Chorus: A Crowd-Powered Conversational Assistant

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

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

USER Group[2]
IBM Research - Almaden
jwnichols@us.ibm.com

MobileWorks, Inc.[3]
anand@mobileworks.com

Human-Computer Interaction Institute[4]
Carnegie Mellon University
jbigham@cmu.edu

ABSTRACT

Despite decades of research attempting to establish conversational interaction between humans and computers, the capabilities of automated conversational systems are still limited. In this paper, we introduce Chorus, a crowd-powered conversational assistant. When using Chorus, end users converse continuously with what appears to be a single conversational partner. Behind the scenes, Chorus leverages multiple crowd workers to propose and vote on responses. A shared memory space helps the dynamic crowd workforce maintain consistency, and a game-theoretic incentive mechanism helps to balance their efforts between proposing and voting. Studies with 12 end users and 100 crowd workers demonstrate that Chorus can provide accurate, topical responses, answering nearly 93% of user queries appropriately, and staying on-topic in over 95% of responses. We also observed that Chorus has advantages over pairing an end user with a single crowd worker and end users completing their own tasks in terms of speed, quality, and breadth of assistance. Chorus demonstrates a new future in which conversational assistants are made usable in the real world by combining human and machine intelligence, and may enable a useful new way of interacting with the crowds powering other systems.

Author Keywords

crowdsourcing; conversational assistants; dialog systems; human computation; crowd-powered systems

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 robust 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. In contrast, real-world conversations between human partners can contain context-dependent terms or phrasing, require memory stretching back over the conversation and past history of interactions and shared experiences, require common sense knowledge about the world or events, or facts, and contain meaningful incomplete and partial statements. Conversational assistants like Siri were greeted with great excitement when first released, but still possess limited capabilities beyond a finite set of pre-defined set of tasks because they cannot truly *understand* the user [8, 9].

While individual humans have no difficulty in maintaining natural-language conversation, it is often infeasible, unscalable, or expensive to hire a human to act as a conversational partner for long periods of time or to support 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 the difficult aspects that only humans can solve 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[1] or MobileWorks[2]. Crowd computing can provide software with human intelligence often while maintaining the scalability of autonomous software, but presents new challenges in reliability, incentivization, and accuracy [14].

In this paper we present Chorus, a crowd-powered conversational assistant that allows a user to receive assistance on any online task through a two-way natural language conversation. Chorus is capable of performing any task that may be accomplished over the web using information that the user is comfortable sharing. Although it appears to the user that the assistant is a single individual, the actions and responses of

UIST'13, October 8–11, 2013, St. Andrews, United Kingdom.

http://dx.doi.org/10.1145/2501988.2502057

[1] www.mturk.com
[2] www.mobileworks.com

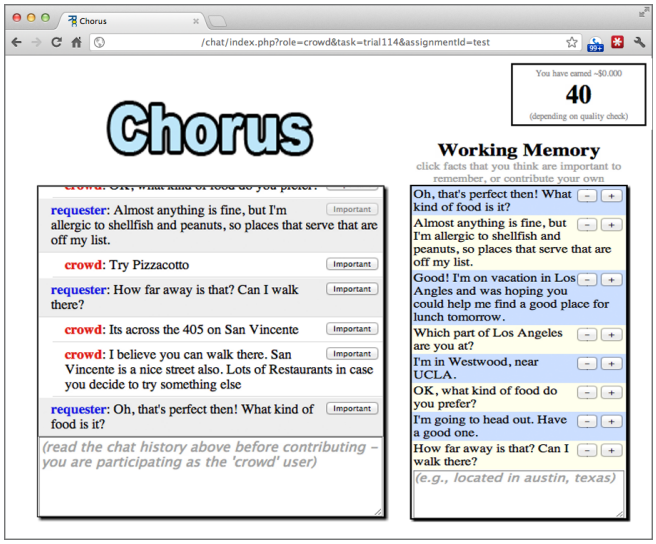

Figure 1. The Chorus interface for crowd workers encourages each worker to both propose responses and vote on the responses of others. Crowd workers can also make use of a collective shared chat history (working memory) that allows them to maintain continuity throughout the conversation, even as some leave and new workers join. The memory view is curated by a separate set of crowd workers. The "user" view is similar but removes the "Working Memory" section and only displays proposed messages once they have sufficient votes.

the assistant are actually the combined effort of a number of crowd workers that may come and go over the course of a single conversation. This approach enables real-time interaction, personalized answers, and takes the first steps towards robust conversational interaction for control over other systems, in addition to question answering.

As part of the crowd worker interface to Chorus, crowd workers generate, agree upon, and submit responses to end users (Figure 1). Key to Chorus are three components that support realistic conversations from multiple crowd workers. First, a *collaborative reasoning* system lets workers select reasonable responses from a number of crowd-produced suggestions, which allows the best responses to be forwarded on to the end user and removing potential responses that do not fit the flow of the conversation. It also allows the crowd to decide when a response is "good enough" and not to wait for another potentially better response. Second, a *dynamic scoring system* rewards workers for interactions that support the goal of consistent conversation, such as making useful statements or highlighting facts that are used later. Finally, a *curated memory system* lets workers promote salient portions of a conversation to a shared memory space, which provides a rapidly-accessible list of facts and statements that are sorted by importance (again judged by the crowd) that ensures new crowd workers are up-to-date with details of current or past conversations that may be relevant for providing future responses. These features enable the crowd to learn and remember information collectively, and allows crowd workers to be rewarded for their performance.

Our experiments with 12 users and 100 crowd workers demonstrate that Chorus can provide highly accurate and on-topic responses, answering over 92.85% of queries appropriately, and staying on-topic in nearly 96% of accepted responses, significantly better than unfiltered crowd responses. Using multiple workers also allows the average task time to fall from 103.4 seconds with just a single worker to 44.6 seconds because multiple workers have a better chance of finding the correct solution more quickly because they are able to search in parallel. Beyond user performance, we gain insight into user's usage and preferences regarding real-time personal assistants. More generally, Chorus demonstrates how crowd-powered communication may serve as a robust alternative for interacting with software systems.

The primary contributions of this paper are:

- Chorus, a crowd-powered conversational assistant able to assist end users consistently with general knowledge tasks;
- A game theoretic incentivization mechanism that encourages crowd workers to participate as part of a consistent conversational assistant;
- An evaluation with 12 end users and 100 crowd workers that both shows the advantage of our approach and offers several interesting qualitative observations about how crowd members work together to answer questions; and
- The description of a path forward from crowd-powered conversational assistants that will gradually introduce more machine intelligence into the process in order to lower costs and improve scalability.

RELATED WORK

Chorus builds on prior work in both real-time and offline human computation. Human computation [25] has been shown to be useful in many areas, including writing and editing [5], image description and interpretation [7, 26], and protein folding [11]. 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 [22, 13]. For instance, the ESP Game uses answer agreement [26] and Soylent uses the multiple-step find-fix-verify pattern [5]. Since these approaches take time, they are not always suitable for interactive real-time applications.

Many crowd computing applications have used the crowd to interpret natural language instructions provided by the user in applications such as image description [7], speech recognition [16], activity recognition [18], interface control [17], document editing [5], and even vacation planning [29, 15]. However, such systems require only a single round of communication, from the requester to the worker and back. The reason for this is that maintaining consistent communication with the crowd is inherently difficult because the pool of online agents is always 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 quickly or intelligibly for various reasons including misunderstanding of task directives, laziness, distractions, 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 itself.

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 [6]. CrowdSearcher [2] introduced a model in which social search was combined with automated search by way of a structured query model.

Most Internet forums rely on groups of workers to answer 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 that 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 systems 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[3] 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 [7] 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 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 [26, 10]. Adrenaline combines the concepts of queuing and waiting to recruit crowds (groups) in less than 2 seconds from existing sources of crowd workers [3]. 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 [4]. Work on real-time captioning by non-experts[16] 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 [17]. 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 [21]). Organizational learning has previously been demonstrated in continuous real-time crowdsourcing using Legion [20]. 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 [12]. This approach falls short when rating dialogues that may not follow the fixed expected path, even when the response itself is valid. PARADISE [27] 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 [28] 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

This section presents the Chorus system, first at a high-level through a scenario, and then through a discussion of the key challenges that Chorus faces and how they were addressed to create a functional system. A diagram of the Chorus system can be seen in Figure 2.

[3] www.chacha.com/

Scenario

To demonstrate the utility of crowd-powered conversational interfaces, we consider the following scenario. Susan is a mother of 3 who uses Chorus as an autonomous personal assistant while driving from work and picking up her daughter. When Susan is driving, she is situationally disabled [23], 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 a store in downtown Chicago, IL to purchase a present for a birthday party, then needs to quickly head over to pick up her daughter and take her there. Since Susan does not know what kind of present is appropriate and is in a hurry, she asks Chorus to recommend a present for a 5-year-old boy that she can buy on the way home.

Individual workers in the crowd can listen to Susan's request, interpret the instruction and then develop their own response. These responses must then be merged into getting a single, unified response, which is a key challenge for Chorus that we call *achieving agreement*. Our method for achieving agreement involves a combination of crowd voting and similarity matching between existing answers, which we will discuss later. When a proposed response has sufficient agreement it is "locked in," presenting it to the user (Susan) via speech or text. Using workers for both response generation and selection has several advantages. First, involving multiple workers leads to an appropriate response more quickly and with more options explored than a single worker could do alone because each worker works in parallel with the others and typically on a different path because of the intrinsic differences between workers. Second, using workers to help achieve agreement improves accuracy since people are generally able to identify correct answers better than they can generate them [24].

Once Susan has asked for suggestions, one worker quickly returns "a Furby," but others have looked up Amazon reviews online and know that this product is not recommended for 5-year-olds. Workers propose a variety of alternatives at first, including some that cost over fifty dollars. The responses with high-priced items do not receive as many votes however, because most workers have noticed that in a previous conversation Susan had said she typically spends around twenty dollars on a gift. This is revealed in the "Working Memory" section of the crowd workers' interface. Ensuring that memory of previous interactions is integrated into crowd workers' decisions is another key challenge addressed by Chorus.

Some workers propose purchasing a specific action figure for twenty dollars, and find a nearby toy store that lists it in stock via its website. Finally, while verifying the route to the toy store, 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 an alternative store close to Susan's work: "take I-90 to the toy store on 444 W. Chicago Ave to avoid traffic on I-55 and an accident on Archer St You can buy an action figure there for twenty dollars." Other workers quickly check this option and see it is the best one. Within moments they switch their vote, and the action figure option is forwarded to Susan and spoken out loud by the application. The agreement and memory features of Chorus have ensured that crowd workers were able to generate a high quality response while taking into account information available to them from current and past conversations.

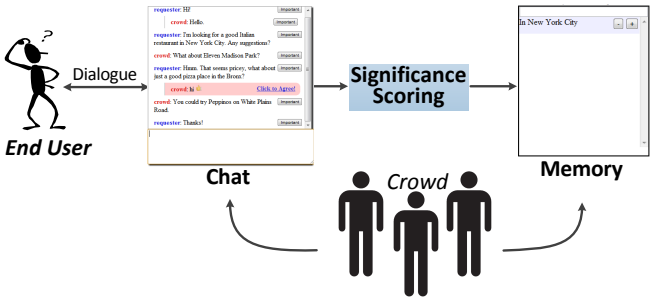

Figure 2. The Chorus system. Users first make a request, which is forwarded to the crowd. Workers can both submit responses and vote for those proposed by others. Responses are hidden from users until sufficient agreement between workers is achieved. Workers can also add information to the crowd's working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

Challenge: Recruiting and Training Workers

To recruit workers in near real-time, we use a retainer pool [3] and a thresholding mechanism to ensure that a certain number of workers arrive to each conversation task simultaneously and when needed, similar to the method used in legion [17]. The number of workers required is configurable, and we explore this in our studies, described later.

To ensure that the workers were educated in the use of the interface and to test their competence for the task, we showed workers a 30 second video describing how to use the interface the first time they joined a task. After showing the video, we then sent workers through two guided tutorials before they could participate in the main task. The first tutorial required workers to correctly propose a response given a message and the second required the user to vote on a set of potential responses given a message. Both of these tutorials provided guidance on how to use the interface, and the correct answer for each tutorial task was extremely simple to deduce. Feedback was provided for incorrect answers and workers had as many opportunities as they wanted to try again if they failed. While these tutorial tasks were *very* simple, they had the effect of filtering out many lazy, idle, malicious, and confused workers, as well as any bots. Filtering the idle workers was particularly important, because in early trials many workers would join the retainer queue but then become idle while waiting for the task to begin.

Informally, we found that those that passed these tutorials were generally more engaged in the task and less likely to become idle. We also found that a majority of the workers who visited the tutorial pages did successfully enter the task.

Challenge: Achieving Agreement

There are at least two possible methods that Chorus could use to identify when agreement has been achieved among the crowd workers: an explicit voting method where crowd workers vote on others' responses, or an automatic method that attempts to identify agreement automatically by comparing the similarity of multiple suggested responses. In the current version of Chorus, we focus on only the explicit voting method.

The crowd worker interface for Chorus (Figure 1) shows all of the suggested responses and allows workers to vote on those responses. Workers may change their votes at any time, including retracting their votes. Chorus requires a majority of workers to agree on a response in order to lock it in, and the majority is calculated based only on the currently connected workers and their votes. If a worker disconnects from the system then their votes are removed, which prevents workers from trying to game the system by voting for a lower quality post and then disconnecting to lower the number of votes required to achieve a majority.

While the use of a voting method seems straight-forward, it is important to design a reward scheme that motivates workers to think carefully about their responses and their votes. A naive award scheme might reward workers for each response and vote that they put into the system, but this approach will simply yield a large number of low quality responses instead the higher quality responses that we wish to encourage.

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 1000 points correlates to 1 cent. At this rate, workers can reasonably earn above the U.S. minimum wage for responding to questions and performing simple search queries, while still providing a reasonably priced service to the user. Furthermore, 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 sent without any user feedback by reducing the value of each subsequent contribution. Reward values are reset with each user input. This means that the points given to workers for each contribution to the same response decreases, removing some of 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 without some kind of user input does not limit the system's functionality, so absolute limits can also be enforced if desired (we limited workers to 3 contributions per user message).

Challenge: 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 that are 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 extensive logs.

To support memory, we add the "Working Memory" section to the crowd workers' user interface. This area shows lines from previous conversations and summarized facts that may be useful when formulating responses.

In the initial version of the system, crowd workers were allowed to curate the memory while also producing and voting on potential responses. In this version, workers could move lines of dialog from the chat window into the working memory area if they believed that it would be useful later, or contribute summarized facts by entering them into the text field at the bottom of the section. Workers could also increase or decrease the importance of each line in the crowd's memory by voting, and the lines were sorted in descending order.

In our studies of the initial prototype, described later, we found that the overhead of asking users to perform both the "chat" task and this "memory curation" task was too much. For this reason, in the current version of the system we split these two tasks to different sets of crowd workers. Workers in the chat task are no longer able to add items to the working memory or change their order in the list. A second set of workers is then recruited for the memory task. They see a new interface that allows them to browse the conversation history, add items to the memory by selecting lines of dialog or typing in their own summaries of facts, and vote for facts they think are important, but cannot participate in the chat.

The chat task and memory task can proceed simultaneously, but work on the memory task does not have the same real-time requirements of the chat task. In fact, workers on the memory task can only be productive if there is a sufficient amount of new conversation content to process. Thus, workers for this task can be recruited only occasionally during a conversation depending on its speed and length, or after the end of a conversation.

Contributions to the collective memory are rewarded using the same multi-tiered point system described above for the chat task, although this may not be ideal given that this task is somewhat different. Further investigating the best reward scheme for the memory task is left for future work.

Challenge: Keeping Workers Engaged

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. Furthermore, 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, which makes it easier for them to respond quickly and accurately.

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 when she may need it. It also provides quicker response time in many cases because workers are able to search for answers on many fronts at once.

Interface Design

To this point, we have focused on a particular interface design that forwards only the single agreed upon response to the user. This masks the fact that multiple crowd workers are actually providing responses in the back-end and creates the impression of talking to a single individual. We call this the Filter version of the interface.

We have also experimented with an alternate interface called Suggest, which allows the user to see multiple responses generated by the crowd as they are working. Some filtering is still performed on the responses, but at a reduced level compared to the Filter interface. This interface allows the user to provide feedback on the responses as they are being generated, which may help guide the crowd to provide a better answer even sooner.

In both interfaces, we also present the users and workers with a "typing" status indicator that tells the other party that a response is being composed, similar to the feature found in many instant messaging clients. 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.

FORMATIVE STUDY

To begin exploring how to use the crowd as a conversational partner, we conducted a formative study with an early version of Chorus. This version used a simple voting mechanism that did not remove votes for users that disconnected before the required agreement threshold was reached, allowed crowd workers to curate the collective memory directly, and did not reduce reward values for actions leading to a single response. In addition, workers were not shown a tutorial before they began the task.

We recruited a 23-year-old user who had no prior experience with Chorus to have a series of conversations with it. This user carried out conversations starting with one of two questions: (i) Will dog breed `X` be good for my apartment? and (ii) what is the best place to eat near `Y`? The dog type and city in these questions were altered each iteration just in case any of the workers overlapped from a previous trial.

	Total Lines	Accurate Responses	Errors Made	Clarifications Asked	Questions Asked	Answers Provided
Consistency #1	24	9	0	0	4	4
Consistency #2	55	50	1	0	7	6
Consistency #3	33	11	0	0	2	2

Figure 3. Results for the conversations with Chorus..

In our analyses, we reviewed several hundred lines of conversation and interaction with Chorus, focusing on overall performance and conversational consistency. Over the course of the study, Chorus recruited a total of 33 unique workers, with an average of 6.7 workers contributing to each trial. Overall, Chorus filtered out 37.5% crowd responses and answered 84.62% of inquiries correctly (with useful information).

To investigate conversational consistency, we ran three experiments. In each, we used 3 conversations spanning a total of 112 lines suggested by the crowd. Each of these lines was hand-coded by two researchers as "on topic" or "off topic." This was filtered down to 70 lines by the curation system, and among these only a single off-topic line was spoken across all three conversations. A set of 13 questions were asked with 12 answered successfully by the crowd during the conversations. These results are summarized in Figure 3.

Throughout these experiments, we observed the crowd providing serendipity unavailable in traditional automated 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 was a relatively frequent occurrence: 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.

EXPERIMENTS

To test the feasibility of using the crowd as a conversational partner, we performed experiments focusing on two different aspects: *(i)* conversational consistency (measured in terms of topic maintenance and coherent answers), and *(ii)* memory of past events (measured in terms of factual recall).

Conditions

Based on our formative study, we designed 3 variants of Chorus that we used in the study. We also included a control condition in which users simply searched the web for their own answers. The conditions were as follows (Figure 4):

- **Filtered:** This is the complete version of Chorus, which asks a group of workers to collectively propose answers to user's questions, and vote on one another's responses to find the best ones. This approach benefits from finding the best answers from a range of people, each with different sets of prior knowledge and approaches to search. However, requiring an explicit voting step prevents answers from being forwarded as quickly as possible.
- **Solo:** In this condition, Chorus is powered by a single worker. This means that there is no underlying collaborative search, or wide variety of prior knowledge. However, it also means that the system does not have to wait for workers to agree with one another, potentially reducing the response time.
- **Unfiltered:** In this condition, Chorus uses a crowd to generate responses, but does not require any agreement to forward them to the end user. This means users can benefit from the diversity of the crowd, while not reducing the speed at which individual answers are provided. However, because responses are not filtered, unhelpful or redundant responses might be included. Thus, this might not be as helpful to users who are occupied with other tasks during their search (e.g. driving or holding another conversation).
- **Google:** The fourth condition asked users to find information themselves using Google. This acted as a baseline for their feedback on the system.

Since chatting via an instant messenger is often not done as a stand-alone task, we allow users to use the web on a different task during our tasks, returning when they received a new message from the system.

In our experiments, the crowd was recruited from a combination of Amazon Mechanical Turk (90% approval rating or higher, composing 77% of our workers) and MobileWorks (no restrictions, composing 23% of workers). These workers were paid 25 cents to complete the interface training, plus an additional three cents for each contributed message accepted by Chorus, and one cent for each vote towards an accepted message contributed by another worker. Our tests used a total of 100 unique workers with at least 4-5 workers beginning each Filter and Unfiltered session (as well as workers coming and going throughout), and exactly 1 worker at any given time during single-worker sessions (described below).

Conversational Consistency and Dialogue

In our experiments we focused on the crowd's ability to generate reasonable responses to user questions. To help control for the potential complexity user questions and ensure results were not skewed by the varying types of participants' impromptu questions, underspecified task descriptions, or particular communication styles, we generated a script for each of our four tasks and asked our participants to follow the

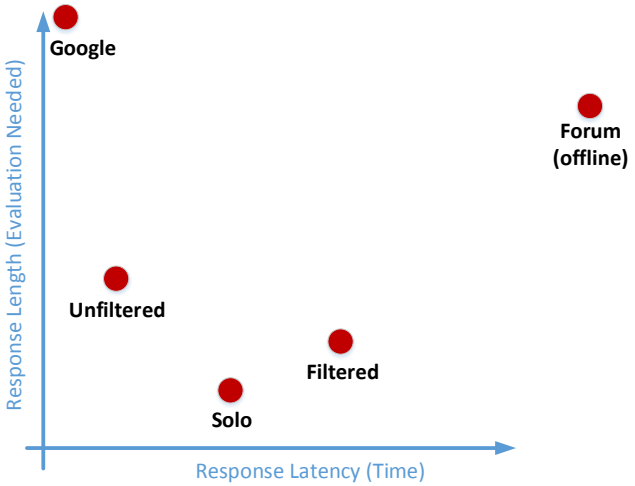

Figure 4. Tradeoff between speed and ease of parsing results. While Solo results in a small set of responses and reasonable response time, the answers also lack variation. Both Filtered and Unfiltered approaches give more broad answers interactively, but Unfiltered returns a result quicker. Older, offline methods, such as Forums, can return varied answers but don't return answers in real-time.

script as closely as they could, while still allowing the conversation to flow naturally as if would if they were talking to another person (all while following the persona laid out in the script). We used the following task goals to generate scripts:

- **(i)** Find information about things to do when traveling to Houston, TX that are not "too touristy".
- **(ii)** Discuss the options available for getting a dog, given the constraints of a small apartment near a park in Washington, D.C.
- **(ii)** Find an interesting dinner date idea that is more unique than just dinner and a movie.
- **(iv)** Get step by step directions about how to make lasagna. Users remember only after an initial response that they failed to mention the dish they want to prepare must be meat and gluten free.

Specifics of the tasks, such as location or dog size, was varied between trials in order to prevent workers from being able to just repeat responses from previous trials. Each participant was asked to complete all four scripts using three different versions of Chorus as well as Google to get a baseline for finding a solution to one of the scripted problems using a conventional search engine. 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, meaning that each test was treated as the crowd's first contact with a given user, and in fact the user had a different identity they played for each question asked. To prevent workers who may have previously completed a session from having an unfair advantage, we altered task details such as user city or preferences. As a baseline, we also compared users' performance on a conventional keyword search engine to find information. While we expect keyword search to be much quicker, because the user must define their own

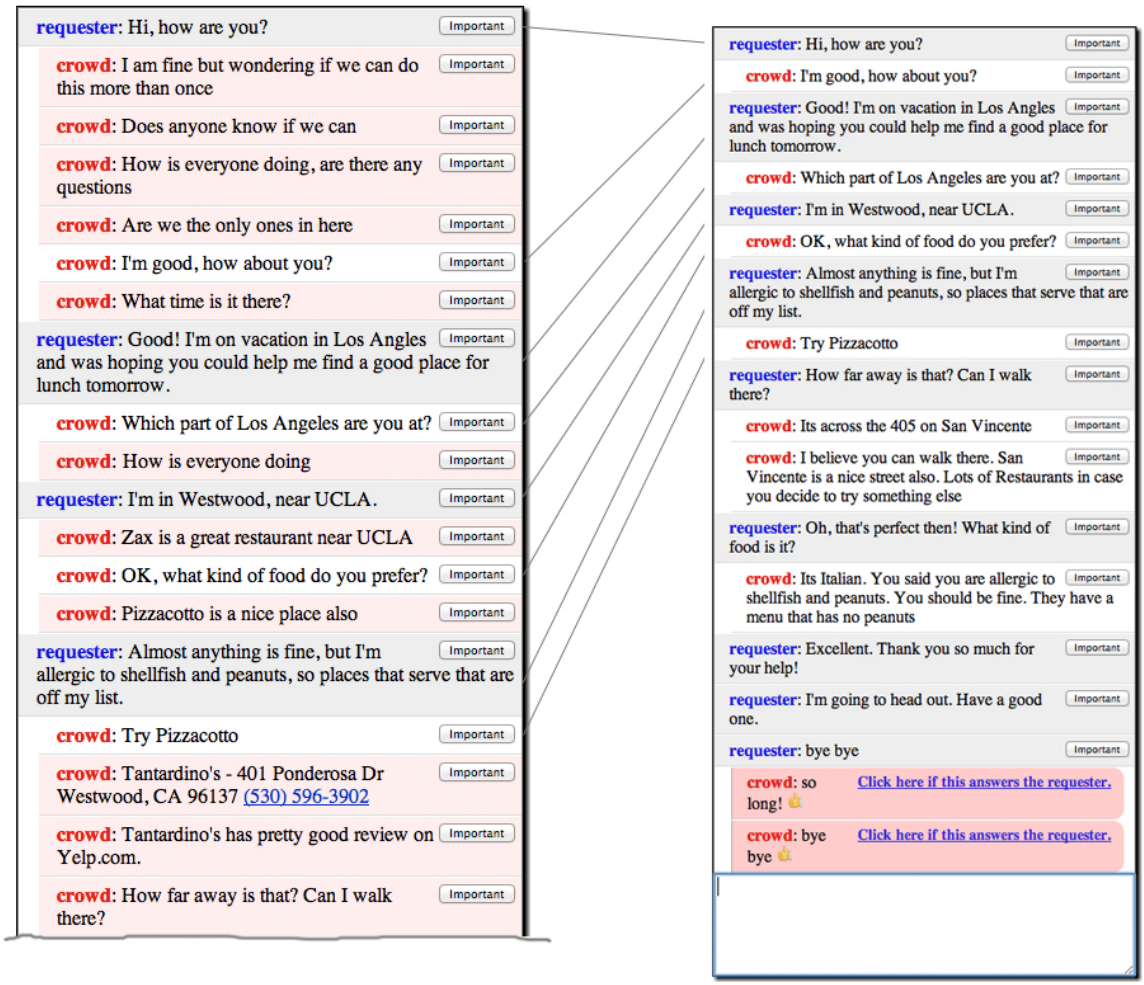

Figure 5. A clipped view of the raw chat log (left), and the filtered view seen by the requester (right). Messages in pink did not receive sufficient votes from the crowd to make it into the requester view. Some of these messages did not contribute to the conversation (e.g. new workers asking questions about the task in the beginning), while others were simply not chosen.

terms even when unfamiliar with a space, there is less potential for finding a useful answer that may not be closely related to the initial search terms.

Trials were spaced out over a period of four days and the order of the conditions randomized. We recruited 12 participants to act as users and hold conversations based on our scripts. They were asked to hold a conversation until Chorus provided an answer to their question (including followups), 15 minutes passed without a single response from the crowd, or 30 minutes total passed without a correct answer being given. Once done, four 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.")

- **Side Chatter:** Conversations between workers either about the task itself, or general conversations and queries.

- **No Code:** The response is blank, a correction or typo, or otherwise unintentionally not interpretable. This does NOT cover messages which are explicitly spam, which are classified as off-topic.

Each of the 960 messages in our dataset was coded by these four researchers, resulting in a Fleiss' Kappa score of 0.67 (strong agreement). Their ratings were then combined with even weight to score the success of the system. We found that 88.3 % of messages were classified as On-Topic, 9.3% of messages were Off-Topic, and 1.7% consisted of Side Chatter between workers. Out of all messages, just 68.6% were accepted, with a final accuracy visible to the user of 95.60%. This relative improvement of 8.3% was significant ($p < 0.05$).

We also tracked the number of user-generated queries and clarification details that were addressed completely in answers, partially addressed, or not addressed at all by the crowd. This

gives us a measure of how well the crowd listened to the user (as opposed to just spouting off answers without correct details). We considered an information-containing message as addressed when the response did not violate any of the facts in the message, partially addressed if it violated some but not all, and not addressed if the response did not take the information in the message into account at all. We used the same rating scale for questions, but required that workers provided an *actionable* solution to the users. 92.85% of questions asked by users were addressed completely, another 3.38% were addressed partially, and only 3.77% were not addressed. In addition, many of those questions and facts that the crowd failed to answer or use were less important tidbits and followup questions similar to those that often go unseen the first time in any conversation or instant messenger. These results demonstrate that the crowd is very attentive to the information that users provide.

Individual Workers and Crowd Assistants

Across 24 tests involving the crowd-powered version of Chorus, 12 with filtering and 12 without, users' information goals were met 100% of the time. In contrast, when a single crowd worker was recruited to answer questions, the system was unresponsive in 2 tests, and failed to finish within the 30 minute bound in 2 more. This means that in a third of all uses, the single-worker version of the system could not complete the task requested. Workers might have been unresponsive because workers did not fully understand the task, queued multiple tasks (contrary to the task instructions) and were planning to get to ours later, lost connection, or some other unknown cause. Workers not responding in a reasonable amount of time may have likewise been for a variety of reasons such as a poor connection, long search times for a single piece of information (which took longer than 15 minutes to complete), or in a few cases, workers leaving as soon as they did the minimum amount of work possible. The reliability of the crowd over single workers recruited from these platforms is one of the primary benefits to using a group in user interaction domains, alongside improved response speed and increased opportunities for creative and novel answers.

Response Speed

To answer a question, workers either need to have prior knowledge of the answer, or find an answer via a web search. On average, individuals have less prior knowledge than a group does collectively, and so we expect individuals to need to search more often. Workers generally disengage from the dialogue to perform a search. Chorus is designed to leverage the prior knowledge of the group. When using Chorus, some workers can continue to engage in conversation with users to help find more helpful points, while others who believe they can find the answer go to search for specific information. Because the collective prior knowledge is larger, many questions may be answered without the need to search at all.

These factors combined result in a significant reduction in the average time until the user gets a first response from the crowd for a given question. Using a single worker, the average response time between a question and a response was 103.4 seconds, likely due to workers already being engaged in a search, or doing other tasks between responses. In the Suggest condition, there was a significant reduction of 56.9% ($p < 0.01$) in this delay, resulting in only a 44.6 second average response delay. The Filter condition also had a significant decrease of 51.5% ($p < 0.05$) over the single-worker case, with an average latency of 50.13 seconds.

	Total Lines	Accurate Responses	Errors Made	Clarifications Asked	Questions Asked	Answers Provided	Memory Successes	Memory Failures
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 6. Results for the conversations with Chorus including memory.

Individual conversations with Chorus averaged 11:21 minutes (median 10:51). While no user took more than 5 minutes on Google, a majority of people independently commented that they preferred Chorus due to the answers they received being easier to scan through and containing suggestions that would not have come up had they simply used a keyword search. In many cases this involved the crowd clarifying a question or suggesting a different search parameter (i.e. type of food in a given city specializes in), which would not have been done by a search engine. As a result, overall search time is can be a deceptive measure, since longer interactions frequently lead to much higher satisfaction with the final result.

Answer Diversity

Online searches for information can be limited by a user's preconceived notions of how to describe the information that is sought. The crowd permits access to a wider range of experience, prior knowledge, search methods, and creativity than any single individual. We observed that the crowd was able to come up with both diverse sets of answers, as well as diverse sets of variations on a given topic. For instance, when asked to provide a set of recommendations for restaurants in the Los Angeles area, crowd members suggested both commonplace suggestions as well as highly unusual ones. This can lead to more effective information provided to end users.

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 they are 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

requester: "Hi"
crowd: "hello"
Requester: "Do you know of 2 or 3 fun things that I can do in Houston that aren't too touristy?"
crowd: "A good place to eat in Boston is Neptune Oyster"
crowd: "Are you on a budget?"
requester: "Yeah I would rather they aren't too expensive."
crowd: "you always need to look at how expensive they are"
crowd: "this places can be very expensive"
crowd: "something good that i can request is for you to use google map"
crowd: "what kind of things are you looking to do? concerts, sports, etc?"
crowd: "Try googling eating on a budget in Boston."
requester: "Anything that's fun but not too touristy."
crowd: "and look for the areas you want to visit and check out the nearby restaurants"
crowd: "Do you enjoy outdoor activities?"
requester: "Yes."
crowd: "well if you go to Boston they have so many free"

Figure 7. In this conversation, the user is asking about Houston, but one of the crowd workers is confused and suggests facts about Boston. These proposals (in red) are not selected to be forwarded on to the user.

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.

Between-Task Consistency

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. 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 (Figure 6). 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. This is likely because a user implying that the crowd should already know a fact makes the workers seek the information within the interface, helping them to notice previously missed information, then recover information that they initially bypassed.

DISCUSSION

The ability to hold consistent, in-depth, conversations with the crowd opens up a range of interaction possibilities. Our tests showed that the crowd was able to generate consistent and helpful responses to users with high reliability. We are currently working on deploying this system in a variety of situations, including more complex software such as 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.

Quality and Reliability

For many applications, we are interested in whether or not the crowd can respond with high quality feedback compared

requester: "thanks! how much does the space center visit cost?"
crowd: "GENERAL ADMISSION AT THE GATE ONLINE DISCOUNT Adults: $22.95 $11.45 Children (Age 4-11): $18.95 $9.45 Seniors: $21.95 $10.95"
requester: "great, and how much would it cost to do the houston zoo, the miller outdoor theatre,the space center, bowling all together?"
crowd: "How many adults and kids would be going?"
requester: "just a single adult"
crowd: "How long will you be staying in Houston?"
crowd: "It will about 150"
requester: "two days (full days)"
crowd: "It will cost around $150"
crowd: "I will be less if you buy tickets online!"
requester: "ok thanks for your help"
crowd: "You Welcome.Have a good day"
crowd: "There might be a city pass available from the local tourist office."
requester: "what does the city pass allow me to do?"
crowd: "CityPass attractions include: Space Center Houston (NASA) Downtown Aquarium Houston Museum of Natural Science Option ticket: Choose to visit either Houston Zoo or Kemah Boardwalk Option ticket: Choose to visit either Museum of Fine Arts, Houston or The Children's Museum of Houston "
requester: "perfect, i will do that. bye!"

Figure 8. In this conversation, the user was concerned with price. Not only does the crowd provide the information the user requested, but it also finds a "City Pass" – a less expensive way to do the activities that the end user wants to do that he didn't know about.

to what an expert individual might present. 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.

The possibility of using Chorus in real-world settings also 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. A particular challenge is encouraging users to dig deeply into possible solutions. For example, a worker might propose an inappropriate restaurant to a user with a particular allergy, perhaps because the food served at the restaurant was not obvious from the name (e.g. "John's Grill" serving seafood). It is possible the other crowd workers will agree with the option without doing further (unrewarded) research.

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.

Difficulties

While our results showed very high rates of correct responses and low rates of unhelpful responses, there are still areas of difficulty. For instance, using the current method for curating crowd memory, only a limited number of facts can be remembered at once without requiring workers to complete a search task within the recorded memories themselves. Thus, over a

requester: "Hi, I want to know how to make lasagna?"

crowd: "Vegetarian or with meat?"

requester: "With meat, but I don't eat red meat."
crowd: "You can find a recipe for a lean lasagna made with turkey meat here: http://www.epicurious.com/recipes/food/views/Lean-Lasagna-230145"

crowd: "It makes 8 servings and takes 1 hour."

crowd: "Would you like another recipe?"

crowd: "Does that use any mushrooms?"
crowd: "It uses 1/2 cup chopped mushrooms. Would you prefer one without mushrooms?"

requester: "Yes, how about a recipe without mushrooms."
crowd: "Here is one made with Italian Turkey sasuage, no mushrooms: http://www.chow.com/recipes/30279-turkey-sausage-lasagna"

crowd: "Would you prefer a different recipe? Or does this work for you?"

Figure 9. In this conversation, the crowd helps the user search for a recipe appropriate for him to make lasagna. Not only does the crowd find relevant web pages and refine their search based on user feedback, but they also ask relevant questions about dimensions the user did not think to specify upfront (such as if the recipe should be vegetarian).

longer span of interactions, most facts would not be remembered by the assistant. in the future, our goal is to create a hybrid approach that learns from the importance of certain facts based on the topic of conversation, then is able to retrieve older pieces of information and display it to workers.

Payments

Another significant challenge in crowdsourcing systems that ask workers for variable input faces is adequately rewarding workers for their contribution. This is especially true of Chorus, where workers might spend anywhere from a few seconds to several minutes finding a response, but are only rewarded based on the number of answers accepted by other workers. While our incentive mechanism makes spending more time result in a better response will increase the likelihood that other workers find it important, there is still no way to determine exactly how valuable the response is.

One way to help solve this problem might be to vary the payment to workers based on the magnitude of the crowd's agreement with their response, since we observed that almost unanimous agreement (among active workers) on answers that were clearly useful or especially helpful. Another approach might be to let the user rate the helpfulness of each answer, which might also prove helpful to give workers ongoing feedback, or train automated conversational assistants.

From reviewing the results for our tasks though, we feel that workers were adequately compensated in the end. In fact, many even emailed us to say they would like to take more of these tasks if possible. This indicates both fair pay, and that workers like the conversational task.

FUTURE WORK

Chorus presents a wide range of potential future directions of work involving systems that have previously been limited by the robustness of automated conversational interaction systems, as well as providing a model for future crowdsourcing workflows. In this section, we discuss some of the potential issues that might arise for such applications.

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.

There are also a variety of tasks that Chorus could perform for users if not for issues of privacy. For example, if a user was comfortable giving the crowd their online banking login information, it would be possible for the crowd to pay bills and carry out other sensitive tasks. Such tasks might become more palatable if techniques could be developed to hide such sensitive information from the crowd workers while allowing them to perform the task. For example, workers might perform web-based tasks through a proxy that automatically hides sensitive information and restricts them from performing known malicious operations.

Merging the Crowd with Machines

The current version of Chorus relies heavily on human involvement in all areas of the interaction, but there is likely room for the strategic addition of machine learning algorithms throughout the process. For example, an automated dialog system could be employed to answer simple requests or an automated system might act as one of crowd workers alongside the other workers in suggesting or voting on responses. The advantage of this approach is that fewer crowd workers might be needed per conversation, or possibly not needed at all in the case of certain conversations, especially for those topics that have come up repeatedly in the past.

Playing a Role

In the current version of Chorus, the virtual agent does not have a pre-defined personality or play the role of a character, but these might be potentially advantageous features for the agent. For example, an agent acting as a customer support representative for a company might be given an agreeable and open personality. In another case, an agent might act as a character in a video game. 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. One way this might be implemented is through integration with machine algorithms, as suggested above. For example, an algorithm might down vote some responses that it identifies as contradictory or manipulate the text of a "locked in" response to be more in line with the pre-defined personality of the agent.

Voice-Enabling Existing Interfaces

One of the most impactful uses of Chorus may be 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. A particularly advantageous use of Chorus might be for disabled users, such as blind [19] or motor impaired users, or ordinary users who are *situationally disabled* [23]. In such a use case, the crowd workers in the chat condition could be given direct access to an existing interface through a crowd control system, such as Legion [17].

CONCLUSION

We have presented Chorus, a crowd-powered conversational assistant that features natural two-way dialogue between the user and the assistant. While the assistant appears to be a single individual, it is actually driven by a dynamic crowd of multiple workers using a specially designed interface. We have demonstrated that Chorus can assist with a variety of tasks across domains, maintain consistency over the course of multiple interactions, and provide a new framework for cooperation among crowd workers. Chorus sets the stage for a rich new area of research in crowd-powered conversational assistants. Such assistants will be able to learn about the user's preferences over time, making them capable of improving the quality of interaction with the user as they gain 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. We believe Chorus represents not only an interesting system for assistance by the crowd, but potentially a more natural way to interact with current and future crowd-powered systems.

ACKNOWLEDGMENTS

This work is supported by National Science Foundation awards #IIS-1149709 and #IIS-1218209, and a Microsoft Research Ph.D. Fellowship.

REFERENCES

1. Allen, J. F., Schubert, L. K., Ferguson, G., Heeman, P., Hwang, C. H., Kato, T., Light, M., Martin, N. G., Miller, B. W., Poesio, M., and Traum, D. R. Enriching speech recognition with automatic detection of sentence boundaries and disfluencies. In *Journal of Experimental and Theoretical AI (JETAI) 7* (1995), 7–48.
2. Bozzon, A., Brambilla, M., and Ceri, S Answering search queries with CrowdSearcher. In *WWW 2012*, 33–42.
3. Bernstein, M. S., Brandt, J. R., Miller, R. C., and Karger, D. R. Crowds in two seconds: Enabling realtime crowd-powered interfaces. In *UIST 2011*, 33–42.
4. Bernstein, M. S., Karger, D. R., Miller, R. C., and Brandt, J. R. Analytic methods for optimizing realtime crowdsourcing. In *Collective Intelligence 2012*.
5. Bernstein, M. S., Little, G., Miller, R. C., Hartmann, B., Ackerman, M. S., Karger, D. R., Crowell, D., and Panovich, K. Soylent: a word processor with a crowd inside. In *UIST 2010*, 313–322.
6. Bernstein, M. S., Teevan, J., Dumais, S., Liebling, D., and Horvitz, E. Direct answers for search queries in the long tail. In *CHI 2012*, 237–246.
7. Bigham, J. P., Jayant, C., Ji, H., Little, G., Miller, A., Miller, R. C., Miller, R., Tatarowicz, A., White, B., White, S., and Yeh, T. Vizwiz: nearly real-time answers to visual questions. In *UIST 2010*, 333–342.
8. Bilton, N. With apple's siri, a romance gone sour. http://bits.blogs.nytimes.com/2012/07/15/with-apple's-siri-a-romance-gone-sour/.
9. Bonnington, C. Steve jobs would have 'lost his mind' over Siri, former employee says. www.wired.com/gadgetlab/2012/05/apple-siri-disappointment/.
10. Chilton, L. Seaweed: A web application for designing economic games. Master's thesis, MIT, 2009.
11. Cooper, S., Khatib, F., Treuille, A., Barbero, J., Lee, J., Beenen, M., Leaver-Fay, A., Baker, D., Popovic, Z., and Players, F. Predicting protein structures with a multiplayer online game. In *Nature 466*, 7307 (2010), 756–760.
12. Hirschman, L., Dahl, D. A., McKay, D. P., Norton, L. M., and Linebarger, M. C. Beyond class a: a proposal for automatic evaluation of discourse. In *Workshop on Speech and Natural Language*, HLT '90, ACL, 1990, 109–113.
13. Kittur, A., Smus, B., and Kraut, R. Crowdforge: Crowdsourcing complex work. In *UIST 2011*, 43–52.
14. Kittur, A., Nickerson, J. V., Bernstein, M., Gerber, E., Shaw, A., Zimmerman, J., Lease, M. and Horton, J. The future of crowd work. In *CSCW 2013*, 1301–1318.
15. Kulkarni, A. P., Can, M., and Hartmann, B. Turkomatic: automatic recursive task and workflow design for mechanical turk. In *CHI 2011*, 2053–2058.
16. Lasecki, W. S., Miller, C. D., Sadilek, A., AbuMoussa, A., R. Kushalnagar, and Bigham, J. Real-time captioning by groups of non-experts. In *UIST 2012*, 23–34.
17. Lasecki, W. S., Murray, K., White, S., Miller, R. C., and Bigham, J. P. Real-time crowd control of existing interfaces. In *UIST 2011*, 23–32.
18. Lasecki, W. S., Song, Y. C., Kautz, H., and Bigham, J. P. Real-time crowd labeling for deployable activity recognition. In *CSCW 2013*, 1203-1212.
19. Lasecki, W. S., Thiha, P., Zhong, Y., Brady, E. and Bigham, J. P. Answering Visual Questions with Conversational Crowd Assistants. In *ASSETS 2013*.
20. Lasecki, W. S., White, S., Murray, K. I., and Bigham, J. P. Crowd memory: Learning in the collective. In *Collective Intelligence 2012*.
21. Levitt, B., and March, J. G. Organizational learning. In *Annual Review of Sociology 14* (1988), 319–340.
22. Little, G., Chilton, L. B., Goldman, M., and Miller, R. C. Turkit: human computation algorithms on Mechanical Turk. In *UIST 2010*, 57–66.
23. Sears, A., Lin, M., Jacko, J., Xiao, Y. When computers fade: Pervasive computing and situationally induced impairments and disabilities. In *HCI International '03*, 1298-1302.
24. Sun, Y.-A., Dance, C. R., Roy, S., and Little, G. 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.
25. von Ahn, L. Human computation. In *Ph.D. Thesis* (2005).
26. von Ahn, L., and Dabbish, L. Labeling images with a computer game. In *CHI 2004*, 319–326.
27. Walker, M. A., Litman, D. J., Kamm, C. A., and Abella, A. Paradise: a framework for evaluating spoken dialogue agents. In *Proceedings of the Association for Computational Linguistics*, ACL '98, Association for Computational Linguistics (1997), 271–280.
28. Webb, N., Benyon, D., Hansen, P., and Mival, O. Evaluating human-machine conversation for appropriateness. In *Proceedings of the Conference on Language Resources and Evaluation*, N. Calzolari, K. Choukri, B. Maegaard, J. Mariani, J. Odijk, S. Piperidis, M. Rosner, and D. Tapias, Eds., LREC'10, European Language Resources Association (ELRA) (May 2010).
29. Zhang, H., Law, E., Miller, R. C., Gajos, K. Z., Parkes, D. C., and Horvitz, E. Human computation tasks with global constraints. In *CHI 2012*, 217–226.