Human OCR: Insights from a Complex Human Computation Process

Greg Little
MIT CSAIL
32 Vassar St. Cambridge, MA
glittle@gmail.com

Yu-An Sun
Xerox Research Center Webster
800 Phillips Road, Webster, NY
YuAn.Sun@xerox.com

ABSTRACT
Human computation is a growing research field. It holds the promise of humans and computers working seamlessly together to implement powerful systems, but this requires in depth knowledge about the communication barrier between man and machine. In this paper, we discuss the creation of a relatively complicated human computation process. We test our process on a set of real world data, and discuss the results. We also extract general human computation principles from this process, including the general merits of independent agreement, as well as the risks of piecework, where workers see only a subset of a larger task.

Author Keywords
human computation, Mechanical Turk, transcription

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

INTRODUCTION
In a human-computer interaction environment, we usually think of the human as being in charge, and the interface as trying to serve their needs for achieving a goal. However, in human computation situations, a computer algorithm is in charge, and it uses interfaces to communicate to humans what they should do. The humans are acting as component steps in a process. In this paper we present a relatively complicated human computation process for performing human OCR on hand-completed forms, and discuss design issues we encountered while building and evaluating the system.

After a discussion of related work, this paper will describe our problem, our solution, and an experiment we ran on our solution. Then we spend the rest of the paper discussing the results, including sharing some high level insights for future research topics that came from this work.

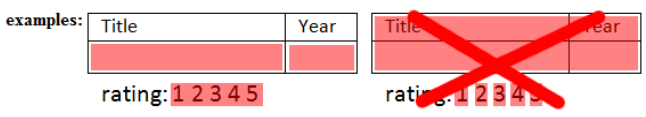
CHI 2011, May 7–12, 2011, Vancouver, BC, Canada.

RELATED WORK
This work falls into the broad category of human computation. Quinn et al. [7] give a good overview of human computation systems. This work is particularly interested in task design, and how human workers react to different tasks. These issues are explored in [3], [6] and [8]. This work is also an example of a relatively complicated human computation system, such as those explored in [2] and [4], using the TurKit toolkit [5].

PROBLEM
We want to perform OCR on handwritten forms. In particular, we want to take a set of scanned forms, and generate a spreadsheet of the data on the forms. The output spreadsheet should have a row for each scanned form, and a column for each field of data (e.g. a "first name" column, and a "zip code" column). We would also like to keep data private by not revealing too much to any of our workers.

SOLUTION
Our basic approach involves showing human workers small cropped windows of each form, each containing a small chunk of information that should be useless on its own (e.g. the word "HIV", without seeing who has it). Since our approach involves showing people cropped views of a form, we will also need some way to discover the location and size of each view. To support this task, we add an additional input to the system, which is a blank template form. Workers may view the entire template form without revealing anyone's private information. The template form is used in the first phase of a three phase process:

Phase 1: Drawing Rectangles
The first phase asks people to draw rectangles on the blank template form where writing would appear. Workers are also shown some positive and negative example rectangles over an example form:

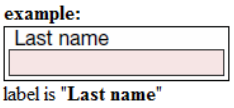


Because the form may have many fields, we tell workers "you don't have to do it all, just help". We only keep rectangles that multiple people draw in roughly the same place, and we show workers all the rectangles that previous workers have agreed upon. These rectangles act as extra examples of what to do, and also encourage workers to draw rectangles elsewhere on the form, so that everyone doesn't draw rectangles for the top few fields. In case all the rectangles are already drawn, we include a submit button labeled "All the fields are covered".

We pay 5 cents for each worker, and keep asking workers to add more rectangles for a set number of iterations (10 in our case). One could also imagine a stopping condition based on the "All the fields are covered" button.

Phase 2: Labeling Rectangles

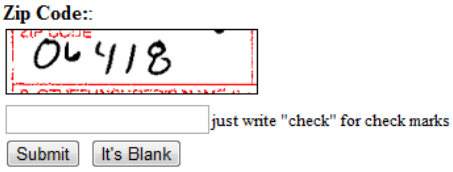
The first phase provides a set of rectangles covering the form. These rectangles mark all of the chunks of information that will appear in the columns of the final spreadsheet. What we need now is a heading for each column, i.e., a label for each rectangle. The interface for this task is fairly straightforward: we show a worker the form with a single rectangle highlighted on it. Then we ask for a label of the given rectangle. We also show this example:



We iteratively ask for labels (1 cent each) until two workers submit the same label. We were concerned that this might take too long, since the space of possible labels might be quite large given different abbreviations for a long label, and different ways to capitalize it. To foster convergence, we show each worker all the labels that previous workers have provided, so that they can simply copy one of these if it seems good enough.

Phase 3: Human OCR

The final phase of the process is the most straightforward. For each rectangle, on each scanned form, we show workers a cropped view of that form where the rectangle is, and prompt them to write what is written there:



We also show people the label for the rectangle (e.g. "Zip Code"), so that they have some context to help them decipher the handwriting (e.g. these must be five numbers). We also expand the rectangle a little bit, in case people write outside the lines. Note that the instruction: "just write 'check' for check marks" was added based on pilot experiments were workers were unsure what to write when they saw a check mark.

For this task, we do not show workers the results from previous workers, since we want *independent* agreement about the correct answer. Hence, we keep asking for more submissions (1 cent each) until two workers submit exactly the same response.

EXPERIMENT

We iteratively developed the solution above by testing each phase on a set of artificially created data, including real handwriting that we procured by hiring MTurk workers to print a fake form, fill it out with fake information, and scan it (50 cents each).

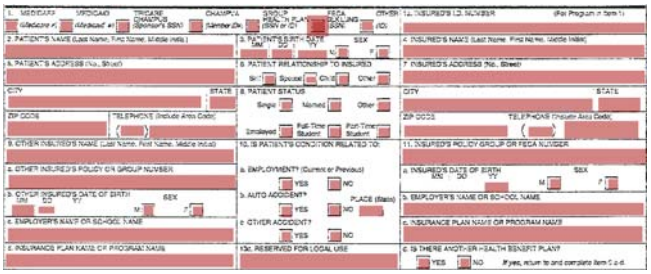
We tested the final process on a set of real forms with real data. We started with 30 scanned medical insurance claim forms, along with a blank template form. First, we removed all identifying information, like names and telephone numbers (we were not yet ready to test the privacy aspect of the system). Then, we cropped the forms, to remove form elements that our system was not prepared to deal with. For instance, the form had a list where users could enter one or more items, and this did not fit our key-value spreadsheet model for the data. (This form element would be like a cell in a spreadsheet containing a nested spreadsheet, which we will leave to future work.) Finally, we manually aligned all the scanned forms with the template form. Aligning the forms was necessary so that the rectangle coordinates found in Phase 1 would be applicable in Phase 3. As future work, we plan to automate the alignment process with image processing.

RESULTS AND DISCUSSION

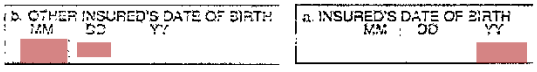
It is easiest to gauge the performance of the process by looking at the results for each phase in turn.

Phase 1: Drawing Rectangles

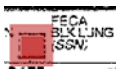
Workers agreed on rectangles for nearly every form element that we were interested in (61/64, or 95%), and we had one duplicate rectangle:



The missing form elements come from these two fields:



We did not anticipate the wide horizontal variance that workers submitted for these form elements, which threw off our fuzzy rectangle matching algorithm, which was based on a fixed tolerance of 10% of the rectangle's width. This tolerance also allowed for a duplicate rectangle (note the dark red inner rectangle):



The extra rectangle actually exhibits two problems. First, we hoped that the algorithm would consider these rectangles to be the same, since they indicate the same form element. Second, the larger rectangle revealed a bug in our system where a single worker drew two overlapping rectangles, and the system counted this is inter-worker agreement.

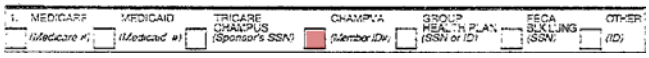
Phase 2: Labeling Rectangles

Most of the labels acquired in Phase 2 were correct, and unambiguous (41/62 or 66%), though they were not always consistent, for instance:

- Yes (Employment current employer or previous)
- employment(current or previous)-no

The next 31% (19/62) were ambiguous, meaning that the same label could apply equally well to more than one form element. For instance, "YES", since four checkboxes had this label. Note that our instructions simply asked workers to label the rectangle, so this is not a case of cheating, but rather underspecified instructions. It is reassuring, however, than many workers provided unique labels anyway.

The final 3 labels were wrong. Two were instances of a checkbox in the middle of a long sequence of checkbox-label-checkbox-label:



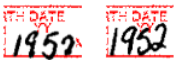
In this case, workers sometimes gave the left label when they should have given the right label. It is possible to determine which label is correct only by looking at the checkboxes on the extreme ends. Note that this is a symptom of poor form design, suggesting that our process may used to detect usability errors of this sort.

The final mistake came labeling the rectangle below:

We can infer from context that the label is a type of biological relationship, and the first letter is "s", but the scan quality is poor. The incorrect winning label was "SON", whereas one worker suggested the better label "Patient Relationship to Insured – Self".

Phase 3: Human OCR

Overall the OCR process seems to work extremely well, although we don't have the ground truth for these forms, so all we really know is that MTurk workers agree with the author's own guesses about people's handwriting. We do know that workers made at least one mistake:



We know these dates indicate the birth-year of a single individual, so they should be the same, however, workers converged on "1957" for the first date, and "1952" for the second. However, this seems understandable, since the "2" in the first date has many qualities of a "7".

Note that we included a special instruction to deal with checkboxes: "just write 'check' for check marks." Out of 261 instances of a check in a box, workers converged on the guess "check" 229 times (88%). Only four times did workers converge on "x" as the answer, where the other 28 times people agreed on a value like "M", which was the label of the checkbox.

We encountered another interesting problem on forms: people using words like "same" to refer to information on other parts of the form:

This may be difficult to overcome while preserving privacy, but maybe not. If people can see a blank form, they may be able to provide a useful guess about which other form element the "same" is probably referring to and recommend that another worker look at both fields together.

DISCUSSION AND FUTURE WORK

Our work generated insights that should be applicable in broader contexts.

Independent Agreement

Bernstein et al. [2] cite independent agreement – multiple workers generating the same response, independently – as a good sign that a response is correct. Our results corroborate this insight. For instance, in Phase 1, we only keep rectangles that multiple workers draw in the same place, and these tend to be correct. In fact, in one pilot experiment, we had multiple workers draw all the rectangles for a form, without seeing any other worker's submissions, and while no single worker got them all correct, the set of independently agreed upon rectangles was correct.

Note that this approach has limitations. The chance of obtaining independent agreement decreases as a question becomes more difficult or has too many possible correct answers. Hence, researching crowdsourcing quality of service is still an important direction for us.

Dynamic Partitioning

Drawing many rectangles on a form presents an interesting challenge: how do we partition the task so that no worker needs to do all the work? The problem is that the form is difficult to break into pieces that can be processed

independently. For instance, showing people a cropped window of the form might result in fields straddling the edge of that window. Our solution is dynamic partitioning, where we show each worker what has been done so far, so that they can decide for themselves where to make the next contribution. This approach may generalize to other problems which are difficult to break into clearly defined pieces. Note that this approach is parallel in the sense that each part of the problem is processed by a subset of the workers. However, it is not parallel in the sense of having workers work simultaneously, since we must wait for a worker to complete their work before showing their results to the next worker.

Risk of Piecework

When we divide a task into small pieces, there is a risk that workers may make mistakes which could be avoided with global knowledge about other pieces. For instance, in Phase 2, workers labeling a rectangle do not know what labels are being given to other rectangles, so they have no way to establish a consistent naming convention. (Contrast this with Phase 1, where rectangles from previous workers act as examples to subsequent workers). Also, in Phase 3, the example of the word "same" appearing in a box shows how obscuring the rest of the form prevents the worker from locating the intended value for the given field.

Human Exception Handling

When humans encounter tasks for which the instructions are poorly specified or ambiguous, they often try to make a good faith effort to solve the problem. For instance, in Phase 2, labelers would often write things like "Other Accident? -- yes", instead of just "Yes", even though the instructions did not say that the label needed to be unique. Also, in Phase 3, some check marks were OCRed as the label next to the checkbox (e.g. “M”), which is an option we had considered writing into the instruction. Of course, in this case, they were disobeying our instruction to write "check" when they see a check. However, it is interesting that these workers would have done something reasonable even without that instruction.

These findings suggest that it may be possible to write better, more succinct instructions, if we have a better understanding of how humans will naturally handle underspecified instructions. It also suggests an interesting direction for future research: can we design tasks that take advantage of human exception handling, perhaps providing an interface for people to notify the system when instructions are ambiguous?

Greg's Bio

Greg has a bachelor degree from Arizona State University and a masters from MIT. He is currently finishing his PhD thesis at MIT on algorithmic human computation. He is interested in orchestrating the efforts of many humans to solve hard design and creative problem solving tasks.

Yu-An's Bio

Yu-An received a doctoral degree in computer science from George Washington University. She is an area manager at Xerox Research Center Webster. She also holds a Master in Business Administration degree from National Central University in Taiwan.

ACKNOWLEDGMENTS

We would like to acknowledge all those who aided directly or indirectly in this work, including Naveen Sharma, Rob Miller, Bo Begole, and the Xerox Research Center Webster.

REFERENCES

1. von Ahn, L., Maurer, B., McMillen, C., Abraham, D., and Blum, M. reCAPTCHA: Human-Based Character Recognition via Web Security Measures. Science, September 12, 2008. pp 1465-1468.
2. Bernstein, M.S., Little, G., Miller, R.C., Hartmann, B., Ackerman, M.S., Karger, D.R., Crowell, D., Panovich, K. Soylent: A Word Processor with a Crowd Inside. UIST '10, ACM Press (2010).
3. Kittur, A., Chi, E.H., and Suh, B. Crowdsourcing user studies with Mechanical Turk. CHI '08, ACM Press (2008).
4. Little, G., Chilton, L., Goldman, M., and Miller, R.C. Exploring Iterative and Parallel Human Computation Processes. ACM SIGKDD Workshop on Human Computation, ACM Press (2010).
5. Little, G., Chilton, L., Goldman, M., and Miller, R.C. TurKit: Human Computation Algorithms on Mechanical Turk. UIST '10, ACM Press (2010).
6. Mason, W. and Watts, D. Financial Incentives and the “Performance of Crowds”. ACM SIGKDD Workshop on Human Computation, ACM Press (2009).
7. Quinn, A.J. and Bederson, B.B. A Taxonomy of Distributed Human Computation.
8. Snow, R., O'Connor, B., Jurafsky, D., and Ng, A.Y. Cheap and fast—but is it good?: evaluating non-expert annotations for natural language tasks. ACL '08, (2008).