted users' answers remains stable over time, thus supporting the synthetic user model presented in Section 4.1.

4.3 Summary

Our extensive simulation results show that MOBS can reach perfect or near perfect accuracy over a broad range of populations, with minimal workload per user, and that the workload is decreased significantly even further as the population size increases. The real-world deployment shows that MOBS can achieve very high accuracy with minimal user workload, and that real users can handle the cognitive load of the questions and quickly answer most of them correctly.

5 Related Work

Mass Collaboration: As far as we know, this is the first work to apply mass collaboration to build information integration systems. Mass collaboration has been studied for many applications, including building sense tagged corpora on the Web (Mihalcea & Chklovski 2004), building tech support websites ((Ramakrishnan 2001), *quiq.com*), knowledge bases (Richardson & Domingos 2003; Richardson, Aggrawal, & Domingos 2003), user trust networks (Agrawal, Domingos, & Richardson 2003), word sense disambiguation (ope a; b), and error correction on *CiteSeer* (Lawrence, Bollacker, & Giles 1999). These works ask users to contribute facts and rules in some specified language. This poses a difficult challenge because potentially *any* fact or rule being contributed constitutes a statement whose validity must be checked. Thus, the number of statements can be very high (potentially in the millions) and checking them can be very difficult in practice. In contrast, the number of statements in an information integration setting is comparatively much smaller and thus potentially much more manageable. As such, we believe building information integration systems can be a "killer app" for mass collaboration.

Information Integration: From information integration perspective, the prohibitive cost of constructing II systems has motivated numerous works on reducing the cost of *specific* II tasks, such as schema matching (Rahm & Bernstein 2001) and wrapper construction (e.g., (Kushmerick, Weld, & Doorenbos 1997; Ashish & Knoblock 1997)). But there has been virtually no work on reducing the cost of the *whole construction process*. Our current work can be viewed as attempting a systematic solution to this problem.

Semantic Web: Our work shares with Semantic Web research the problems of enticing users to provide feedback and combining varying-quality information. In the spirit of mass collaboration as discussed in this work, we believe it would also be interesting to explore mass collaboration for marking up data on the Semantic Web. Virtually all current works ask the *owner* of a Web page to mark up the page's data. This approach however often leads to a catch-22 situation: owners do not mark up pages because they do not see any interesting applications, but applications are not developed because there is no marked-up data. To break this catch 22, it would be interesting to explore a conceptually opposite solution: instead of asking the page owner (i.e., the *producer*), we ask the people who visit the page (i.e., the *consumers*) to help mark it up.

Machine Learning: Our work is most related to active learning (e.g., (Sarawagi & Bhamidipaty 2002; Lau *et al.* 2004)), but differs in important aspects. First, the goal of active learning has typically been to learn a better *classification model* or learn to perform a task (Lau *et al.* 2004), whereas our goal is to improve the predictions made by an *integration tool*. Second, active learning usually enlists users to provide extra *training data*, by labeling selective examples. In contrast, we enlist users for a broader range of tasks: providing training data, eliciting domain knowledge, and verifying tool predictions.

Third, active learning does not place a limit on the complexity of the work that the user does, whereas in our context most users are non-technical, raising the challenge of being able to ask only relatively simple questions, yet also making the most out of them. The final, and most important, distinction is that active learning has usually involved only a *single* user, whereas our work involves *multiple* users, and thus must address the challenge of combining their noisy answers. MOBS therefore is a form of *collective active learning*.

In our work each user can be viewed as a learner, which has been trained and is ready to make predictions. The problem of how to learn the accuracy of learners and combine a very large number of learners in an efficient way raises interesting learning issues that have not been considered before, and thus warrants further study.

Others: Numerous works have developed rule- or learning-based automatic solutions for schema matching, a fundamental problem in information integration (Rahm & Bernstein 2001). In contrast, our current work leverages the feedback of a multitude of users to find the mappings. To our knowledge, this is the first work on schema matching in this direction.

Our work here is also related to autonomic systems in that mass collaboration-assisted II systems also exhibit autonomic properties similar to self-healing and self-improving. The key difference is that traditional autonomic systems achieve these properties by observing the external environment and adjusting themselves appropriately. In contrast, our systems are observed by the external environments (i.e., the multitude of users) and then are adjusted by them accordingly.

6 Conclusion and Future Work

In this paper we have proposed a mass collaboration approach to efficiently build II systems. The basic idea is to shift the enormous development cost from the producers of the system to the consumers, but spreading it "thinly" over a large number of consumers. We presented MOBS, a mass collaboration framework to build II systems, then applied MOBS to build one such system on the Deep Web. We described experiments which suggest that MOBS can build systems accurately and efficiently, thus demonstrating the promise of our approach.

Our future work includes developing more sophisticated models of MOBS, studying the formal properties of the models, investigating learning issues within the MOBS context, and developing a broad variety of real-world applications, to further evaluate and validate the approach.

In addition, we plan to study how to design information integration systems such that MOBS can learn from traces of user activities. This way, MOBS can leverage the implicit user feedback whenever explicit feedback (as in the current framework) is unavailable or difficult to obtain.

Acknowledgment: We thank the anonymous reviewers for invaluable comments on this paper. This work is supported by NSF CAREER Award IIS-0347903 and NSF ITR Grant 0428168 to the first author.

References

Agrawal, R.; Domingos, P.; and Richardson, M. 2003. Trust management for the semantic web. In *Proc. of the Int. Semantic Web Conf.*

Arens, Y.; Hsu, C.; and Knoblock, C. 1996. Query processing in the SIMS information mediator. In Tate, A., ed., *Advanced Planning Technology*. AAAI Press.

Ashish, N., and Knoblock, C. 1997. Wrapper generation for semi-structured internet sources. *SIGMOD Record* 26(4):8–15.

Chang, K. C.-C.; He, B.; Li, C.; and Zhang, Z. 2003. Structured databases on the web: Observations and implications. Technical Report UIUCDCS-R-2003-2321, Department of Computer Science, UIUC.

Chen, J.; DeWitt, D.; Tian, F.; and Wang, Y. 2000. Niagaracq: A scalable continuous query system for internet databases. In *SIGMOD '00*.

Friedman, M., and Weld, D. 1997. Efficiently executing information-gathering plans. In *Proc. of the Int. Joint Conf. of AI (IJCAI)*.

Garcia-Molina, H.; Papakonstantinou, Y.; Quass, D.; Rajaraman, A.; Sagiv, Y.; Ullman, J.; and Widom, J. 1997.

The TSIMMIS project: Integration of heterogeneous information sources. *Journal of Intelligent Inf. Systems* 8(2).

Golle, P.; Leyton-Brown, K.; Mironov, I.; and Lillibridge, M. 2001. Incentives for sharing in peer-to-peer networks. In *WELCOM*.

Kamvar, S.; Schlosser, M.; and Garcia-Molina, H. 2003. Incentives for combatting freeriding on p2p networks. In *Euro-Par*.

Knoblock, C.; Minton, S.; Ambite, J.; Ashish, N.; Modi, P.; Muslea, I.; Philpot, A.; and Tejada, S. 1998. Modeling web sources for information integration. In *Proc. of the National Conference on Artificial Intelligence (AAAI)*.

Kushmerick, N.; Weld, D.; and Doorenbos, R. 1997. Wrapper Induction for Information Extraction. In *Proc. of IJCAI-97*.

Lambrecht, E.; Kambhampati, S.; and Gnanaprakasam, S. 1999. Optimizing recursive information gathering plans. In *Proc. of the Int. Joint Conf. on AI (IJCAI)*.

Lau, T.; Bergman, L.; Castelli, V.; and Oblinger, D. 2004. Sheepdog: Learning procedures for technical support. In *Proc. of the IUI Conf. (IUI)*.

Lawrence, S.; Bollacker, K.; and Giles, C. L. 1999. Distributed error correction. In *Digital Libraries 99 - The Fourth ACM Conference on Digital Libraries*. New York: ACM Press.

Levy, A. Y.; Rajaraman, A.; and Ordille, J. 1996. Querying heterogeneous information sources using source descriptions. In *Proc. of VLDB*.

McCann, R.; Shen, W.; Kramnik, A.; Varadarajan, V.; and Doan, A. 2005. Learning from multiple non-technical users to improve accuracy of data integration tasks. Technical Report TR-05-01, available from http://anhai.cs.uiuc.edu/home, Dept. of Computer Science, Univ. of Illinois.

McDowell, L.; Etzioni, O.; Gribble, S.; Halevy, A.; Levy, H.; Pentney, W.; Verma, D.; and Vlasseva, S. 2003. Evolving the semantic web with mangrove. Technical Report UW-TR-2003, Dept. of CSE, Univ. of Washington.

Mihalcea, R., and Chklovski, T. 2004. Building sense tagged corpora with volunteer contributions over the web. In *Current Issues in Linguistic Theory: Recent Advances in Natural Language Processing, Nicolas Nicolov and Ruslan Mitkov (eds)*. John Benjamins Publishers.

http://web.media.mit.edu/ push/omcs-research.html.

http://www.openmind.org/lectureslides.html.

Rahm, E., and Bernstein, P. 2001. On matching schemas automatically. *VLDB Journal* 10(4).

Ramakrishnan, R. 2001. Mass collaboration and data mining. Keynote address, KDD-01, www.cs.wisc.edu/ raghu/Kddrev.ppt.

Resnick, P.; Iacovou, N.; Suchak, M.; Bergstrom, P.; and Riedl, J. 1994. Grouplens: Collaborative filtering of netnews. In *Proc. of CSCW-94*.

Richardson, M., and Domingos, P. 2003. Building large knowledge bases by mass collaboration. Technical Report UW-TR-03-02-04, Dept. of CSE, Univ. of Washington.

Richardson, M.; Aggrawal, R.; and Domingos, P. 2003. Building the Semantic Web by mass collaboration. Technical Report UW-TR-03-02-05, Dept. of CSE, Univ. of Washington.

Salton, G., and McGill, M. 1983. *Introduction to Modern Information Retrieval*. McGraw-Hill.

Sarawagi, S., and Bhamidipaty, A. 2002. Interactive deduplication using active learning. In *KDD*.

Yang, B., and Garcia-Molina, H. 2003. Ppay: Micropayments for peer-to-peer systems. In *ACM CSS*.