Chorus: Letting the Crowd Speak with One Voice

University of Rochester Technical Report #983

Walter S. Lasecki[1], **Anand Kulkarni**[2], **Rachel Wesley**[1], **Jeffrey Nichols**[3]
Chang Hu[4], **James F. Allen**[1], **and Jeffrey P. Bigham**[1]

University of Rochester[1]
Department of Computer Science, ROC HCI Lab
{wlasecki,james,jbigham}@cs.rochester.edu
{rwesley2}@u.rochester.edu

University of California, Berkeley[2]
Department of IEOR
anandk@berkeley.edu

IBM Research - Almaden[3]
USER Group
jwnichols@us.ibm.com

University of Maryland[4]
Department of Computer Science
changhu@cs.umd.edu

ABSTRACT

Autonomous systems cannot yet reliably engage in an open-ended dialogue with users due the complexity of natural language processing, but online crowds present new opportunities to do so. We introduce Chorus, a system enabling real-time two-way natural language conversation between an end user and a single virtual agent powered by a distributed crowd of online humans. Chorus maintains consistent, on-topic conversations with end users across multiple sessions even as individual members of the crowd come and go by storing a shared, curated dialogue history. While users see only a steady stream of dialogue with a single conversational partner, multiple crowd workers collaborate to select responses via an interface that allows them to rapidly scan conversational history, identify relevant parts of a conversation, and select between responses. Experiments show that dialogue with Chorus demonstrates conversational memory and interaction consistency, answering over 84% of all user queries correctly. More generally, Chorus demonstrates the ability of crowd-powered communication interfaces to serve as a robust alternative to virtual agents when interacting with software systems.

Author Keywords

crowdware, natural language processing, virtual agents, human computation, crowd computing

ACM Classification Keywords

H.4.2 Information Interfaces & Presentation: User Interfaces

INTRODUCTION

Using natural language dialogue to interact with automated software has been a goal of both artificial intelligence and human-computer interaction since the early days of computing. However, the complexity of human language has made robustly handling two-way conversation with software agents a consistent challenge [1]. Existing dialogue-based software systems generally rely on a fixed input vocabulary or restricted phrasings, have a limited memory of past interactions, and use a fixed output vocabulary. Real-world conversations between human partners can contain context-dependent terms or phrasing, rely on conversational memory, require commonsense knowledge about the world, events, or facts, retain memory stretching back over a long history of interactions and shared experiences, and infer meaning from incomplete and partial statements. Even the most advanced virtual agents have difficulty handling all of these scenarios.

While individual humans have no difficulty in maintaining natural-language conversation, it is often infeasible, unscalable, or expensive to hire individual humans or expert agents to act as conversational partners for long periods of time or large numbers of conversational partners. In recent years, *crowd computing* has become a popular method to scalably solve problems that are beyond the capabilities of autonomous software by subcontracting them to groups of paid humans over the web. In this model, *the crowd* refers to a transient pool of online, semi-anonymous workers recruited for short periods of time from online microtask marketplaces such as Amazon's Mechanical Turk. Crowd computing models can provide software with many of the abilities of individual humans while maintaining much of the scalability of autonomous software, but also presenting a new range of challenges in reliability, incentivization, and accuracy.

Many recent crowd computing applications have used the crowd to interpret natural language instructions provided by the user, in applications such as image description [6], speech recognition [12], interface control [13], document editing [4], and even vacation planning [22, 11]. Such systems require only a single round of communication however, from the requester to the worker and back. The reason is that maintaining consistent communication with the crowd is inherently difficult because the pool of online agents is always

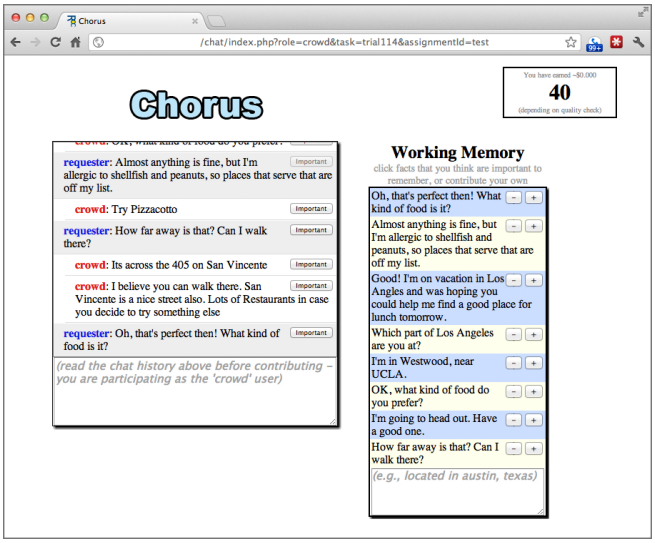

Figure 1. The chat interface for workers encourages them to both propose responses and agree with those of others. It also enables members of the crowd to curate the crowd's shared chat history (working memory). The "requester" view is similar but removes the "Working Memory" section and only displays suggested crowd messages once they have sufficient votes.

changing and no individual worker can be relied upon to be available at a given time to respond to a query or to continue a dialogue for more than a few moments. Individual workers may fail to respond to queries intelligibly for various reasons including misunderstanding of task directives, laziness, or outright maliciousness. Furthermore, individual workers may experience delays that are beyond their control, such as network bandwidth variability, that make conversation inefficient. As a result, the crowd has historically been used only as a tool to interpret human instructions, rather than the foundation of a dialogue-based system.

In this paper we present Chorus, an online collaborative interface that allows a crowd to workers to engage in a two-way natural language conversation with users as if they were a single, reliable individual (Figure 1). Using Chorus, workers are able to produce, agree upon and submit responses to user statements and queries quickly and easily using a combination of human and machine filtering. Chorus uses three components to simulate realistic conversations. First, a *collaborative reasoning* system lets workers select reasonable responses from a range of crowd-produced suggestions, filtering out responses from individual workers that do not fit the flow of the conversation and presenting users with a consistent conversational flow. Second, a *curated memory system* lets workers highlight salient parts of a conversation as they emerge and presents it in a highly visible region of the interface. This provides a rapidly-accessible short-term memory of facts and statements that are sorted by importance, and supplements a long-term memory of interactions that remains over time. Finally, a *dynamic scoring system* rewards workers for collaborative interactions that support the goal of consistent conversation, such as making useful statements or highlighting facts that are used later. These features enable the crowd to learn and remember information collectively while filtering out low quality workers.

In trials with the crowd, we found that Chorus was able to maintain qualitatively consistent and natural conversations between a single user and large numbers of crowd participants, remaining on-focus with single topics much as an individual user would even as individual themes and points of discussion changed. Moreover, we found that Chorus was capable of retaining meaningful long-term conversational memory across multiple sessions, even as individual users changed. Lastly, we showed that Chorus could be used to enable existing interfaces with dialogue capabilities.

RELATED WORK

Chorus builds on prior work in both real-time and offline human computation. Human computation [18] has been shown to be useful in many areas, including writing and editing [4], image description and interpretation [6, 19], and protein folding [8]. Chorus aims to enable a conversation with a crowd of workers in order to leverage human computation in a variety of new ways. Existing abstractions obtain quality work by introducing redundancy and layering into tasks so that multiple workers contribute and verify results at each stage [16, 10]. For instance, the ESP Game uses answer agreement [19] and Soylent uses the multiple-step find-fix-verify pattern [4]. Since these approaches take time, they are not always suitable for interactive real-time applications.

Crowdsourcing Web Search and Question Answering

Prior work has looked at providing specific answers to a wide range of uncommon questions searched for on the web by having workers extract answers from automatically generated candidate webpages [5].

Most Internet forums rely on the idea of using groups of workers to answer to queries. Contributors are expected to read through the thread history and gain context before submitting responses. Those submitting the original question can also respond to the answers provided, and give feedback concerning issues the group has with the query. This is similar to what Chorus aims to elicit from web workers. The difference is that forums and similar system typically generate answers offline, often taking hours or even days to arrive at a final answer. In order for Chorus to enable conversational interfaces, it needs to be able to provide real-time interactive responses. Other systems such as ChaCha [1] try to get answers back to users in nearly-realtime, but provide answers from individual workers, without considering the history of the user. Another difference from typical forums is that participants in Chorus collectively participate in dialogue as though they were a single individual, instead of responding independently.

Real-Time Human Computation

Researchers have only recently begun to investigate real-time human computation. VizWiz [6], was one of the first systems to elicit nearly-realtime response from the crowd. It introduced a queuing model to help ensure that workers were available both quickly and on-demand. For Chorus to be available on-demand requires multiple users to be available

[1] www.chacha.com/

at the same time in order to collectively contribute. Prior systems have shown that multiple workers can be recruited for collaboration by having workers wait until a sufficient number of workers have arrived [19, 7]. Adrenaline combines the concepts of queuing and waiting to recruit crowds (groups) in less than 2 seconds from existing sources of crowd workers [2]. Further work has used queuing theory to show that this latency can be reduced to under a second and has also established reliability bounds on using the crowd in this manner [3]. Work on real-time captioning by non-experts[12] uses the input of multiple workers, but differs because it engages workers for longer continuous tasks. These systems introduce a variety of methods for rapidly recruiting crowds for a task that we use in Chorus, but focus only on one-way interaction with the crowd rather than extended engagement.

Legion enables real-time control of existing user interfaces by allowing the crowd to collectively act as a single operator [13]. Each crowd worker submits input independently of other workers, then the system uses an *input mediator* to combine the input into a single control stream. Our work allows systems such as Legion to be more easily and naturally controlled by users by adding a conversational layer on top. Importantly, Chorus does not need to change the underlying system itself, making development of crowd systems with natural language interfaces more modular.

Organizational Learning

The idea of crowd memory and learning in continuous real-time and collective answer tasks is related to the organizational learning theory (summarized in [15]). Organizational learning has previously been demonstrated in continuous real-time crowdsourcing using Legion [14]. There learning was shown in implicit cases where new workers learned from experienced workers by observing the collective actions of the group. This means that activities in which information is ideally only provided once, such as conversation, cannot be accounted for using their model. Here, we instead aim to make the crowd's memory more explicit by having workers curate a knowledge base to be used by future workers in much the same way historians do on a societal scale by aggregating documents and other accounts of past events.

Measuring Human-Crowd Conversation

Several approaches have been proposed for measuring the effectiveness of conversational agents. One approach is to use reference answers, then compare agent-provided responses to these [9]. This approach falls short when rating dialogues that may not follow the fixed expected path, even when the response itself is valid. PARADISE [20] attempts to create a structured framework for evaluating spoken dialogue that is separate of the specific task being performed. However, because PARADISE tries to minimize completion time, it is generally biased in favor of shorter conversations, not just ones that accomplish a task more effectively. Another approach is to elicit subjective feedback from users of the system itself to get a more comprehensive notion of whether or not the system is helpful. Webb et al [21] explore measuring conversations for appropriateness, rather than via a fixed 'performance' metric.

In this paper, we evaluate the crowd's ability to hold a conversation using a combination of measuring appropriateness and obtaining user evaluations of the interaction. Since our metric must account for variations in Chorus that are not seen in automated systems, such as varying time to find the same answers, this gives us a more complete metric of the conversation without being biased towards speed alone. It also allows conversations to take one of many appropriate paths, which is important given the multitude of workers and opinions that Chorus will incorporate.

CHORUS

To demonstrate the utility of crowd-powered conversational interfaces, we consider the following scenario. Susan is a mother of 3 who uses Chorus as a personal assistant and semi-autonomous search tool while driving home from work and picking up her kids. When Susan is driving, she is situationally disabled, as she is unable to use her hands or divert her focus from the road. We will see how Chorus can listen to Susan's requests and interact with an interface on her behalf, much like a personal assistant.

When Susan first opens Chorus at the start of a drive, the system begins recruiting crowd workers from the web. Within a few seconds, the system is ready to begin responding. Since Susan is driving, she opts to use the voice input mode. She is on her way from a meeting to her office in downtown Chicago, IL to pick up her belongings, then needs to quickly head over to pick up her kids from soccer practice. Since Susan does not usually head out from the office at this hour, she asks Chorus to find the best route to take from downtown to Lemont, IL at 3:30 P.M.

Individual workers in the crowd can listen to Susan's request, interpret the instruction and then develop an individual response. However, getting a single, unified response back from the crowd as a whole requires achieving agreement on what needs to be done and said. To support this, Chorus presents each worker with the responses from other workers, letting them see what has been proposed, then asks them to select the responses that they agree with. Using workers for both response generation and selection improves accuracy since people are generally able to to identify correct answers better than they are able to generate them [17].

When a proposed answer has sufficient crowd agreement it is "locked in", presenting it to the user via speech or text (Susan). Chorus requires a majority of workers to agree with a message in order to lock it in, but this can be tuned to better suit different tasks' needs. The end result is that users are not overwhelmed by a flood of repetitive and possibly competing feedback. Chorus can use both crowd voting and automatic agreement found by measuring the similarity between multiple submitted answers. Automatically finding agreement allows the system to forward answer with high agreement to the user without needing to wait for the crowd to vote. The Chorus worker interface is shown in Figure 1.

The user sees only a chat window with the "crowd" respondent. We also present both users and workers with a "typing"

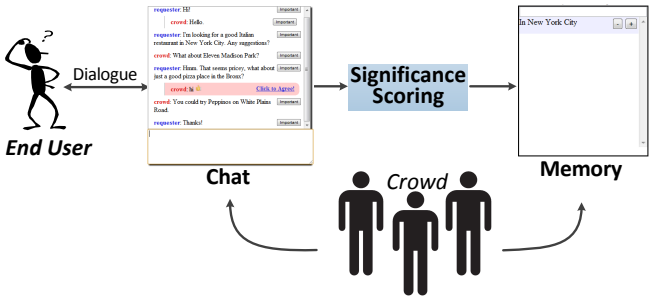

Figure 2. The Chorus system. Users first make a request, which is forwarded to the crowd. Workers can both submit responses and vote for those proposed by others. Responses are hidden from users until sufficient agreement between workers is achieved. Workers can also add information to the crowd's working memory by either voting on existing lines from the conversation, or adding summaries of key facts.

status indicator that tell the other party that a response is being composed. To determine when the crowd response is being composed, Chorus checks if there were currently any active proposals that have not yet been forwarded to the user. Figure 2 shows a diagram of the Chorus system.

Once Susan has asked for directions, one worker quickly returns "take I-55", but others have looked up traffic guides online and know that I-55 is often backed up with heavy traffic at that time of day. Workers propose a variety of alternatives at first, then one proposes asking if Susan is open to taking side roads. Other workers agree, and the questions is forwarded to Susan, who replies that she would prefer to stay on main thoroughfares. Using this information, some workers propose taking Archer St. instead. Finally, while verifying this route, one worker sees a traffic report saying there is a major accident on Archer St. that will not be cleared for at least an hour, so they propose "take I-90 to W. Chicago Ave to avoid traffic on I-55 and an accident on Archer St." Other workers quickly check this option and see it is the best one. Within moments they switch their vote and the I-90 option is forwarded to Susan and spoken out loud by the application. Requiring agreement from the crowd prevents several competing answers from being forwarded to Susan, which might have otherwise caused a distraction.

Finding Consensus

Asking workers to agree on answers before forwarding them to the user reduces the chance of erroneous answers, but it is difficult to automatically identify agreement. To avoid this problem, we let workers directly vote on the answer they agree with, or propose a new one if needed. However, using a voting system to find consensus requires reward schemes designed to elicit accurate answers quickly rather than rewarding sheer bulk of answers.

To encourage workers to submit only accurate responses, Chorus uses a multi-tiered reward scheme that pays workers a small amount for each interaction with the interface, a medium reward for agreeing with an answer that subsequently gets enough votes from others to forward to the user, and a large reward for proposing an answer that the crowd eventually chooses. The difference between these reward values for each of these cases can be used to adjust workers' willingness to select one option over another. Our experiments used 20, 1000, and 3000 points for each value respectively, where 1500 points correlates to 1 cent. These numbers can be tuned by developers to best suit their application.

To prevent these rewards from being abused by workers, we also adjust the points over sequential inputs by reducing the value of each subsequent contribution by two thirds. Reward values are reset with each user input. This means that the points given to workers for each contribution to the same response exponentially decreases, removing the incentive for workers to provide excess input. Since Chorus's goal is a consistent dialogue, reducing the number of responses a worker can submit to answer a single user input does not limit the system's functionality.

Conversational Memory

Memory is an important aspect to any conversation because it allows participants to have a shared context, and history of events and topics covered. This reduces the total amount of information that must be conveyed by largely eliminating the need for information to be repeated, and allows more information to be transferred with each utterance as the conversation progresses.

Context is typically gained over the course of one or more conversations. However, since the crowd is dynamic, we must account for workers who join conversations already in-progress and may not have been around for previous conversations. The chat history logs contain this contextual information, but having each worker read through the entire conversational history with a particular user eventually becomes prohibitively time consuming and difficult. Instead, we need a means of directing users to the most important aspects of a conversation without making them parse large amounts of data.

In our example, Susan has picked up her kids from soccer and now wants to find a place to pickup food nearby. The crowd is able to see that the recent query for directions took her to Lemont. The crowd quickly confirms this is her current location and begins to search for options. Furthermore, because Susan has used Chorus to find a place to eat before, the crowd is able to see that one of her sons is allergic to seafood, and Susan herself is on a diet. The crowd responds with the name of a take-and-bake pizza place near by that also serves popular salads, has been a favorite of Susan's in the past, and has not been recommended recently. Happy with the decision, Susan confirms the address of the restaurant and proceeds in that direction. By using workers' understanding of the information stored in the shared memory, the crowd was able to return useful information, even though the query itself underspecified the problem by not mentioning a location, the son's food allergies, or the family's preferences. This semi-autonomous behavior goes beyond the underlying system being controlled (the search engine) and shows how Chorus can act as an intelligent agent that is able to mediate interactions with the system's interface.

In order to perform multi-session memory curation in the crowd, we ask workers to highlight portions of the conversation that contain important information in order to increase the saliency of these facts for future workers reading through the chat history by highlighting them and making them visible in a separate 'memory' window. Workers can also contribute summarized facts to this window by entering them in separately. Workers can increase or decrease the importance of each line in the crowd's memory by voting. The lines are then sorted in descending order importance.

To prevent new additions from being excluded from the memory window based on total votes, a separate portion of the window is used to show the newest entries proposed by workers. To promote temporal relevance, the ranking of a given fact will also decay automatically over time if no workers have voted for it recently. Contributions to the collective memory are also rewarded using the multi-tiered point system described above. Allowing this curation process to take place during the course of the conversation reduces the per-worker overhead involved in maintaining a shared history.

Paying Workers for Continuous Tasks

In order to pay workers who may remain active in the task for different lengths of time fairly, we base payments on the number and accuracy of their contributions. This means that workers are paid based on their contributions to the conversation over any span of time, preventing them from being encouraged to leave sooner by paying a fixed amount. Further more, we expect workers will be encouraged to remain connected longer than they otherwise would because of the promise of continued pay and because they are compensated for their increased knowledge of the task as it continues.

During our example, Susan only paid a small amount for workers to complete her task because workers were paid based on their useful input, and she was able to pay the crowd only for a small unit of time (a few minutes). This is considerably cheaper than hiring an individual, dedicated employee to serve as Susan's assistant.

EXPERIMENTS

To test the feasibility of using the crowd as a conversational partner, we performed experiments focusing on two different aspects: conversational consistency (measured in terms of topic maintenance) and memory of past events (measured in terms of factual recall). In all experiments, the crowd was recruited from Amazon Mechanical Turk users who were US-based with accuracy of 90% or higher. These workers were paid 5 cents plus an additional half-cent for each message successfully accepted by Chorus. Our tests used a total of 33 unique workers with a mean of 6.7, and median of 7 workers per trial.

In our experiments we focused on the crowd's ability to generate reasonable responses in a controlled environment. To ensure that the results were not skewed by the varying quality of users, task descriptions, or communication styles, we generated a script for each of our tasks and asked a student volunteer to follow the script as closely as they could, while still allowing the conversation to flow naturally as it would if they were talking to another person (all while following the persona laid out in the script).

Conversational Consistency and Dialogue

In this test, we sought to measure whether the crowd can hold a *consistent* conversation. We define consistency as the ability to hold a conversation without inappropriately changing topics, repeating information unnecessarily, or providing answers not desired by the other party.

To test consistency, we created user scripts that gave a high-level description of two topics that could be covered in the conversation: i) where to go to lunch, including if it would be worth going to a particular destination, and what the best mode of transportation was recommended to get there, and ii) what the average weight of a given breed of dog was, then determining if that size dog was appropriate for life in an apartment. One of these scripts was used in each test, and conversations were allowed to be free-form as long as the topic was addressed. For these tests, workers were not provided any chat history on which to base their answers.

Trials were spaced out over a period of two days and the conditions (memory or consistency) randomized. Parameters such as the location and dog breed were changed between tasks so that any returning workers would not already know what to expect from the conversations.

To avoid bias when holding the conversations, we recruited an undergraduate student who was unfamiliar with the project to hold conversations based on our scripts. They were asked to hold a conversation until responses to all three questions were provided or 10 minutes passed without a correct answer. Once done, two of the authors coded the responses according to the following scheme:

- **On-Topic:** The response furthers the topic of conversation that the user proposed by providing an answer or opinion related to the question that the user asked. If the relevance of a response is unclear, the user's response (correction or affirmation) is used to determine its status.
- **Off-Topic:** The response is either not clearly related to the user's current topic (including unsolicited information about the worker them self) or provides an instantly identifiable incorrect answer to the user's question (for example, Q:"What is the capitol of Texas?" A:"Outer space.")
- **Clarification:** The response asks a clarifying question to either confirm a fact or request the user provide more details about their question. The question must be clearly based on the user's topic.
- **Generic:** The response does not obviously contribute to the conversation, but is not invalid or out of place (does not break the topic flow). Generic input also does not elicit or receive a response from the user.

We discarded generic statements that were not clearly relevant to the conversation at hand (i.e. "I see.") since, even if appropriate, these responses could have been provided at any

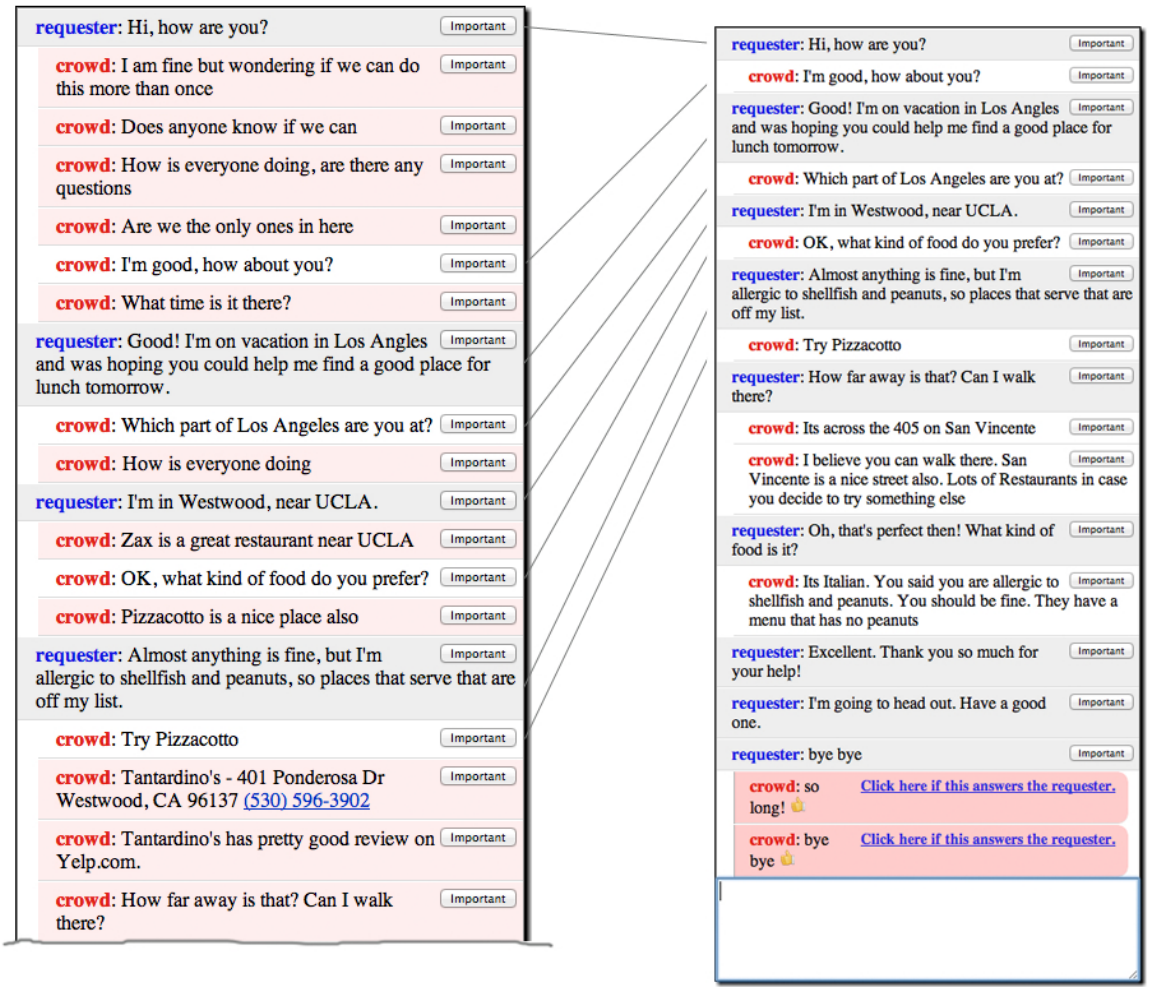

Figure 3. A clipped view of the raw chat log (left), and the filtered view seen by the requester (right). Messages in pink did not receive sufficient votes from the crowd to make it into the requester view. Some of these messages did not contribute to the conversation (e.g. new workers asking questions about the HIT in the beginning), while others were simply not chosen.

point by users without first understanding the context of the current speech. Including these utterances in the final numbers would only have improved results, since none of them occurred at inappropriate times in the conversation, such as when the user was expecting a factual response. Additionally, we tracked the number of user-generated questions that were answered correctly and incorrectly. We considered an answer correct when the information provided eventually lead to a response that required no followup or additional action to by the user.

Memory

To test the system's ability to remember information over time, we ran an experiment that simulates new members of the crowd arriving to Chorus to find pre-populated short- and long-term memory logs. We wanted to find whether the crowd would be willing and able to use the highlighted history logs to bring themselves up to speed on the current conversation. We generated an artificial chat history and presented new workers with information pre-highlighted by the authors in the memory window. This simulates a prior chat interaction carried out by past participants in Chorus. In our test, the artificial chat history simulates a conversation between a user and Chorus concerning a trip the user is taking, and contains information about the user's dietary restrictions and the city she is visiting. We asked the crowd to 'continue' the conversation with a new live user, and asked them questions that assume background available from the memory, including suggestions for new restaurants in the area. The user's script assumed that the crowd had full knowledge of prior interactions, and did not contain reminders about facts previously mentioned. The user was free to specify these facts only if explicitly prompted by the crowd.

Measurements

From these dialogues, we calculated the total percentage of all dialogue that was forwarded to users (visible dialogue), the proportion of the visible dialogue that was on-topic, the error rate in the visible dialogue, the percentage of user queries that were successfully answered, the number of times workers correctly leveraged the memory interface when referencing facts that had been covered before they arrived, and the number of times workers failed to do so. Each of these events were identified by two of the authors independently reviewing and marking results, then marking an event when there was agreement.

	Total Lines	Accurate Responses	Errors Made	Clarifications Asked	Questions Asked	Answers Provided	Memory Successes	Memory Failures
Consistency #1	24	9	0	0	4	4	-	-
Consistency #1	55	50	1	0	7	6	-	-
Consistency #1	33	11	0	0	2	2	-	-
Memory #1	138	53	30	3	5	3	4	2
Memory #2	63	15	1	1	4	2	1	0
Memory #3	30	29	1	1	3	3	1	0
Memory #4	28	7	0	2	3	2	2	0

Figure 4. Results for the conversations with Chorus..

RESULTS

In our analyses, we reviewed several hundred lines of conversation and interaction with Chorus following our two lines of inquiry. Overall, we found that Chorus filtered out all but 46.9% of responses given by the crowd and answered 84.62% of inquiries correctly and with useful information. The details of each set of trials are discussed below.

Consistency

We ran three consistency experiments. In each, we used 3 conversations spanning a total of 112 lines suggested by the crowd. This was filtered down to 70 lines by the curation system, resulting in only a single off-topic line being spoken across all three conversations. A set of 13 questions were asked with 12 answered successfully by the crowd during conversation. These results are summarized in Figure 4. For these conversations, collective memory was not needed.

Memory

We ran four memory experiments. In each, we held 3 conversations spanning a total of 259 lines suggested by the crowd. This was filtered down to 104 lines by the curation system. In a majority of trials, the chat history contained potentially useful information or highlights almost exclusively. However, in one trial, a single user from a large crowd flooded the selection area with irrelevant answers, blocking out other suggestions and leading to a significant source of error. This suggests that selective muting features should be added to the interface to help workers filter more efficiently, helping to mitigate the impact of actively malicious workers. Alternatively, we are exploring the effects of imposing a rate limit on the number of responses that can be given to one query by the user.

The crowd was frequently successful in recalling facts using working memory. We separately recorded both successful use of the working memory to recall facts and failure to do so (see Figure 4). In 8 of 10 conversations, the crowd was able to recall facts stored in the working memory from prior sessions, but in the remaining 2 scenarios, required prompting. We found that prompted crowds displayed a cascade effect, where workers were rapidly able to identify forgotten facts when prompted and mark them as salient.

Additional Observations

Interestingly, in some cases the crowd provided serendipity unavailable in traditional autonomous search, with workers suggesting outcomes not considered by the original participant and building on one another's responses. For instance, in a typical example, the requester provided Chorus with the query, "I am looking for a place to eat dinner." Chorus responded with "There is Asian Fusion joint on 3rd and Molly Ave. It serves amazing cuisine." Following a brief interlude, the crowd provided the additional, unsolicited suggestion, "I also saw a Groupon for 1/2 off!" This occurred commonly: in 26 accepted lines of discussion, workers prompted requesters with additional information concerning aspects not initially suggested as part of the starting query. These results (an effect of the information foraging behavior of groups) are deeper than those that would be obtained through search on a site such as Yelp, and suggests that there is additional utility in using conversations with humans as an intermediary when interacting with software. In particular, a crowd-based dialogue partner can significantly augment and improve the utility of software interfaces by providing parallelized search and the ability to recognize and provide relevant facts based on their personal knowledge.

DISCUSSION AND FUTURE WORK

The ability to hold consistent, in-depth, conversations with the crowd opens up a range of interaction possibilities. In this section we focus on potential improvements that can be made to Chorus to improve performance, longer-term issues with reliability in critical situations, and some of the implications for users' privacy.

Future Studies

Our tests showed that the crowd was able to generate consistent and helpful responses to users with reasonably high reliability. We are currently working on deploying this system in a variety of situations, including more complex software such as a restaurant reservation and flight booking applications. We are also investigating the idea of 'embodied crowds' via a mobile crowd-powered information kiosk that can answer general questions using Chorus to present users with a conversational interface.

For many applications such as these, we are also interested whether or not the crowd can respond with high quality feedback, compared to what an individual might present. Quality could be gauged by not only if the user's need was met but how well it was met. For example, in the restaurant finding task, proposing McDonalds as a solution might be valid answer, but not one that is always very helpful to the user. Qualitatively, in both the experiments we present in this paper and in continued testing, the crowd has consistently made very useful suggestions that went well beyond the minimum effort. This behavior is supported by the two-role (proposers and voters) system that requires only a small subset of the workers to seek out high-quality information, whereas the rest need only to identify the relative utility of an answer and agree with the best one by voting.

System Improvements

Initial tests revealed the ability of malicious workers to continuously spam and abuse the voting feature, even when they are not being rewarded to do so. To rectify this problem, we've added restrictions on participation between exchanges with the end user. We found that roughly three total contributions was an effective limit on the number of responses to a single user query. In addition, displaying this information

provides clear feedback that workers cannot submit unlimited contributions prior to their first spam message.

We also observed that some workers mistakenly used the collective memory window as another chat interface rather than filtered history. This appeared to be mostly due to workers not reading the directions fully or trying to bypass the filtering step and speak directly to users (which was not possible). To remedy this, we placed a minimum score threshold that must be met in order to submit free-text input to the crowd memory, ensuring that only previously-successful workers who fully understood the task were eligible to make this type of contribution.

Due to some irrelevant messages passing through our filters, we provided the end user with the ability to reject these messages. The poster as well as any worker who voted for the post, would receive a point penalty. User rejections provide workers with feedback on input and additionally increase incentives for positive contributions.

Complex Agreement

Our tests dealt with straight forward factual and general opinion questions as a proof of concept for the system, however, some use cases of Chorus might require the crowd to answer polarizing or even fictitious queries, such as in the case of using Chorus to portray characters in a virtual environment such as a video game or training program. In this case, the crowd must act consistently with beliefs that might not be held by any constituent member, and prevent biases and contradictory opinions from coming through in the response.

Reliability

The possibility of using Chorus is real-world settings means that important situations will arise in which an answer from the crowd must be trusted and thus needs to be as reliable as possible. While our work shows Chorus is able to remove noise from the crowd's feedback to users, it is not yet clear that it can successfully cope with the input of intentionally malicious workers. If a worker were to propose a restaurant which they knew served a type of food that the user was allergic to but was not obvious from the name, such as "John's Grill" serving seafood, then it is possible the other crowd workers will agree with the option without doing further (unrewarded) research. Similarly, worker's might propose such an option with good intentions, but not do sufficient information gathering to find out subtle but important facts.

In future work, our goal is to find ways to encourage workers to do these type of deep knowledge searches, then use redundant results of deep searches performed by different workers to increase the reliability of the system on critical tasks where the additional cost is warranted.

Privacy

Crowds that are capable of learning about users over time introduce privacy concerns. As with many communication technologies, users must be mindful of the fact that others can view the information they provide to the system and adjust their behaviors and use of the system accordingly. To support this, developers must take care to make users aware of the fact that people, not just software, are used to support their system in order to facilitate these behavioral choices.

APPLICATIONS

Providing a means of two-way natural language communication with a crowd agent who is not only reliable, but is available on-demand, and able to perform tasks autonomously for indefinite amounts of time presents the opportunity for many new systems which were previously not feasible. We discuss a few of those in this section.

Natural Language Interfaces to Existing Systems

One of the most impactful uses of this system is to enable semi-autonomous natural language interfaces for existing systems via both text and voice. In this work we focus on tasks that the workers can individually perform then return with the results, such as a web search task or general question answering. However, to allow users to control specific systems that they might not otherwise have direct access to, we need workers to be able to collectively control the current interface based on the knowledge of the situation and intention, and provide feedback that takes into account the current task and system state. In this paper we outline how context sensitivity can be supported, and show crowds of web workers can effectively maintain awareness of the current task. To *control* the interface itself, an existing crowd control system such as Legion [13] can be used.

Hands-free Interfaces

Using Chorus with crowd controlled interfaces allows developers to create single user interfaces designed to be use in ordinary situations that are still able to be voice-controlled and partially automated without any modification or additional development time. Furthermore, we can retrofit existing interfaces in the same way. We envision this being particularly helpful to disabled users, such as blind and motor impaired users, and to ordinary users who are *situationally disabled* [], such as when driving a car or performing another task that does not allow for the use of a user's vision or hands while performing it.

Crowd Assistants

Chorus paves the way for using the crowd as an assistant who is able to autonomously look up information and process it in the context of past events in order to provide assistance to users. Such an assistant will also be able to learn about the user's preferences over time, making it capable of improving the quality of interaction with the user as it gains experience. In this way, Chorus also acts to add a notion of agency to existing software, allowing them to complete tasks without the user's explicit guidance at every step. In our example scenario, the version of Chorus that Susan is using acts as a crowd assistant operating from her mobile phone via a web interface.

CONCLUSION

We have presented Chorus, a system that allows two-way dialogue between a user and the crowd. Chorus allows the

crowd to have one voice, as if it were a single individual, instead of burdening the user with managing multiple lines of conversation or limiting the interaction to use only one-way communication. Through our example with Susan, who needed directions and a restaurant recommendation while driving, we showed how such a system could be of great use in the real world.

We have demonstrated that the crowd could not only maintain a consistent conversation with users, but also learn and remember over the course of both single and repeated interactions. Finally, we outlined how this approach could be used to more naturally interact with both existing and future crowd-powered systems.

REFERENCES

1. J. F. Allen, L. K. Schubert, G. Ferguson, P. Heeman, C. H. Hwang, T. Kato, M. Light, N. G. Martin, B. W. Miller, M. Poesio, and D. R. Traum. Enriching speech recognition with automatic detection of sentence boundaries and disfluencies. *Journal of Experimental and Theoretical AI (JETAI)*, 7:7–48, 1995.
2. M. S. Bernstein, J. R. Brandt, R. C. Miller, and D. R. Karger. Crowds in two seconds: Enabling realtime crowd-powered interfaces. In *Proceedings of symposium on User interface software and technology*, UIST '11, pages 33–42, New York, NY, USA, 2011. ACM.
3. M. S. Bernstein, D. R. Karger, R. C. Miller, and J. R. Brandt. Analytic methods for optimizing realtime crowdsourcing. In *Proceedings of Collective Intelligence*, CI 2012, page to appear, New York, NY, USA, 2012.
4. M. S. Bernstein, G. Little, R. C. Miller, B. Hartmann, M. S. Ackerman, D. R. Karger, D. Crowell, and K. Panovich. Soylent: a word processor with a crowd inside. In *Proceedings of the symposium on User interface software and technology*, UIST '10, pages 313–322, New York, NY, USA, 2010. ACM.
5. M. S. Bernstein, J. Teevan, S. Dumais, D. Liebling, and E. Horvitz. Direct answers for search queries in the long tail. In *Proceedings of the conference on Human Factors in Computing Systems*, CHI '12, pages 237–246, New York, NY, USA, 2012. ACM.
6. J. P. Bigham, C. Jayant, H. Ji, G. Little, A. Miller, R. C. Miller, R. Miller, A. Tatarowicz, B. White, S. White, and T. Yeh. Vizwiz: nearly real-time answers to visual questions. In *Proceedings of the symposium on User interface software and technology*, UIST '10, pages 333–342, New York, NY, USA, 2010. ACM.
7. L. Chilton. Seaweed: A web application for designing economic games. Master's thesis, MIT, 2009.
8. S. Cooper, F. Khatib, A. Treuille, J. Barbero, J. Lee, M. Beenen, A. Leaver-Fay, D. Baker, Z. Popovic, and F. Players. Predicting protein structures with a multiplayer online game. *Nature*, 466(7307):756–760, 2010.
9. L. Hirschman, D. A. Dahl, D. P. McKay, L. M. Norton, and M. C. Linebarger. Beyond class a: a proposal for automatic evaluation of discourse. In *Proceedings of the workshop on Speech and Natural Language*, HLT '90, pages 109–113, Stroudsburg, PA, USA, 1990. Association for Computational Linguistics.
10. A. Kittur, B. Smus, and R. Kraut. Crowdforge: Crowdsourcing complex work. Technical Report CMUHCII-11-100, Carnegie Mellon University, 2011.
11. A. P. Kulkarni, M. Can, and B. Hartmann. Turkomatic: automatic recursive task and workflow design for mechanical turk. In *Proc. of CHI 2011*, pages 2053–2058, 2011.
12. W. Lasecki, C. Miller, A. Sadilek, A. AbuMoussa, and J. Bigham. Real-time captioning by groups of non-experts. In *In Submission*, 2012. http://hci.cs.rochester.edu/pubs/pdfs/legion-scribe.pdf.
13. W. Lasecki, K. Murray, S. White, R. C. Miller, and J. P. Bigham. Real-time crowd control of existing interfaces. In *Proceedings of the symposium on User Interface Software and Technology*, UIST '11, pages 23–32, New York, NY, USA, 2011. ACM.
14. W. S. Lasecki, S. White, K. I. Murray, and J. P. Bigham. Crowd memory: Learning in the collective. In *Proc. of Collective Intelligence 2012*, 2012.
15. B. Levitt and J. G. March. Organizational learning. *Annual Review of Sociology*, 14:319–340, 1988.
16. G. Little, L. B. Chilton, M. Goldman, and R. C. Miller. Turkit: human computation algorithms on mechanical turk. In *Proceedings of the 23nd annual ACM symposium on User interface software and technology*, UIST '10, pages 57–66, New York, NY, USA, 2010. ACM.
17. Y.-A. Sun, C. R. Dance, S. Roy, and G. Little. How to assure the quality of human computation tasks when majority voting fails? In *Workshop on Computational Social Science and the Wisdom of Crowds*, NIPS 2011, 2011.
18. L. von Ahn. Human computation. In *Ph.D. Thesis*, 2005.
19. L. von Ahn and L. Dabbish. Labeling images with a computer game. In *Proceedings of the conference on Human factors in computing systems*, CHI '04, pages 319–326, New York, NY, USA, 2004. ACM.
20. M. A. Walker, D. J. Litman, C. A. Kamm, and A. Abella. Paradise: a framework for evaluating spoken dialogue agents. In *Proceedings of the Association for Computational Linguistics*, ACL '98, pages 271–280, Stroudsburg, PA, USA, 1997. Association for Computational Linguistics.
21. N. Webb, D. Benyon, P. Hansen, and O. Mival. Evaluating human-machine conversation for appropriateness. In N. Calzolari, K. Choukri, B. Maegaard, J. Mariani, J. Odijk, S. Piperidis, M. Rosner, and D. Tapias, editors, *Proceedings of the Conference on Language Resources and Evaluation*, LREC'10, Valletta, Malta, may 2010. European Language Resources Association (ELRA).
22. H. Zhang, E. Law, R. C. Miller, K. Z. Gajos, D. C. Parkes, and E. Horvitz. Human computation tasks with global constraints. In *Proc. of CHI 2012*, 2012. To appear.