Real-time On-Demand Crowd-powered Entity Extraction (Supplementary Technical Report)

Ting-Hao (Kenneth) Huang
Carnegie Mellon University
Pittsburgh, PA, USA
tinghaoh@cs.cmu.edu

Yun-Nung (Vivian) Chen
National Taiwan University
Taipei, Taiwan
yvchen@csie.ntu.edu.tw

Jeffrey P. Bigham
Carnegie Mellon University
Pittsburgh, PA, USA
jbigham@cs.cmu.edu

[Note] This is the supplementary technical report of our paper "Real-time On-Demand Crowd-powered Entity Extraction," which was published at the 5th Edition of The Collective Intelligence Conference (CI 2017) as an oral presentation (Huang et al. 2017). The original paper was only 3-page long, so we decided to share extra technical details in this report.

Abstract

Modern dialog systems rely on accurate entity extraction to understand user utterances. However, entity extraction is brittle due to data scarcity, language variability, and out-of-vocabulary entities. To bridge this gap, we propose a real-time crowdsourcing solution based on the ESP game for image labeling. When multiple players agree, entities can be reliably extracted from an utterance. This approach is advantageous because it does not require training data. Further, it is robust to unexpected input and capable of recognizing new entities. Our approach achieves better F1-scores than that of the automated baseline for complex queries with a reasonable response time. The proposed method is also evaluated via Google Hangouts' text chat and demonstrates the feasibility of real-time crowd-powered entity extraction.

Introduction

To understand user utterances, modern dialog systems rely heavily on entity extraction, known as the core task of *slot filling* in many dialog system frameworks such as Olympus (Bohus et al. 2007). The goal of slot filling is to identify from a running dialog different *slots*, which correspond to different parameters of the user's query. For instance, when a user queries for nearby restaurants, key slots for *location* and *preferred food* are required for a dialog system to retrieve the appropriate information. Thus, the main challenge in the slot-filling task is to extract the target *entity*.

Dialog systems face three key challenges in entity extraction. Due to **data scarcity**, labeled training data, which many existing technologies require to identify entities such as Conditional Random Fields (CRF) (Raymond and Riccardi 2007; Xu and Sarikaya 2014) and Recurrent Neural Networks (Mesnil et al. 2015a), are often unavailable for the wide variety of dialog system tasks. Furthermore, it is more difficult to acquire the complicated conversational data required by other alternative dialog technologies, such as statistical dialog management (Young 2006) or state tracking (Williams et al. 2013). Second, existing entity extraction technologies are not robust enough to identify **out-of-vocabulary entities**. Even when labeled training data for the targeted slot could be collected, state-of-the-art supervised learning approaches are brittle in extracting unseen entities. (Xu and Sarikaya 2014) find that the CRF-based entity extractor performed significantly worse when dictionary features were not used. Third, challenges are also posed by **language variability**. Successful applications process diverse input languages where potential entities are unlimited. Therefore, to robustly serve arbitrary input, dialog systems must collect new sources of entities and update accordingly.

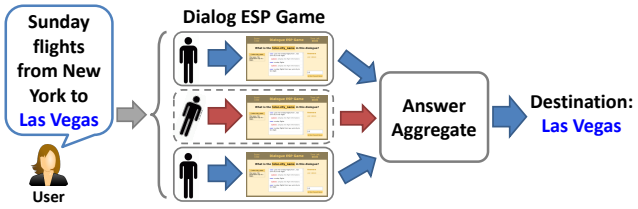

Figure 1: The crowd-powered entity extraction with a multiplayer Dialog ESP Game. By aggregating input answers from all players, our approach is able to provide good quality results in seconds.

Research on dialog systems has focused on utilizing the Internet resource to extract entities such as movie names (Wang, Heck, and Hakkani-Tur 2014); Unsupervised slot-filling approaches have also been developed in recent years (Chen, Hakkani-Tür, and Tur 2014; Heck, Hakkani-Tür, and Tür 2013). However, these methods are still underdeveloped.

To address these challenges, we propose to use real-time crowdsourcing as an entity extractor in dialog systems. To the best of our knowledge, few previous works have attempted to use crowdsourcing to extract entities from a running conversation. (Wang et al. 2012), for example, studied various methods to acquire natural language sentences for a given semantic form by the crowd. (Lasecki, Kamar, and Bohus 2013) utilized crowdsourcing to collect dialog data, and

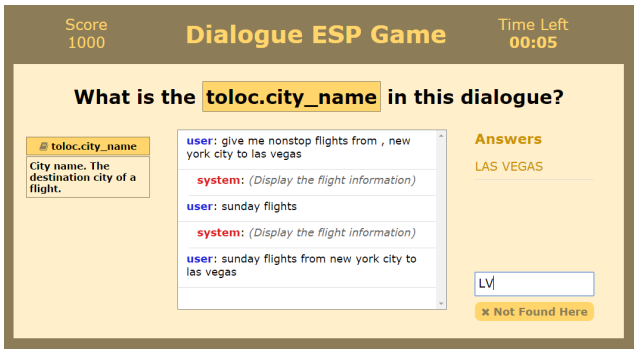

Figure 2: The Dialog ESP Game interface is designed to encourage quick and correct entity identification by crowd workers. Workers are shown the complete dialog and a description of the entity they should identify.

illustrated CrowdParse, a system that uses the crowd to parse dialogs into semantic frames. Recently, (Huang, Lasecki, and Bigham 2015) presented a crowd-powered dialog system called Guardian that uses the crowd to extract information from input utterances. However, none of these works conducted formal studies on crowd-powered entity extraction in real-time.

Inspired by the ESP game for image labeling (Von Ahn and Dabbish 2004), we propose a **Dialog ESP Game** to encourage crowd workers to accurately and quickly perform entity extraction. The ESP Game matches answers among different workers to ensure label quality, and we use a timer on the interface (Figure 2) to ensure input speed. Our method offers three main advantages: 1) it does not require training data; 2) it is robust to unexpected input; and 3) it is capable of recognizing new entities. Furthermore, answers submitted from the crowd can be used as training data to bootstrap automatic entity extraction algorithms. In this paper, we conduct experiments on a standard dialog dataset and user experiments with 10 users via Google Hangouts' text chatting interface. Detailed experiments demonstrate that our crowd-powered approach is robust, effective, and fast.

In sum, the contributions of our work are as follows:

1. We propose an ESP-game-based real-time crowdsourcing approach for entity extraction in dialog systems, which enables accurate entity extraction for a wide variety of tasks.
2. To strive for real-time dialog systems, we present detailed experiments to understand the trade-offs between entity extraction accuracy and time delay.
3. We demonstrate the feasibility of real-time crowd-powered entity extraction in instant messaging applications.

Real-time Dialog ESP Game

We utilize real-time crowdsourcing with a multi-player Dialog ESP Game setting to extract the targeted entity from a dialog[1]. The ESP Game was originally proposed as a crowdsourcing mechanism to acquire quality image labels (Von Ahn and Dabbish 2004). The original game randomly pairs two players and presents them with the same image. Each player guesses the labels that the other player would answer. If the players match labels, each is awarded 1000 points. Our approach replaces the image in the ESP Game with a dialog chat log and players answer the required entity name within a short time. We also relax the constraints of player numbers to increase game speed. As Figure 1 shows, by aggregating input answers from all players, the Dialog ESP Game is able to provide high quality results in seconds.

Figure 2 shows the worker's interface. When input dialog utterances reach the crowd-powered entity extraction component, workers are recruited from crowdsourcing platforms such as Amazon Mechanical Turk (MTurk). The timer begins counting down when the input utterance arrives, and the worker sees the remaining time on the top right corner of the interface (Figure 2). When two workers match answers, a feedback notification is displayed, and the workers earn 1000 points. When the time is up, the task automatically closes.

To recruit crowd workers quickly, many approaches have been used in real-time crowd-powered systems such as VizWiz (Bigham et al. 2010) and Chorus (Lasecki et al. 2013). The *quikTurkit* toolkit (quikturkit.googlecode.com) attracts workers by posting tasks and using old tasks to queue workers. Similarly, the Retainer Model maintains a retaining pool of workers-in-waiting, who receive a signal when tasks become available. Prior research shows that the Retainer Model is able to recall 75% of workers within 3 seconds (Bernstein et al. 2011). In Experiment 1, we first focus on the speed and performance of the Dialog ESP Game itself instead of recruiting time. In Experiment 2, we propose a novel approach to recruit workers within 60 seconds and discuss details of the end-to-end response speed.

Experiment 1: Applying Dialog ESP Game on ATIS Dataset

To evaluate the Dialog ESP Game for entity extraction, we conducted experiments on MTurk to extract names of destination cities from a flight schedule query dialog dataset, the Airline Travel Information System (ATIS) dataset.

ATIS Dataset The ATIS dataset contains a set of flight schedule query sessions, each of which consists of a sequence of spoken queries (utterances). Each query contains automatic speech recognized transcripts and a set of corresponding SQL queries. All queries in the data set are annotated with the query category: A, D, or X. Class A queries are context-independent, answerable, and formed mostly in a single sentence; however, real-world queries are more

[1]The source code of worker interface and the data collected in Experiment 2 are available at:
https://github.com/windx0303/dialogue-esp-game

complex. In the ATIS data set, 32.2% queries are context-dependent (Class D) and 24.0% of the queries are cannot be evaluated (Class X) (Hirschman 1992). The "context-dependent" Class D queries require information from previous queries to form a complete SQL query. For instance, in one ATIS session, the first query is "From Montreal to Las Vegas" (Class A). The second query in the session is "Saturday," which requires the destination and departure city name from the first query, and is thus annotated as Class D. Class X is of all the problematic queries, e.g., hopelessly-vague or unanswerable.

Data Pre-processing & Experiment Setting For Class A, we obtain the preprocessed data used in many slot filling works (Xu and Sarikaya 2014; Mesnil et al. 2015a; He and Young 2003; Raymond and Riccardi 2007; Tur, Hakkani-Tur, and Heck 2010), which contain 4,978 queries for training, 893 queries for testing, and 491 queries for developing. 200 queries are randomly extracted from the developing set for our study; For Class D and X, we obtain the original training set of ATIS-3 data (Dahl et al. 1994), which contains 364 sessions and 3,235 queries. 200 Class-D queries are randomly selected from 200 distinct sessions. For each extracted query, all previous queries before it within the same session are also obtained and displayed in the worker's interface (Figure 2). The same process is used to extract 150 Class-X queries for the experiments. Note that in this work we focus only on the `toloc.city_name` slot (name of destination city), which is the most frequent slot type in ATIS. For each extracted query of Class D and X, we define the last-mentioned destination city name of the flight in the query history (including the extracted query) as the gold-standard slot value.

Understanding Accuracy and Speed Trade-offs

In order to design an effective crowd-powered real-time entity extraction system, it is crucial to understand trade-offs between accuracy and speed. These trade-offs correspond to the three main variables in our system: the **number of players** recruited to answer each query in the Dialog ESP Game, the **time constraint** that each player has to answer a query, and the **method to aggregate input answers**. We have 3 ways to aggregate the input answers from the ESP game:

- **ESP Only**: Return the first matched answer. If no answers match within the given time, return an empty label.
- i**th Only**: Return the ith input answer (i = 1, 2, ...). For example, i = 1 means to return the first input answer.
- **ESP +** i**th**: Return the first matched answers of the ESP game. If no answers match within the given time, return the ith answer.

We recruit 10 players for each ESP game, and randomly select player results to simulate the conditions of various player numbers. All results reported in Experiment 1 are the averages of 20 rounds of this random-pick simulation process. After empirically testing the interface, we run two sets of studies with time constraints set at 20 and 15 seconds, respectively. Different methods to aggregate input answers could result in different response speed and output quality. Note that if there are not any input answers, the methods above will wait until the time constraint and return an empty label. In the actual experiments, 5 Dialog ESP Games for 5 different Class-A queries are aggregated in one task, with an extra scripted game at the beginning as a tutorial. When the first game ends, the timer of the second ESP game starts and a browser alert informs the worker. All experiments are run on MTurk; 800 Human Intelligence Tasks (HITs) are posted, and 588 unique workers participate in this study.

Time Const.	Aggregate	# Player	Avg. Resp. Time	P	R	F1
20s	ESP+ 1st	10	7.837s	.867	.916	**.891**
		5	11.160s	.828	.877	.852
	1st Only	10	**5.590s**	.713	.753	.732
		5	6.924s	.730	.769	.749
	ESP Only	10	7.837s	.867	.916	**.891**
		5	11.160s	.856	.797	.826
15s	ESP+ 1st	10	8.129s	.837	.893	.864
		5	10.628s	.799	.798	.798
	1st Only	10	5.895s	.739	.764	.751
		5	7.136s	.729	.726	.727
	ESP Only	10	8.129s	.860	.865	.863
		5	10.628s	.872	.637	.736

Table 1: Dialog ESP Game results in Class A given different settings of number of players, time constraint (Time Const.), and the method to aggregate input answers.

Table 1 shows the results on Class A queries. With 10 players and a 20-second time constraint, the Dialog ESP Game achieves a best F1-score of 0.891 by the "ESP+1st" setting, and achieves the fastest average response time of 5.590 seconds by the "1st" setting. The **ESP+1st** setting achieves the best F1-score, and the 1**st Only** setting has the shortest response time. In most cases, tightening the time constraint provides a faster response but reduces output quality.

We also analyze the relations among worker numbers, performance, and response time. First, Figure 4 shows output quality with respect to answer's input order. On average, earlier input answers are of better quality, unless 10 or more players participate in the game. However, with 10 players, almost all ESP games have at least one matched answer pair so that the ith answer is not solely used. Therefore, for the following experiments, we set i as 1. Second, in Figure 3(a) we observe the relations between the number of players and average response time. Adding players reduces the average response time for all settings. Third, the relations between number of players and output quality are also analyzed. Figure 3(b) shows that the F1-scores increase when adding more players, even with the "1st Only" setting.

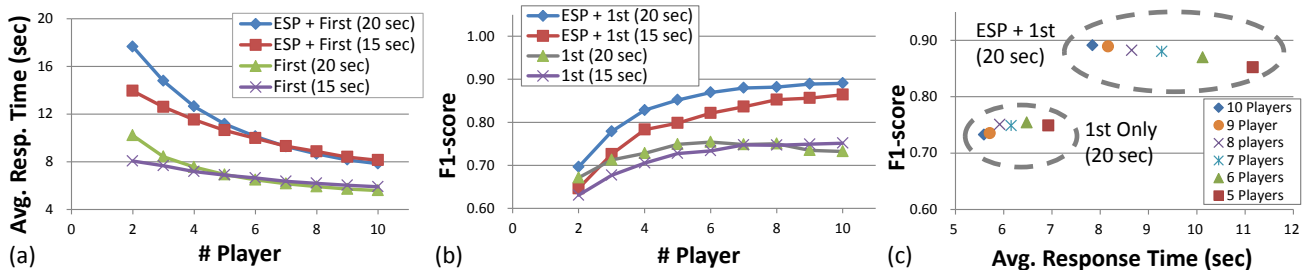

Figure 3: Trade-off curves between accuracy, average response time and number of players.

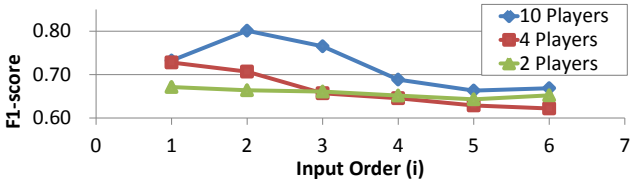

Figure 4: F1-score of the "ith Only" setting. Earlier input answers are generally of better quality (unless #players $\geqslant$ 10, where almost all ESP games have at least one matched answer and the ith answer might not be solely used.)

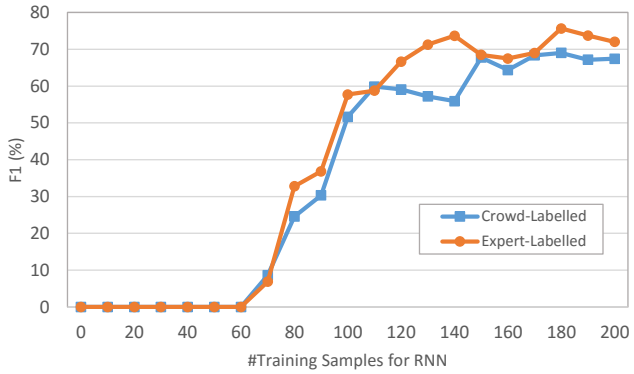

Figure 5: The performance of the extracted destination city predicted by the RNN-GRU trained on crowd-labelled and expert-labelled data. Crowd-labelled labels can be used to train machine-learning models and achieve comparable performance to expert-labelled data.

Finally, Figure 3(c) demonstrates the trade-offs between performance and speed. For a fixed number of players, different input aggregate methods have different response times and F1-scores. The ESP game requires more time for input answer matching, but in return output quality increases.

Bootstrapping Automated System Performance

Once our crowd-powered system starts extracting entities, the collected annotations can serve as training data for the automated system. In order to see whether the crowd-annotated data is good enough for training a machine learning model, and how many of text instances are required to achieve reasonable performance, we train a state-of-the-art language understanding model, implemented by an recurrent neural network (RNN) with GRU cells (Mesnil et al. 2015b; Chen et al. 2016), on the crowd-labelled or expert-labelled 200 ATIS conversations used in our study, and use the standard ATIS Class-A testing set to compare the performance in terms of the number of training samples. Figure 5 illustrates the performance curves of models trained on crowd-labelled and expert-labelled data. The result shows that crowd-generated labels can be used to train machine-learning models for bootstrapping in new domains, and achieve comparable performance to expert-labelled data.

Evaluation on Complex Queries

Based on the study above, for Class D and X queries, we use the Dialog ESP Game of 10 players with "ESP+1st" and "1st Only" settings to measure the best F1-score and speed. The time constraint is set to 20 seconds. The experiments are run on MTurk and all settings are identical as the previous section. 76 distinct workers participate in Class D experiments, and 68 distinct workers participate in Class X experiments.

Experimental results are shown in Table 2. An automated CRF model is implemented as a baseline.[2] The CRF model is trained on the Class-A training set mentioned above by using neighbor words (window size = 2) and POS tag features. The CRF model is decoded and timed on a laptop with Intel i5-4200U CPU (@1.60GHz) and 8GB RAM. As a result, the proposed crowd-powered approach largely outperforms the CRF baseline in terms of F1-score on both Class D and X queries . Although the CRF approach is well-developed on Class A data, it is not effective on the remaining data.

Surprisingly, we find similar average response times in each query category. Note that the text length is different for each category: the average token number of Class-A queries is 11.47, of Class-D queries (including the query history) is 48.64, and of Class-X queries is 67.72. Studies showed that eyes' warm-up time (Inhoff and Rayner 1986) and word frequency influence speed of text comprehension (Rayner and Duffy 1986; Healy 1976). These factors might reduce the effect of text length to the reading speed of crowd works.

We also conduct an error analysis on the result of "ESP+1st" setting, which achieves our best F1-score. The distribution of error types are shown in Table 3. The "`fromloc.city_name`" type indicates that the crowd extracts the departure city, rather than destination city; In "In-

[2]Implemented with CRF++: http://taku910.github.io/crfpp/

Query Category	Class D (context-dependent)				Class X (unevaluable)				Class A (context-independent)			
Methods	Resp. Time	P	R	F1	Resp. Time	P	R	F1	Resp. Time	P	R	F1
CRF Baseline	0.043s	.776	.307	.440	0.061s	.636	.285	.393	0.019s	.985	.987	**.986**
1st Only	5.460s	.658	.641	.649	6.342s	.563	.577	.570	5.590s	.713	.753	.732
ESP + 1st	7.118s	.814	.797	**.805**	8.301s	.654	.675	**.664**	7.837s	.867	.916	.891

Table 2: Result for Class D, X and A. Crowd-powered entity extraction outperforms the CRF baseline in terms of F1-score on both Class D and X queries. Although the CRF baseline is well-developed on Class A, it is not effective on complex queries.

Error Type	Class D	Class X	Class A
`fromloc.city_name`	39.53%	16.67%	40.00%
False Negative	18.60%	26.67%	0.00%
Incorrect City	16.28%	18.33%	8.00%
Correct City & Soft Match	16.28%	5.00%	12.00%
False Positive	9.30%	33.33%	40.00%

Table 3: Error Analysis for Class D, X and A.

correct City" type, the crowd extracts an incorrect city from the query history (but not the departure city); "Correct City & Soft Match" type means the extracted city name is semantically correct but does not match the gold-standard city name (e.g., "Washington" and "Washington DC"). From the error analysis, we conclude two directions to improve performance: 1) treat the cases of absent slot more carefully, and 2) use domain knowledge if available. First, 28% of errors in Class D and 50% in Class X occur when either the gold-standard label or the predicted label does not exist. It suggests that a more reliable step to recognize the existence of the targeted entity might be required. Second, 16.28% of Class-D queries and 5% of Class-X queries are of the "Soft Match" cases. By introducing domain knowledge like a list of city names, a post-processor that finds the most similar city name of the predicted label can fix this type of error.

Experiments 2: User Experiment via a Real-world Instant Messaging Interface

To examine the feasibility of real-time crowd-powered entity extraction in an actual system, we conduct lab-based user experiments via Google Hangouts' instant messaging interface. Our proposed method has a task completion time of 5-8 seconds, per Experiment 1. In this section, we demonstrate our approach is robust and fast enough to support a real-world instant messaging application, where the average time gap between conversational turns is 24 seconds (Isaacs et al. 2002).

System Implementation

We implemented a Google Hangouts chatbot by using the Hangupsbot[3] framework. Users are able to send text chats to our chatbot via Google Hangouts. The chatbot recruits crowd workers on MTurk in real-time to perform the Dialog ESP Game task upon receiving the chat. Figure 6 shows the overview of our system. We record all answers submitted by recruited workers and log the timestamps of following activities: 1) users' and workers' keyboard typing, 2) workers' task arrival, and 3) the workers' answer submissions.

To recruit crowd workers, we introduce *fleeting task*, a recruiting practice inspired by *quikturkit* (Bigham et al. 2010). This approach achieves low latency by posting hundreds of short lifetime tasks, which increases task visibility. Its short lifetime (e.g., 60 seconds) encourages workers to complete tasks quickly. A core benefit of the *fleeting task* approach is its ease in implementation: the method bypasses the common practices of pre-recruiting workers and maintaining a waiting pool (Bigham et al. 2010; Lasecki et al. 2013; Bernstein et al. 2011). In a system deployed at scale, a retainer or push model is likely to work as well.

User Experiment Setup

We conduct lab-based user experiments to evaluate the proposed technology on extracting "food" entities. Ten Google Hangouts users enter our lab with their own laptops. We first ask them to arbitrarily create a list 9 foods, 3 drinks, and 3 countries based on their own preferences. Then we explain the purpose of the experiments, and introduce five scenarios of using instant messaging:

1. **Eat**: You discuss with your friend about what to eat later.
2. **Drink**: You discuss with an employee a coffee place, bar, or restaurant to order something to drink.
3. **Cook**: You plan to cook later. You discuss the details with your friend who knows how to cook.
4. **Chat**: You are chatting with your friend.
5. **No Food**: You are chatting with your friend. You do not mention food. Instead, you mention a country name.

We also list three types of conversational acts which could emerge in each scenario:

1. **Question**: Ask a question.
2. **Answer**: Answer a question that could be asked under the current scenario.
3. **Mentioning**: Naturally converse without asking or answering any specific questions.

Using their laptops, users send one text chat for each combination of [scenario, conversational act] to our chatbot, i.e., 15 chats in total. In the Eat, Cook, and Chat scenarios, users must mention one of the foods they listed earlier; in the Drink scenario, they must mention one of the drinks they listed. In the No Food scenario, users must mention one of

[3]https://github.com/hangoutsbot/hangoutsbot

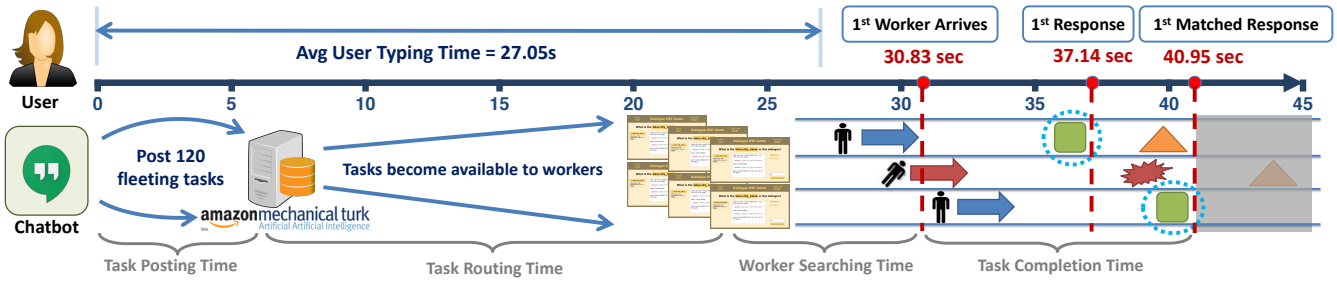

Figure 6: Timeline of the Real-time Crowd-powered Entity Extraction System. On average, the first worker takes 30.83 seconds to reach, the first answer is received at 37.14 seconds, and the first matched answer occurs at 40.95 seconds. A user on average spends 27.05 seconds to type a chat line, i.e., the perceived response time to users falls within 10-14 seconds.

	Acc (%)	Response Time (s) Mean (Stdev)
1st Only	77.33%	37.14 (14.70)
ESP Only	81.33%	40.95 (13.56)
ESP + 1st	84.00%	40.95 (13.56)
1st Worker Reached Time (s)		30.83 (16.86)
User Type Time (s)		27.05 (25.28)

Table 4: Result of User Experiment. A trade-off between time and output quality can be observed.

		1st		ESP + 1st	
		Avg. Time(s)	Acc. (%)	Avg. Time(s)	Acc. (%)
Entity Type	Food[5]	36.64	70.00%	40.19	**78.89%**
	Drink	37.43	80.00%	41.37	**83.33%**
	None	38.33	96.67%	42.83	**100.00%**
Conv. Act	Question	34.26	82.00%	37.94	**90.00%**
	Answer	39.90	68.00%	43.88	**78.00%**
	Mention	37.26	82.00%	41.04	**84.00%**
Avg.		**37.14**	**77.33%**	**40.95**	**84.00%**

Table 5: Results of user experiment for each scenario and conversational act.

the countries they listed, and no food names can be mentioned. In total, we collect 150 chat inputs from 10 user experiments. Correspondingly, instructions on the workers' interface (Figure 2) is modified as "What is the `food_name` in this dialog?", and the explanation of `food_name` is modified as "*Food name. The full name of the food. Including any drinks or beverages.*" In the experiments, our chatbot post 120 HITs with a lifetime of 60 seconds to MTurk upon receiving a text chat. The price of each HIT is $0.1. We use the interface shown in Figure 2 with a time constraint of 20 seconds.

Experimental Results

Results are shown in Table 4. The "ESP+1st" setting achieves the best accuracy of 84% with an average response time of 40.95 seconds. The "1st Only" setting has the shortest average response time of 37.14 seconds with an accuracy of 77.33%.[4] A trade-off between time and output quality can be observed.This trade-off is similar to the results of Experiment 1 (shown in Figure 3(c)). On average, 14.45 MTurk workers participated in each trial and submitted 33.81 answers.

Robustness in Out-of-Vocabulary Entities & Language Variability The results over each entity type are shown in Table 5. Without using any training data or pre-defined knowledge-base, our crowdsourcing approach achieves an accuracy of 78.89% in extracting food entities and 83.33% in extracting drink entities. Despite the significant variety of the input entities[6], our approach extracts most entities correctly. Furthermore, our method is effective in identifying the absence of entities; Table 5 also shows the robustness of the proposed method under various linguistic conditions. The "ESP+1st" setting achieves accuracies of 90.00% in extracting entities from questions, 78.00% in extracting from answers, and 84.00% in extracting from regular conversations. Qualitatively, our approach can handle complex input, such as strange restaurant names and beverage names, which are essentially confusing for automated approaches. For example, "Have you ever tried bibimbap at *Green pepper*?" and "I usually have *Magic Hat #9*", where *Green pepper* and *Magic Hat #9* are names of a restaurant and beverage, respectively.

[4] We only consider the answers submitted within 60 seconds.

[5] Including the results from Food, Cook, and Chat scenarios.

[6] The food entities arbitrarily created by our users are quite diverse: From a generic category (e.g., Thai food) to a specific entry (e.g., Magic Hat #9), and from a simple food (e.g., cherry) to a complex food (e.g., sausage muffin with egg). The list covers the food of many other countries (e.g., Okonomiyaki, Bibimbap, Samosa.)

Error Analysis Table 6 shows the errors in the user experiments ("ESP+1st" setting). 45.83% of errors are caused by absence of answers, mainly due to the task routing latency of the MTurk platform. We discuss this in more detail below. 37.50% of errors are due to various system problems such as the string encoding issues. More interestingly, 12.50% of incorrect answers are sub-spans of the correct answers. For instance, the crowd extracts "rice" for "stew pork over rice", and "tea" for "bubble tea". This type of error is similar to the "Soft Match" error in Experiment 1. Finally, 4.17% of errors are caused by user typos (e.g., *latter* for *latte*), which the crowd tends to exclude in their answers.

Error Type	%
No Answers Received	45.83%
System Problem	37.50%
Substring of a Multi-token Entity	12.50%
Typo	4.17%

Table 6: Error Analysis for User Experiment.

Response Speed Table 4 shows the average response time in the user experiment. On average, the first worker takes 30.83 seconds to reach to our Dialog ESP Game, the first answer is received at 37.14 seconds, and the first matched answer occurs at 40.95 seconds. For comparison, we illustrate the timeline of our system in Figure 6. In the user experiments, a user on average spends 27.05 seconds to type a chat line. If we align the user typing time along with the system timeline, the theoretical perceived response time to users falls within 10-14 seconds, while the average response time in instant messaging is 24 seconds (Isaacs et al. 2002). (Baron 2010) reports that 24.5% of instant messages get responses within 11-30 seconds, and 8.2% of messages have even longer response times. The proposed technology proves to be fast enough to support instant messaging applications. The main bottleneck of the end-to-end response speed is the *task routing time* in Figure 6, which approximately ranges from 5-40 seconds and changes over time. The task routing time also causes the major errors in Table 6. The task lifetime begins when a task reaches the MTurk server instead of when it becomes visible to workers. When the task routing time is longer than a task's lifetime, the task could expire before it is selected by workers. Because MTurk requesters can not effectively reduce the task routing time, pre-recruiting and queuing workers seems inevitable for applications which require a response time sharply shorter than 30 seconds.

Discussion

Incorporating domain-specific knowledge is a major obstacle in generalization of crowdsourcing technologies (Huang, Lasecki, and Bigham 2015). We think that automation helps resolve this challenge. One most common errors in our system are the *soft match*, where the crowd extracts a sub-string of the target entity instead of the complete string. Domain knowledge can help to fix this type of errors. However, unlike automated technology, we do not have a generic method to update human workers with new knowledge. Thus, our next step is to incorporate automated components. It is easy to replace some workers with automated annotators in our multi-player ESP Game. Despite fragility in extracting unseen entities, automated approaches are robust in identifying known entities and can be easily updated if new data is collected. We will develop a hybrid approach, which we believe will be robust in unexpected input and easily incorporate new knowledge.

Conclusion and Future Work

We have explored using real-time crowdsourcing to extract entities for dialog systems. By using an ESP Game setting, our approach is absolute 36.5% and 27.1% better than the CRF baseline in terms of F1-score for Class D and X queries in the ATIS dataset, respectively. The timing cost is about 8 seconds, which is slower than machines but still reasonable given the large gains in accuracy. The proposed method also has been evaluated via Google Hangouts' text chat with 10 users. The results demonstrate the robustness and feasibility of our approach in real-world systems. In the future, we will generalize our approach by adding automated components, and also explore the possibility of using audio input.

Appendix: List of Food and Drinks Used in the Experiment 2

The followings are the lists of 9 food and 3 drinks created by 10 participants in our user experiment.

Food

1. spaghetti, burger, vindaloo lamb, makhani chicken, kimchee, wheat bread pizza, cornish pasty, mushroom soup
2. burger, french fries, scallion cake, okonomiyaki, oyakodon, gyudon, fried rice, wings, salad
3. Stinky Tofu, Acai Berry Bowl, Tuna Onigiri, Rice Burger, Seared Salmon, Milkfish Soup, Mapo Tofu, Beef Pho, Scallion Pancake
4. pizza, fried rice, waffle, alcohol drink, chocolate pie, cookie, dimsum, burger, milk shake
5. Pho, BBQ, Thai food, beef noodles, steak, Tomato soup, Spicy hot pot, Soup dumplings, Ramen
6. chocolate, donut, cheesecake, pad thai, seafood pancake, fish fillets in hot chili, hot pot, bibimbap, japchae
7. chocolate, pancakes, strawberries, fried fish, fried chicken, sausages, gulaab jamun, paneer tika, samosa
8. Dumplings, noodle, stew pork over rice, Sandwich, pasta, hot pot, Potato slices with green peppers, Chinese BBQ, pancakes
9. stinky tofu, stew pork over rice, yakitori, baked cinnamon apple, apple pie, stew pork with potato and apple, teppanyaki, okonomiyaki, crab hotpot
10. hot pot, cherry, Chinese cabbage, Pumpkin risotto, Tomato risotto, Boeuf Bourguignon, stinky tofu, sausage muffin with egg (McDonald), eggplant with basil

Drink

1. tea, coke, latte
2. green tea latte, bubble tea, root beer
3. medium latte with non-fat milk, green Tea Latte, Soymilk
4. water, pepsi, tea
5. Latte with nonfat milk, Magic hat #9, Old fashion
6. vanilla latte, strawberry smoothie, iced tea
7. coffee, milk shake, beer
8. Mocha coffee, beers, orange juice
9. caramel frappuccino, caramel macchiato, coffee with coconut milk
10. ice tea, macha, apple juice

References

[Baron 2010] Baron, N. S. 2010. Discourse structures in instant messaging: The case of utterance breaks. *Language@ Internet* 7(4):1–32.

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

[Bigham et al. 2010] Bigham, J. P.; Jayant, C.; Ji, H.; Little, G.; Miller, A.; Miller, R. C.; Miller, R.; Tatarowicz, A.; White, B.; White, S.; et al. 2010. Vizwiz: nearly real-time answers to visual questions. In *UIST*, 333–342. ACM.

[Bohus et al. 2007] Bohus, D.; Raux, A.; Harris, T. K.; Eskenazi, M.; and Rudnicky, A. I. 2007. Olympus: an open-source framework for conversational spoken language interface research. In *Proceedings of the workshop on bridging the gap: Academic and industrial research in dialog technologies*, 32–39. Association for Computational Linguistics.

[Chen et al. 2016] Chen, Y.-N.; Hakkani-Tur, D.; Tur, G.; Celikyilmaz, A.; Gao, J.; and Deng, L. 2016. Knowledge as a teacher: Knowledge-guided structural attention networks. *arXiv preprint arXiv:1609.03286*.

[Chen, Hakkani-Tür, and Tur 2014] Chen, Y.-N.; Hakkani-Tür, D.; and Tur, G. 2014. Deriving local relational surface forms from dependency-based entity embeddings for unsupervised spoken language understanding. *Proceedings of SLT*.

[Dahl et al. 1994] Dahl, D. A.; Bates, M.; Brown, M.; Fisher, W.; Hunicke-Smith, K.; Pallett, D.; Pao, C.; Rudnicky, A.; and Shriberg, E. 1994. Expanding the scope of the atis task: The atis-3 corpus. In *HLT*, 43–48. Association for Computational Linguistics.

[He and Young 2003] He, Y., and Young, S. 2003. A data-driven spoken language understanding system. In *ASRU'03*, 583–588. IEEE.

[Healy 1976] Healy, A. F. 1976. Detection errors on the word the: Evidence for reading units larger than letters. *Journal of Experimental Psychology: Human Perception and Performance* 2(2):235.

[Heck, Hakkani-Tür, and Tür 2013] Heck, L. P.; Hakkani-Tür, D.; and Tür, G. 2013. Leveraging knowledge graphs for web-scale unsupervised semantic parsing. In *INTERSPEECH*, 1594–1598.

[Hirschman 1992] Hirschman, L. 1992. Multi-site data collection for a spoken language corpus. In *Proceedings of the Workshop on Speech and Natural Language*, HLT '91, 7–14. Stroudsburg, PA, USA: Association for Computational Linguistics.

[Huang et al. 2017] Huang, T.-H. K.; Chen, Y.-N.; Bigham, J. P.; et al. 2017. Real-time on-demand crowd-powered entity extraction. In *In Proceedings of the 5th Edition Of The Collective Intelligence Conference (CI 2017, oral presentation)*.

[Huang, Lasecki, and Bigham 2015] Huang, T.-H. K.; Lasecki, W. S.; and Bigham, J. P. 2015. Guardian: A crowd-powered spoken dialog system for web apis. In *HCOMP*.

[Inhoff and Rayner 1986] Inhoff, A. W., and Rayner, K. 1986. Parafoveal word processing during eye fixations in reading: Effects of word frequency. *Perception & Psychophysics* 40(6):431–439.

[Isaacs et al. 2002] Isaacs, E.; Walendowski, A.; Whittaker, S.; Schiano, D. J.; and Kamm, C. 2002. The character, functions, and styles of instant messaging in the workplace. In *Proceedings of the 2002 ACM CSCW*, 11–20. ACM.

[Lasecki et al. 2013] Lasecki, W. S.; Wesley, R.; Nichols, J.; Kulkarni, A.; Allen, J. F.; and Bigham, J. P. 2013. Chorus: A crowd-powered conversational assistant. In *UIST '13*, UIST '13, 151–162. New York, NY, USA: ACM.

[Lasecki, Kamar, and Bohus 2013] Lasecki, W. S.; Kamar, E.; and Bohus, D. 2013. Conversations in the crowd: Collecting data for task-oriented dialog learning. In *HCOMP*.

[Mesnil et al. 2015a] Mesnil, G.; Dauphin, Y.; Yao, K.; Bengio, Y.; Deng, L.; Hakkani-Tur, D.; He, X.; Heck, L.; Tur, G.; Yu, D.; et al. 2015a. Using recurrent neural networks for slot filling in spoken language understanding. *Audio, Speech, and Language Processing, IEEE/ACM Transactions on* 23(3):530–539.

[Mesnil et al. 2015b] Mesnil, G.; Dauphin, Y.; Yao, K.; Bengio, Y.; Deng, L.; Hakkani-Tur, D.; He, X.; Heck, L.; Tur, G.; Yu, D.; et al. 2015b. Using recurrent neural networks for slot filling in spoken language understanding. *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 23(3):530–539.

[Raymond and Riccardi 2007] Raymond, C., and Riccardi, G. 2007. Generative and discriminative algorithms for spoken language understanding. In *INTERSPEECH*, 1605–1608.

[Rayner and Duffy 1986] Rayner, K., and Duffy, S. A. 1986. Lexical complexity and fixation times in reading: Effects of word frequency, verb complexity, and lexical ambiguity. *Memory & Cognition* 14(3):191–201.

[Tur, Hakkani-Tur, and Heck 2010] Tur, G.; Hakkani-Tur, D.; and Heck, L. 2010. What is left to be understood in atis? In *SLT, 2010 IEEE*, 19–24. IEEE.

[Von Ahn and Dabbish 2004] Von Ahn, L., and Dabbish, L. 2004. Labeling images with a computer game. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, 319–326. ACM.

[Wang et al. 2012] Wang, W. Y.; Bohus, D.; Kamar, E.; and Horvitz, E. 2012. Crowdsourcing the acquisition of natural language corpora: Methods and observations. In *SLT 2012*, 73–78. IEEE.

[Wang, Heck, and Hakkani-Tur 2014] Wang, L.; Heck, L.; and Hakkani-Tur, D. 2014. Leveraging semantic web search and browse sessions for multi-turn spoken dialog systems. In *ICASSP 2014*, 4082–4086. IEEE.

[Williams et al. 2013] Williams, J.; Raux, A.; Ramachandran, D.; and Black, A. 2013. The dialog state tracking challenge. In *Proceedings of the SIGDIAL 2013 Conference*, 404–413.

[Xu and Sarikaya 2014] Xu, P., and Sarikaya, R. 2014. Targeted feature dropout for robust slot filling in natural language understanding. In *Fifteenth Annual Conference of the International Speech Communication Association*.

[Young 2006] Young, S. 2006. Using pomdps for dialog management. In *SLT*, 8–13.