Lydia B. Chilton Research Statement

I build **hybrid human-computer systems** that combine the strengths of human and machine intelligence to solve problems that neither one could solve independently.

People have innate problem-solving abilities that computers do not yet have: people synthesize **contextual clues** and apply rich world knowledge. Yet people struggle when problems become complex: there are too many contextual clues to fit into human memory, and the possibilities of trial and error are overwhelming.

Fortunately, computers excel at **search** and have memory that far exceeds human limitations. With crowdsourcing as a source of human conceptual understanding, we can harness the **crowd's** ability to explore conceptual solution spaces and use algorithms to guide the search to completion. However, the conceptual understanding people have is open-ended, and thus rigid or static workflows are not amenable to leveraging our rich understanding of the world. We need dynamic crowd algorithms to solve problems that are **hard, big, ill-defined and creative**.

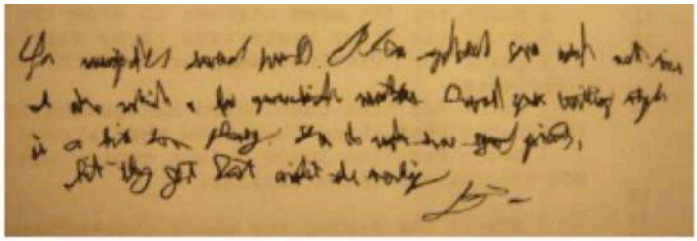
Dynamic Crowd Algorithms

Dynamic crowd algorithms solve complex problems by using people to explore conceptual solution spaces with **microtasks**, and **algorithms** to guide the exploration to completion. I have built and deployed four systems that decompose and solve complex problems using dynamic crowd algorithms. My evaluation shows increased group efficacy and individual creativity.

Problems **too hard** for one person to solve

When one person cannot solve a problem, multiple perspectives can bring new insights. Microtask crowdsourcing began with labeling images [1], and other individual microtasks. My colleagues and I realized that human computation could be used more generally in a computational framework such as algorithms that coordinate human effort. We built the TurKit toolkit for writing crowd algorithms on Mechanical Turk [10, 11].

For example, in Figure 1, Mechanical Turk workers could not individually decipher messy handwriting. However, by building on the partial solutions of others, the handwriting was almost completely deciphered. The Improve-and-Vote" algorithm iteratively solicits potential improvements to the transcription, and solicits workers to vote on whether to adopt the improvement or solicit a different improvement. TurKit uses microtasks to get suggested improvements and to vote on which branch is better. Iterative application of Improve-and-Vote is the algorithm that guides these microtasks to a solution.



Iteration 1:

You (?) (?) (?) (work). (?) (?) (?) work (not) time. I (?) (?) a few grammatical mistakes. Overall your writing style is a bit too (phoney). You do (?) have good (points), but they got lost amidst the (writing). (signature)

Iteration 6:

You (misspelled) (several) (words). Please spellcheck your work next time. I also notice a few grammatical mistakes. Overall your writing style is a bit too phoney. You do make some good (points), but they got lost amidst the (writing). (signature)

Figure 1. Results of TurKit's Improve-and-Vote algorithm to decipher bad handwriting. Iterations 1 and 6 are shown. Red text indicates missing or incorrect words.

Other AI researchers independently optimized our Improve-and-Vote algorithm for quality and cost using POMDPs [9]. Other HCI researchers (Michael Bernstein at Stanford [3], and Jeffrey Bigham at CMU [4]) implemented their systems using TurKit and were directly inspired by our introduction of crowd algorithms.

TurKit was an early break-through in crowdsourcing. I saw the potential to generalize the technique of exploring alternatives at a crowd-scale and intelligently combine partial answers to solve big, harder, more open-ended problems in an even more dynamic way.

Problems **too big** for one person to solve

Problems that require a cohesive solution synthesized from many pieces of information are often overwhelming for a single person to solve. Data organization tasks often need people's contextual knowledge, but overwhelm people as the size of the task grows large. I developed the Cascade algorithm running on TurKit to crowdsource the creation of taxonomies that allow people to get the big picture from large datasets.

For example, in Figure 2, Cascade [6] was used to create a taxonomy out of 100 photos. Creating a taxonomy is too big and overwhelming to be done by one person iteratively, so Cascade uses crowds of people working in parallel. Cascade was also used to quickly organize colors, travel tips, writing advice, and hackathon ideas.

Cascade issues microtasks that help people search a conceptual space of possible labels by generating labels for data items individually and then testing the labels globally. The high-level process is a machine algorithm that infers relationships from the network structure connecting the items. Cascade recursively generates and tests more labels until all the items are taxonomized.

Many researchers have followed in my footsteps, re-implementing Cascade, optimizing it with machine learning [5], and being inspired to tackle similar global-from-local problems [2, 12]. I received a Facebook Fellowship for my Cascade work based on its ability to help people collaboratively structure their photos into cohesive narratives.

Figure 2. A taxonomy crowdsourced by Cascade. The left represents the input: 100 nature photos. The right side is the resulting taxonomy with call outs showing examples of three categories: tiger, worker, and historical landmarks.

Problems too **ill-defined** to automate

Algorithmic techniques work well when problems can be fully specified. However, many real-world problems have important cultural and social constraints that cannot be written down, and may not be evident until a problem surfaces them. For example, in software development, it is often not until a prototype has been developed that the client realizes it needs additional features.

I developed Frenzy [7], a real-time collaboration platform for crowds of experts to organize data into clusters. This work builds on Cascade but recognizes that for many problems as new global constraints emerge, the solution will have to adjust. I deployed a version of Frenzy to the two largest HCI conferences, CSCW and CHI, to organize hundreds of accepted papers into sessions. Each session had to have four similar papers and a unique name describing the theme. However, as the sessions started to emerge, more constraints appeared. For example, there were four papers that fit together perfectly in a session called "Collaborative Crowdsourcing." Unfortunately, all four papers had the same author, and it was deemed culturally inappropriate to have one session to showcase the work of one author. In light of this newly discovered constraint, the clusters had to be changed.

Frenzy builds on the microtasks that I first used in Cascade, but embeds them in a real-time updating platform that enables all contributors to see other's contributions and the current global structure in real time. Additionally, Frenzy uses a new algorithm to guide the microtasks to a solution. Frenzy presents a sequence of goals to all contributors that drives the crowd toward ideation in a flare-and-focus pattern: come up with a lot of ideas for categories, select the best category for each paper, based on the categories (which are often as large as 40 items) put every paper in a smaller sessions with appropriate names, lastly make sure every session has exactly four papers. The flare-and-focus algorithm leaves the experts in charge of the overall outcome and can adjust to dynamically emerging constraints, but still allows as many as 60 people to collaborate together in real time. In our deployment at CSCW 2013, the process that normally takes 6-8 hours (with breaks), was done in 88 minutes using Frenzy.

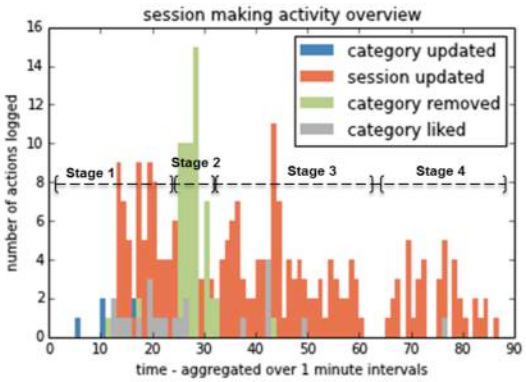


Figure 3 Activity during Frenzy session-making. Multiple interleaved task types over time show the dynamic nature of the Frenzy algorithm.

Frenzy was deployed in production to the two largest conferences in HCI to create the conference sessions: CSCW 2013 (136 papers) and CHI 2014 (431 papers). The Frenzy paper won Honorable Mention for Best Paper at CHI 2014.

Problems that require **creativity**

Some of the hardest problems are those that cannot pattern match to a similar problem we have previously solved. Developing a novel solution requires creativity. Often we think of creativity as magical and something that cannot be broken down into concrete steps. This is a myth that I help dispel with HumorTools [8] – a system that helps people write humorous news satire by breaking the process into microsteps. Given a real news headline such as *"People Bending iPhones at Apple Stores"*, the system helps people write jokes relevant to the headline such as *"I can't believe people would just walk into an Apple Stores and start breaking things like it's a Best Buy."*

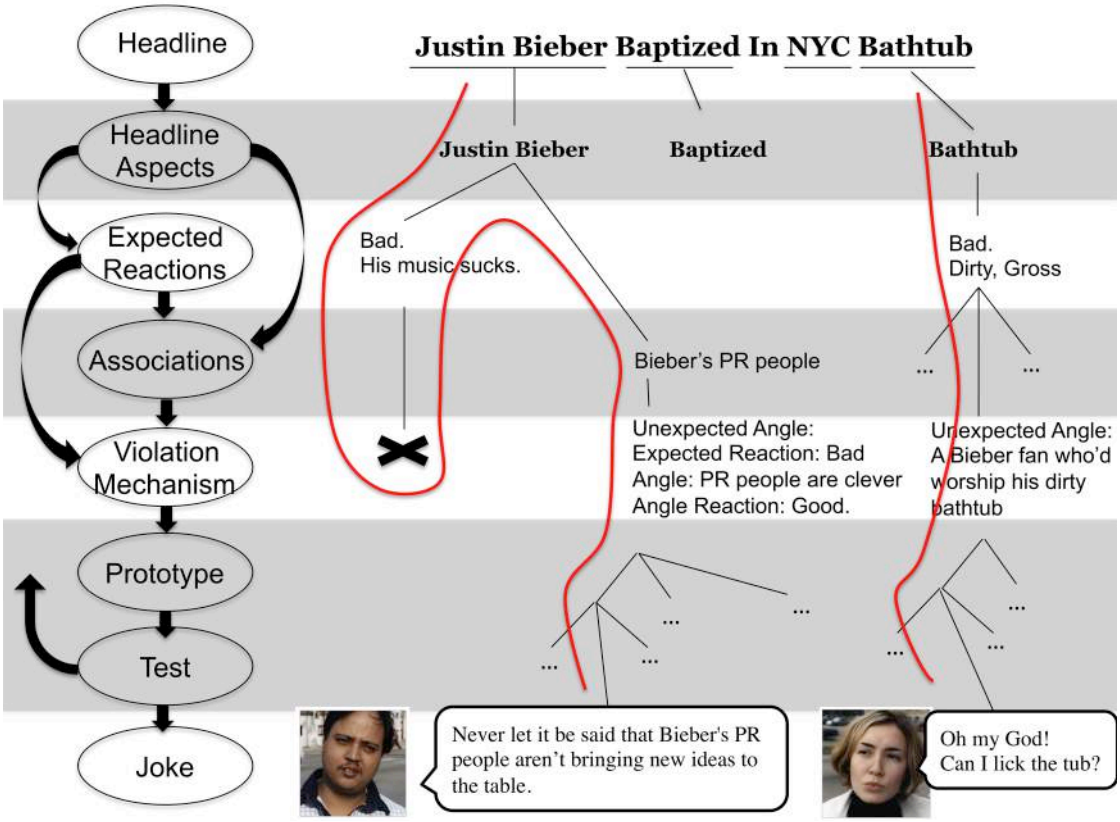


Figure 4 HumorTools uses microtasks in the iterative design process to help people write news satire. On the left are the stages of the iterative design process with arrows showing typical transitions between states. On the right is an illustration of the process of generating two jokes – an exploration of conceptual spaces in a search-tree like structure. The red lines indicate the flow of activity, with the red line on the left indicating backtracking to find a humorous solution that satisfied the humor constraints.

HumorTools decomposes humor writing into microtasks that help novices explore the conceptual space of humor, such as identifying possible points of view, insult targets, and falsifiable assumptions. HumorTools teaches people to apply the microtasks in a dynamic algorithm that is derived from both the central principle of HCI – iterative design – and the AI technique of dynamic constraint satisfaction (See Figure 4). It combines conceptual search with dynamic constraint discovery, and uses backtracking when it gets stuck in a constraint that is difficult to meet.

I tested the HumorTools website with 20 participants. My study showed that users could search a wider range of ideas, employ more solution patterns, and write better jokes. HumorTools externalizes the creative process and it helps dispel the myth that writing humor requires magic. HumorTools received a year-long grant from the Brown Institute for Media Innovation to extend the website to new creative domains and deploy a production humor-generation system.

Future Work: Automating Iterative Design

My work decomposes large, hard, ill-defined, and creative problems by using microtasks for people to explore a conceptual search space, and algorithms to guide them to a solution. My goal is to build **hybrid human-computer systems** that do **open-ended problem solving**. Iterative Design in HCI is a powerful open-ended problem solving method. I am the first person to demonstrate that Iterative Design can be scaffolded with crowdsourced microtasks to carry out

creative problem solving. I will **generalize** the process across domains, **optimize** the process with people, and **automate** the process with machine intelligence. My five year research plan is as follows:

Apply Iterative Design Crowdsourcing to Other Domains

Humor is creative and impactful. Persuasive argumentation is also creative and impactful. I will discover the search spaces of persuasive argumentation and extend the iterative design approach to this new conceptual search space. This method might be used to author opinion editorials (op-eds) for local and national newspapers on issues ranging from education to medicine to technology. In addition to producing persuasive text, I will generalize my approach to producing **persuasive visuals** to accompany the articles and grab attention.

Optimizing Transitions Between Iterative Design Stages

In complex processes, the dynamic transitions between stages of the process (e.g., from prototyping to testing in iterative design) are often a source of difficulty for people. For example, should you wait to come up with a better idea or should you prototype the idea you have now? To optimize the process we can use optimization approaches, such as POMDPs and reinforcement learning, to model the transitions as a function of the current context and the user's ability. POMDPs have been successfully used to optimize TurKit's Improve-and-Vote algorithm and Cascade's taxonomy creation algorithm. It is currently the best candidate to optimize humor or other domains that require complex, open-ended problem solving.

Automating Conceptual Exploration

Machine intelligence is quickly getting better at conceptual understanding, such as for image descriptions, and it may soon be better at conceptual search than people are. If this happens, we can embed this ability in a hybrid human-computer algorithm for dynamic content creation. The microtasks I identified in HumorTools represent a clear starting point to replace some or all of the conceptual search with machine intelligence. Eventually, we may need machine intelligence to solve problems that are too large and complex for us to solve in areas such as physics, biology, writing software and medicine. Advances in these fields will need humans in the loop for the foreseeable future, thus hybrid human-computer systems can guide us forward.

Conclusion

Richard Hamming challenges us to work on the most important problems in your field. To me, the most important problem in Human-Computer Interaction (HCI) is "enabling people and computers to do iterative design better." Iterative design is fundamental to engineering usable, secure, efficient, and novel software. It solves open-ended problems that other methods cannot solve because it jointly discovers the *real* problem and solves it. We know iterative design is powerful, but we do not yet fully understand how it works. In the future, I am going to generalize, optimize, and automate iterative design to make hybrid human and machine problem solving approaches systematically more intelligent.

References

1. Luis von Ahn and Laura Dabbish. Labeling Images with a Computer Game. CHI 2004.

2. Paul Andre, Aniket Kittur, Steven P. Dow. Crowd Synthesis: Extracting Categories and Clusters from Complex Data. CSCW 2014.

3. Michael S. Bernstein, Greg Little, Robert C. Miller, Björn Hartmann, Mark S. Ackerman, David R. Karger, David Crowell, and Katrina Panovich. Soylent: a word processor with a crowd inside. UIST 2010. Best Student Paper Award.

4. Jeffrey P. Bigham, Chandrika Jayant, Hanjie Ji, Greg Little, Andrew Miller, Robert C. Miller, Robin Miller, Aubrey Tatarowicz, Brandyn White, Samual White, and Tom Yeh. VizWiz: nearly real-time answers to visual questions. UIST '10. Best Paper Award.

5. Jonathan Bragg, Mausam, Daniel S. Weld. Crowdsourcing Multi-Label Classification for Taxonomy Creation. HCOMP 2013. Best Paper Award.

6. Lydia B. Chilton, Greg Little, Darren Edge, Daniel S. Weld, and James A. Landay. 2013. Cascade: crowdsourcing taxonomy creation. CHI 2013.

7. Lydia B. Chilton, Juho Kim, Paul Andre, Felicia Cordeiro, James A. Landay , Daniel S. Weld, Steven P. Dow, Robert C. Miller, Haoqi Zhang. Frenzy: Collaborative Data Organization for Creating Conference Sessions. CHI 2014. **Honorable Mention for Best Paper**

8. Lydia B. Chilton, Daniel S. Weld, James A. Landay. HumorTools: A Microtask Workflow for Generating News Satire. In submission to WWW 2016.

9. Dai, P.; Lin, C. H.; Mausam; and Weld, D. S. 2013. POMDP-based control of workflows for crowdsourcing. Artificial Intelligence 202(0):52–85.

10 Exploring Iterative and Parallel Human Computation Processes. Greg Little, **Lydia B. Chilton**, Max Goldman, Robert C. Miller. KDD-HCOMP 2010.

11 Greg Little, **Lydia B. Chilton**, Max Goldman, Robert C. Miller. TurKit: Human Computation Algorithms on Mechanical Turk. UIST 2010.

12. Vasilis Verroios and Michael S. Bernstein. Context Trees: Crowdsourcing Global Understanding from Local Views. HCOMP 2014.