Conversations in the Crowd: Collecting Data for Task-Oriented Dialog Learning

Walter S. Lasecki[1]*, **Ece Kamar**[2], **Dan Bohus**[2]
Computer Science Department[1]
University of Rochester
wlasecki@cs.rochester.edu
ASI Group[2]
Microsoft Research
{eckamar,dbohus}@microsoft.com

Abstract

A major challenge in developing dialog systems is obtaining realistic data to train the systems for specific domains. We study the opportunity for using crowdsourcing methods to collect dialog datasets. Specifically, we introduce ChatCollect, a system that allows researchers to collect conversations focused around definable tasks from pairs of workers in the crowd. We demonstrate that varied and in-depth dialogs can be collected using this system, then discuss ongoing work on creating a crowd-powered system for parsing semantic frames. We then discuss research opportunities in using this approach to train and improve automated dialog systems in the future.

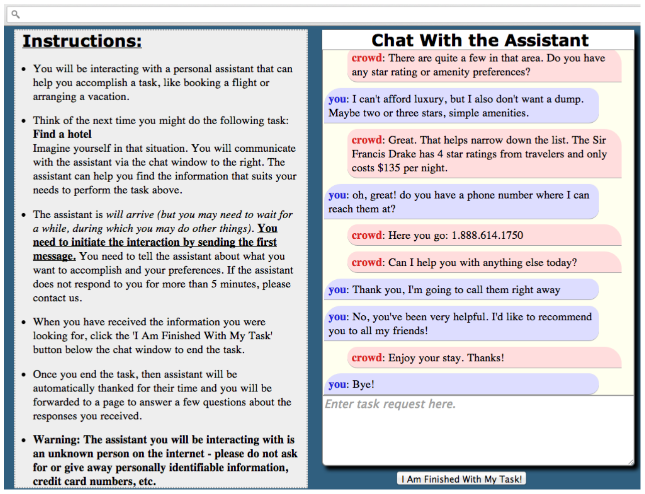
Introduction

One of the primary bottlenecks in the development of task-oriented dialog systems, and in scaling them to multiple domains, is the availability of domain-specific dialog data. Dialog systems harness multiple components, such as speech recognition, natural language understanding, dialog management, natural language generation, and each of these components requires and significantly benefits from the availability of domain-specific data resources and models. Examples include acoustic and language models, spoken language understanding models, domain ontologies, domain interaction plans, natural language generation templates, etc.

Although many AI problems have benefitted from immensely growing data sources, end-to-end data acquisition for task-oriented dialog systems remains a challenging problem. Existing approaches for collecting dialog data result in high development costs and are time consuming for system developers. Unless external resources happen to already be available (which is not the case for most domains), in-domain data collection requires having a deployed system capable of sustaining a dialog with a user. This leads to a bootstrapping problem: given the lack of data to train the initial systems, system developers carry the burden of developing grammars and language models either manually or with Wizard-of-Oz studies. Collecting dialog data with an early version of a deployed system has shortcomings: data collection quality may suffer from the inadequacies of the system itself, and users may bias their language to adjust for the deficiencies of the system in the pursuit of having a successful dialog. As a result, the speed of data collection may be slower than desired. Finally, this costly development process must be repeated all over again for each new domain or system, or even when new functionality is added.

We investigate the use of crowdsourcing for collecting task-oriented text-based dialog data. We present ongoing work on ChatCollect, a system that coordinates workers on a crowdsourcing platform, by pairing them up in real-time, assigning them dialog roles, asking them to accomplish a given task together, and recording their interactions. The system requires minimal input from system developers; it only asks for a description of the task to be given to workers.

We discuss results and lessons learned from an initial experiment, and present statistics that help characterize the trends we observed in these conversations. While further investigation is needed, our initial results suggest that this is a feasible approach for collecting task-oriented dialog data. We discuss future steps to improve the efficiency of data collection with the system. We conclude with a discussion



Instructions:

- You will be interacting with a personal assistant that can help you accomplish a task, like booking a flight or arranging a vacation.
- Think of the next time you might do the following task:
  Find a hotel
  Imagine yourself in that situation. You will communicate with the assistant via the chat window to the right. The assistant can help you find the information that suits your needs to perform the task above.
- The assistant is *will arrive (but you may need to wait for a while, during which you may do other things).* **You need to initiate the interaction by sending the first message.** You need to tell the assistant about what you want to accomplish and your preferences. If the assistant does not respond to you for more than 5 minutes, please contact us.
- When you have received the information you were looking for, click the 'I Am Finished With My Task' button below the chat window to end the task.
- Once you end the task, then assistant will be automatically thanked for their time and you will be forwarded to a page to answer a few questions about the responses you received.
- **Warning: The assistant you will be interacting with is an unknown person on the internet - please do not ask for or give away personally identifiable information, credit card numbers, etc.**

Chat With the Assistant

crowd: There are quite a few in that area. Do you have any star rating or amenity preferences?
you: I can't afford luxury, but I also don't want a dump. Maybe two or three stars, simple amenities.
crowd: Great. That helps narrow down the list. The Sir Francis Drake has 4 star ratings from travelers and only costs $135 per night.
you: oh, great! do you have a phone number where I can reach them at?
crowd: Here you go: 1.888.614.1750
crowd: Can I help you with anything else today?
you: Thank you, I'm going to call them right away
you: No, you've been very helpful. I'd like to recommend you to all my friends!
crowd: Enjoy your stay. Thanks!
you: Bye!

Figure 1: User view of our task interface. We recruit pairs of crowd workers (who assume 'User' and 'Assistant' roles) to hold task-oriented conversations.

*The author did this work while at Microsoft Research of ongoing work on providing training data for spoken language understanding models by having the crowd collectively parse logs from the conversation into their corresponding semantic frames. Our work suggests a possible future in which the crowd can be used to increasingly automate the process of building dialog systems in new domains, on-demand, with minimal effort from developers.

Background

Our approach draws from rich prior work in both conversational dialog systems, and crowdsourcing.

Conversational Dialog Systems

Our ChatCollect system is designed to alleviate the development efforts for dialog systems. The manual authoring of these systems has been recognized as a time consuming effort that requires domain expertise (Ward and Pellom 1999). This challenging and time-consuming process has been used in the development of many dialog systems (e.g., (Gandhe et al. 2009; Aleven et al. 2006; Allen et al. 1996)). Alternative approaches have focused on learning from transcribed human-human dialogs (Gorin, Riccardi, and Wright 1997) or dialogs collected through Wizard-of-Oz studies (Lathrop 2004). However, existing human-human corpora apply to limited domains, and collecting data for new domains with the existing techniques is difficult and expensive. Moreover, they do not scale well – an important aspect given the amount of data needed to train accurate models for dialog systems. The system proposed in this paper holds promise for obtaining natural dialog data for various domains with little to no involvement from system designers. The approach may enable automated training of future dialog systems and speed up the real-world deployment of such systems.

Crowdsourcing

Crowdsourcing has gained immense popularity in recent years as a means to provide programmatic, easy, and scalable access to human intelligence. Crowdsourcing has been applied to a large variety of tasks ranging from search relevance (Chen et al. 2013) to image description (Bigham et al. 2010). Recently, there has been growing interest in applying crowdsourcing to language problems. Crowdsourcing tasks have been designed for speech transcription (Lasecki et al. 2012), speech acquisition (Lane et al. 2010), translation (Zaidan and Callison-Burch 2011) and paraphrase generation (Burrows, Potthast, and Stein 2013). Some prior work has particularly focused on crowdsourcing methods for the development of dialog systems. Wang et. al. studied methods for eliciting natural language for a given semantic form (Wang et al. 2012). The Asgard system uses crowdsourcing for free-form language generation and for the semantic labeling of segments of the collected language (Liu et al. 2013). Bessho et. al. (Bessho, Harada, and Kuniyoshi 2012) developed a dialog system that requests efforts from a real-time crowd to make progress in a dialog when automated approaches fail.

Continuous real-time crowdsourcing was introduced by Lasecki et. al (Lasecki et al. 2011) in order to create crowd-powered systems capable of interacting with their users and their environment. This approach was applied to conversational interaction in Chorus (Lasecki et al. 2013b; 2013a), a conversational question answering system. Both of these systems are able to quickly get input from crowd workers by pre-recruiting (Bigham et al. 2010; Bernstein et al. 2011). ChatCollect differs from these systems by recruiting all sides of the dialog from the crowd for automated dialog generation.

ChatCollect System

The ChatCollect system is designed to collect realistic task-oriented dialog data from crowd workers. The only inputs of the system developer to the ChatCollect system are the different tasks the developer wants to collect data about. For example, a developer may tell ChatCollect to collect data about flight reservations, planning a night out or deciding what car to buy.

The ChatCollect system hires a pair of workers from a crowdsourcing marketplace in real-time and links them to an instant-messenger style chat window to have a conversation about the given task, and assigns a role to each worker. The first worker routed to the chat window is assigned the role of the "Assistant" and is instructed to help someone with the task at hand, such as finding a flight. The second worker is assigned the role of the "User" and is given a task type to complete with the help of the Assistant (e.g., find flight). In order to add realistic depth to the conversations held by workers playing a part, the system does not prime the User with the details of the task she needs to complete, but instead it asks the User to imagine such a setting. Both workers are instructed not to share any personally identifiable information about themselves.

Since the Assistant is hired first and routed to the chat interface, the User finds the Assistant waiting when she is routed to the chat interface. The User is instructed to start the dialog when she is at the chat interface by telling the Assistant about the task. The User and Assistant both share information about the task, discuss possible solutions, and revise the task.

Workers are asked to complete the task as well as possible (rather than meet the minimum requirements, or just accept any response). Once the User deems that the task is complete, she can signal the end of the interaction by clicking "done", which forwards her and the Assistant worker to a survey page asking them each about the quality of the interaction they had. The survey also asks the Assistant workers about the resources (e.g., websites) they used to complete the given task.

We designed a payment scheme for the ChatCollect system that rewards workers based on the time they spend on the task as well as the quality of dialog they had (assessed by evaluators, or their conversational partner). Workers are paid a fixed amount per minute for waiting for their partner to arrive and for the dialog to start. We then review the dialogs after completion and assign a bonus amount for each dialog to reward the dialogs that successfully complete their task.

Experiments

We evaluated the ChatCollect system on the Universal Human Relevance System (UHRS) crowdsourcing marketplace, Microsoft's internal crowdsourcing platform. Similar to other platforms, UHRS connects a large worker pool from different regions in the world with requesters seeking human input. Previous work on this platform shows that workers provide high quality work with low latency (Wang et al. 2012). In our experiments, we used American English speaking workers.

In order to see what these conversations will look like, we collected an initial dataset that could be manually analyzed. We collected this data in 3 sessions, resulting in 16 completed conversations focusing on two tasks: finding a flight, and finding a hotel. The details for these tasks (e.g., city) were not told to workers, but instead chosen by the worker as part of their role-playing.

Success rate Our 16 successful conversations were drawn from 38 started sessions. While this is only a 42.1% completion rate for paired workers, our manual evaluation shows that every one of the conversations that lasted more than one round were marked as complete by the workers and were evaluated by us as successful. This means that filtering out bad sessions is easy, helping to keep the cost of data collection low. All of the completed conversations contained valid responses to the questions asked.

This analysis suggests that once two workers are paired for the task and start a conversation, they complete a successful dialog. The unsuccessful conversations resulted from workers not being successfully paired in real-time. For example, in a number of the instances, the assistant worker left the system before a user worker could be assigned to the task to start a dialog. The average wait time for a worker in a successful conversation was 4:41 minutes (median 2:16 minutes), while the average wait time for an unsuccessful one was 34:35 minutes (median 6:16 minutes).

Length of Conversations There were a total of 343 turns in the conversations we collected, with an average of 21.4 (median 19) turns containing 268.5 (median 243) words per conversation. The standard deviation was 13.7 turns and 197.5 words, showing the conversations varied significantly in length. In fact, the minimum number of turns was 6, while the maximum was 58. The minimum number of words was 75 (from a conversation with 8 turns) and the maximum was 748. As may be expected, number of turns was a strong predictor of amount of interaction (number of words).

Variation Between Conversations In the 16 conversations between pairs of workers, 19 unique workers participated. Despite the fact that some workers repeated (which we explicitly allowed), none of the conversations repeated which city or pair of cities were involved in the travel plans.

Worker Feedback From the 16 conversations, we collected 25 survey responses: 13 from Users and 12 from Assistants. Of the Users, 11 (84.6%) replied that they were "very satisfied" with the responses from the assistant, one said they were "somewhat satisfied", and one said they were "very dissatisfied", though this worker claimed to be happy with the response in the chat itself.

11 of 13 Users said the interface made their task "very easy" to complete, while two said it was "somewhat easy". 8 out of 12 of the Assistants thought the task was "very easy" to complete, 3 found it "somewhat easy", and one was neutral. Six of these workers reported that the hardest part of the task was finding the right questions to ask to determine the user's preferences. To complete their task, Assistants used typical web tools such as Google, Kayak, or Expedia. Overall, 5 workers specifically mentioned enjoying the task in the free-response section of the survey, supporting prior work using Mechanical Turk that also found a large percentage of workers enjoyed a conversational task (Lasecki et al. 2013b).

Crowd Parsing of Semantic Frames

One potential avenue for using the data collected in this manner is to construct corpora for developing the domain-specific spoken language understanding component of a dialog system. In slot-filling type domains, this amounts to identifying the domain ontology, i.e. the set of domain-specific frames, the corresponding slots and values, and developing a labeled corpus that captures the semantic parses for each lexical turn in the dialog.

We address this problem by developing CrowdParse, a system that uses the crowd to parse dialogs into semantic frames as the conversation progresses. As a result of this task, we can generate dialog data accompanied with semantic labels that can be directly used in training spoken language understanding components (Liu et al. 2013). CrowdParse asks workers to evaluate the information provided in a task-oriented dialog and extract frames (tasks), fields (types of information), and values (information provided) for both sides of the conversation. As the conversation progresses, workers add information, building on the prior information captured by either their previous step or prior workers.

When workers take the CrowdParse task, they are routed to a point in a conversation. They then update the frame they are presented with, and continue to a new task until they choose to stop.

Discussion and Future Work

Our initial results suggest that ChatCollect can provide an effective tool for collecting task-oriented dialog data. We limited the size of the data collected for this exploratory work to 16 in order to more easily review each conversation manually, but ChatCollect can be run continuously to collect larger datasets. In future work, it will be interesting to explore how the dialog data collected with our system compares to prior datasets generated by bringing participants into a lab setting, or collected through a deployed system.

While the completed conversations were all successful, fewer than half of the chat sessions that workers were routed to were completed. The main reason for this was that sometimes a large delay would occur between the arrivals of the two workers. Accordingly, none of the incomplete conversations had more than one round of dialog (each party spoke at most once when they arrived, but the first left before the second arrived). A quick way to address this issue is following marketplace specific strategies to attract workers to our tasks more quickly, such as reposting tasks to increase task visibility. As a more permanent fix, future versions of ChatCollect will detect when one party disconnects, and route the other to an active task (with appropriate compensation for any work done so far). This also helps in the case when a conversation is partially complete, and one worker leaves, as was observed in preliminary testing. Also, pre-recruiting approaches used in prior work can help ensure worker availability (Bigham et al. 2010; Bernstein et al. 2011).

Our ultimate goal is to enable the automatic training and scaling of dialog systems to new domains. Future versions of the system will investigate collecting speech from workers with microphones in order to train spoken language understanding components. We believe such a pipeline can reduce the cost of developing dialog systems that are able to easily generalize to new domains.

Conclusion

Crowdsourcing methods such as the ones presented here offer new opportunities for developing dialog systems that can continuously learn on demand with low cost. In this paper, we introduced ChatCollect, a system for collecting task-oriented dialog data using pairs of crowd workers. We presented some results from an initial set of 16 conversations containing a total of 4296 turns which we analyzed manually, finding that these conversations are appropriately varied and on-topic. We also discussed ongoing work on CrowdParse, a system that uses the crowd to parse semantic frames from dialogs. We believe this work takes first steps towards a future in which dialog systems can be easily trained in new domains by using crowd-generated datasets.

Acknowledgements

The authors would like to thank Eric Horvitz for many insightful discussions about this work.

References

Aleven, V.; Sewall, J.; McLaren, B. M.; and Koedinger, K. R. 2006. Rapid authoring of intelligent tutors for real-world and experimental use. In *ICALT 2006*, 847–851.

Allen, J. F.; Miller, B. W.; Ringger, E. K.; and Sikorski, T. 1996. A robust system for natural spoken dialogue. In *Proceedings of ACL 1996*, 62–70.

Bernstein, M. S.; Brandt, J. R.; Miller, R. C.; and Karger, D. R. 2011. Crowds in two seconds: Enabling realtime crowd-powered interfaces. In *Proc. of UIST 2011*, 33–42.

Bessho, F.; Harada, T.; and Kuniyoshi, Y. 2012. Dialog system using real-time crowdsourcing and twitter large-scale corpus. In *Proceedings of SIGDIAL 2012*, 227–231.

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. 2010. Vizwiz: nearly real-time answers to visual questions. In *Proceedings of UIST 2010*, 333–342.

Burrows, S.; Potthast, M.; and Stein, B. 2013. Paraphrase acquisition via crowdsourcing and machine learning. *ACM Trans. Intell. Syst. Technol.* 4(3):43:1–43:21.

Chen, X.; Bennett, P. N.; Collins-Thompson, K.; and Horvitz, E. 2013. Pairwise ranking aggregation in a crowdsourced setting. In *Proceedings of WSDM 2013*, 193–202.

Gandhe, S.; DeVault, D.; Roque, A.; Martinovski, B.; Artstein, R.; A. Leuski, J. G.; and Traum, D. 2009. From domain specification to virtual humans: An integrated approach to authoring tactical questioning characters. In *Proceedings of Interspeech 2008*.

Gorin, A. L.; Riccardi, G.; and Wright, J. H. 1997. How may i help you? *Speech Commun.* 23(1-2):113–127.

Lane, I.; Waibel, A.; Eck, M.; and Rottmann, K. 2010. Tools for collecting speech corpora via mechanical-turk. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, CSLDAMT '10, 184–187.

Lasecki, W.; Murray, K.; White, S.; Miller, R. C.; and Bigham, J. P. 2011. Real-time crowd control of existing interfaces. In *Proceedings of UIST 2011*, 23–32.

Lasecki, W.; Miller, C.; Sadilek, A.; AbuMoussa, A.; and Bigham, J. 2012. Real-time captioning by groups of non-experts. In *Proceedings of UIST 2012*.

Lasecki, W. S.; Thiha, P.; Zhong, Y.; Brady, E.; and Bigham, J. P. 2013a. Answering visual questions with conversational crowd assistants. In *Proc. of ASSETS 2013*, To Appear.

Lasecki, W.; Wesley, R.; Nichols, J.; Kulkarni, A.; Allen, J.; and Bigham, J. 2013b. Chorus: A crowd-powered conversational assistant. In *Proceedings of UIST 2013*, To Appear.

Lathrop, B. e. a. 2004. A wizard of oz framework for collecting spoken human-computer dialogs: An experiment procedure for the design and testing of natural language in-vehicle technology systems. In *Proceedings of ITS 2004*.

Liu, J.; Pasupat, P.; Cyphers, S.; and Glass, J. 2013. Asgard: A portable architecture for multilingual dialogue systems. In *Proceedings of ICASSP 2013*.

Wang, W. Y.; Bohus, D.; Kamar, E.; and Horvitz, E. 2012. Crowdsourcing the acquisition of natural language corpora: Methods and observations. In *SLT 2012*. IEEE.

Ward, W., and Pellom, B. 1999. The cu communicator system. In *Proceedings of IEEE ASRU*.

Zaidan, O. F., and Callison-Burch, C. 2011. Crowdsourcing translation: professional quality from non-professionals. In *Proceedings of HLT 2011*, 1220–1229.