WingIt: Efficient Refinement of Unclear Task Instructions

V. K. Chaithanya Manam, Alexander J. Quinn
Purdue University
West Lafayette, Indiana, USA
{vmanam, aq}@purdue.edu

Abstract

Crowdsourcing has the inherent potential to shift tedious work from requesters with limited time (or patience) to an on-demand workforce. However, the time saved by requesters is offset by the time they must spend preparing instructions and refining them to address the ambiguities that typically arise. Hastily written instructions should be welcomed and supported. This paper presents a set of methods that enable workers to cope with unclear or ambiguous instructions and produce high quality results with minimal reliance on the requester. Workers' intuition about the requester's needs is leveraged to move onus for answering questions and revising instructions to workers. Our system, WingIt, implements these methods and demonstrates the relative tradeoffs between resolving by questions versus editing instructions directly, and between waiting for immediate response from the requester versus allowing workers to proceed based on their best guess of the requester's intent.

Introduction

Crowdsourcing platforms offer busy on-demand help with tedious work. The principal requirement–other than payment–is a description of the problem that all workers can understand and follow. When instructions are vague, inconsistent, imprecise, or ambiguous, they leave room for interpretation, and this ultimately results in diminished accuracy (Gadiraju, Yang, and Bozzon 2017). Quality management efforts (Ipeirotis, Provost, and Wang 2010) may result in rejection (in the case of Mechanical Turk). Clear communication of requirements is essential to workers and requesters alike.

Designing clear instructions that reliably elicit accurate results requires time and attention. Several iterations may be required, especially for tasks with intricate requirements. The aforementioned requisites offset the benefits of delegating the task in the first place.

To the requesters, the instructions may seem intuitive and straightforward because they have a clear picture of the expected response to the tasks. They expect the workers to understand it from their point of view. However, workers may have difficulty in understanding the task due to lack of prior knowledge, or due to their cultural and educational background. Even though the task may not be difficult, without sufficient information or guidance, workers may find it difficult. Another reason for improper instructions is due to situations that the requester might not be aware of. For instance, if the requester posts a task to get the name of the chair of the computer science department in a specific university, what should a worker do when there is more than one chairperson? In crowdsourcing, these situations arise frequently when searching for information. If workers do not get an answer to their question, they will either give an answer that they feel is correct, which might not be in alignment with what the requester needs or what other workers think is correct. To overcome such situations, we propose two methods called *Q&A* and *Edit* which are designed to facilitate the disambiguation process with the requester. In cases where the requester is unavailable (inactive), workers can collaborate with each other and find the best possible solution to the problem.

According to the reviews on TurkOpticon (Lilly C. and M. Six 2013) and prior research (Gadiraju, Yang, and Bozzon 2017), there have been several instances where workers were not able to complete a task or submitted a task with incorrect results due to ambiguous or sloppy instructions. This results in the squandering of workers' time and effort. To the best of our knowledge, we are the first to provide a reactive approach for problems that arise due to improper task instructions.

The key contributions of this paper are as follows:

- We present Q&A and Edit methods to resolve ambiguity in task instructions.
- Our results show that WingIt gives better results compared with Naïve.

In this paper, we will first present related work on ambiguity, task specification, the quality of work in crowdsourcing. We will then describe our system WingIt and present the evaluation of the system. Finally, we will discuss the results and conclude with future extensions.

Related Work

WingIt is related to prior work in ambiguity and task specification in crowdsourcing, and the quality of work in crowdsourcing.

Ambiguity and Task Specification in Crowdsourcing

Philipp Gutheim et al. proposed Fantasktic (Gutheim and Hartmann 2012), a system for novice requesters to create a better task and receive a higher quality of response from the crowd. Fantasktic proposed three task design techniques: a guided task specification interface, a preview interface, and a worker tutorial. A guided task specification interface provides guidelines and recommendations to the requester while creating a task. A preview interface presents the task from the perspective of the worker to the requester. A worker tutorial is generated automatically based on the sample answers provided by the requester. The quality of responses from workers showed significant improvement with instructions based on the guided task interface. Task previews and worker tutorials did not have any impact on the workers' response. Fantasktic is a proactive task design technique that helps novice workers to create a better task. However, WingIt is a reactive task design technique and can be used by both novice and expert requesters. WingIt provides the possibility of worker-requester interactions which are not provided by Fantasktic.

Gaikwad et al. proposed Daemo (Gaikwad, Whiting, and others 2017), a proactive mechanism to identify ambiguities in the task instructions. Daemo allows requesters to post a few instances of the task to workers, get their feedback, and improve the task based on their feedback. Their results show that an improved task gives a better quality of results compared to the original task. Similar results are seen with survey data. By improving the clarity of survey questions, the quality of results also improved (Fowler Jr 1992). With Daemo, workers can identify only the problems that are described within a few instances of the task. If there are problems with the other instances of the task, Daemo may not be able to find them. However, WingIt provides a reactive mechanism where workers can identify and resolve the ambiguities in the task while they are working on it.

Prior research on crowdsourcing emphasizes the impact of task instructions and design on the quality of the result (Marshall and Shipman 2013; Berg 2016; Kittur, Chi, and Suh 2008; Alagarai Sampath, Rajeshuni, and Indurkhya 2014). Gadiraju et al. identified the role of task instructions in crowdsourcing and modeled the task clarity with predictive features (Gadiraju, Yang, and Bozzon 2017). Meng-Han et al. studied in detail on how the quality of instructions affect the quality of crowd work (Wu and Quinn 2017). Alonso and Baeza-Yates suggested providing clear and colloquial instructions is an important part of task design (Alonso and Baeza-Yates 2011). Grady and Lease identified the importance of wording and terminology in task design (Grady and Lease 2010). Kittur et al. suggested improving task design through better communication as one of the crucial next steps in designing the future crowdsourcing solutions (Kittur et al. 2013). Khanna et al. showed that user interface, task instructions, and the workers' cultural context are the key barriers that prevent workers with scant digital literacy skills from participating and completing tasks on AMT (Khanna et al. 2010). Chang et al. proposed a collaborative system, Revolt, to deal with incomplete or ambiguous guidelines of image-labeling tasks (Chang, Amershi, and Kamar 2017). Revolt allows multiple workers to label the task with the specified instructions, and when there is a conflict, workers relabel it based on the description provided by other workers. Salehi et al. proposed a content creation workflow for complex writing tasks (Salehi et al. 2017). The workflow consists of the worker posting a series of questions to the requester before starting the task. After writing the paragraph, the requester rates it and communicates this to the worker. The worker then edits their work based on the rating and the new information from the requester.

Researchers in crowdsourcing have built innovative applications for specific uses that hide the complexity of task specification. For instance, Adrenaline demonstrated the use of a real-time camera to select the best photographic moment in a short movie with the help of crowdsourcing (Bernstein et al. 2011). Soylent, a powerful word processing tool, edits, shortens, and proofreads documents with the help of crowdsourcing (Bernstein et al. 2010). By adding Soylent as a plug-in for Microsoft Word, researchers were able to hide the complexity of task specification through a simple interface. Turkomatic (Kulkarni, Can, and Hartmann 2012), CrowdForge (Kittur et al. 2011), and Crowd4u (Ikeda et al. 2016) help with decomposing a complex task posted in natural language into small tasks for crowdsourcing platforms. These applications are successful, however their approaches are not generalizable to all the task specifications in crowdsourcing.

Quality of Work in Crowdsourcing

To improve the quality of work in crowdsourcing, ESP Game (von Ahn and Dabbish 2004) proposed the idea of multiple workers working simultaneously and then aggregating their responses by voting. Le et al. used qualifying tests called gold standards to preselect qualified users (Le et al. 2010). Peer review workflow (Bernstein et al. 2010) allows workers to rate a sample answer, and iterative workflow allows workers to collaborate and improve answers from previous workers (Little et al. 2010; Kulkarni, Can, and Hartmann 2012). Sampath et al. proposed a cognitive-inspired task design which makes tasks efficient for workers to provide correct responses (Alagarai Sampath, Rajeshuni, and Indurkhya 2014). Basu and Christensen proposed methods to teach and educate the crowd (Basu and Christensen 2013). Bragg et al. showed that an adaptive teaching strategy for the crowd will be more effective than fixed-length teaching (Bragg, Mausam, and Weld 2015).

WingIt proposes another way to improve the quality of work via interaction between worker and requester when the requester is active. In the situation of an inactive requester, we consider an iterative workflow, allowing workers to collaborate and improve answers from previous workers.

Ambiguity Classification

We present the classification of ambiguities (Table 1) in task instructions by using the input-process-output (IPO) model, which is widely used in system analysis and software engineering for analyzing a process (Bushnell 1990;

Ambiguity type		Example Task	Problem
Input	Entity	When was the Harry Potter movie released?	Which one in the series?
	Syntax	Find the headquarters of Amazoon.	Do you mean Amazon?
	Wrong	Find the price of Samsung Chromecast.	Do you mean Google Chromecast or Samsung Chromebook?
	Units	Find a city in Texas that has an average temperature between 20 to 30 during May.	Do you mean 20 to 30 Celsius or Fahrenheit?
Process	Steps	Find the i10-index and h-index for Prof. Don Norman.	How are i10-index and h-index found?
	Words	Find the weight of the any smartphone in MOTO G that can be purchased on BestBuy.com for $30000 to $49900.	Do you mean $300.00 to $499.00?
	Wrong	Go to the Google Store (https://store.google) and search for the Nexus Pixel. Enter the price of the least expensive option.	Do you mean store.google.com?
Output	Entity	Find the cost of attending this university for undergraduates.	In-state or out-of-state?
	Exception	Find an new Moto G mobile phone that costs less than $100.	What if no phone is available for this criteria?
	Units	Find the maximum capacity of a portable hard disk from Seagate.	In GB or TB?
	Format	Find the phone number for Internal Revenue Service.	Which format? XXX-XXX-XXXX or (XXX) XXX-XXXX ?
	Precision	Find the address of a movie theatre close to the SFO airport.	How close? Walking or Driving?

Table 1: Classification of ambiguity in task instructions. Example tasks shown are paraphrased.

Ilgen et al. 2005). Ambiguity can occur when the requester tries to describe a task that workers need to perform (input ambiguity), how they need to perform the task (process ambiguity), what workers need to submit back to requester (output ambiguity). The mathematical representation of our model which indicates the completeness of our classification is as follows.

f	(x)	$\longrightarrow$	$\{l_1 : y_1, l_2 : y_2, l_3 : y3, ..\}$
process	input		output

We further classify each of these ambiguities as follows.

Input Ambiguity

Entity: Input task description may contain some entities which have different meanings depending on the interpretation.
Syntax: Words in the input task description are misspelled to a degree that could cause a reasonable worker to doubt if they were doing the task correctly.
Wrong: Description of the Input task is incomprehensible but one might infer what was intended.
Units: Missing the units of the input task.

Process Ambiguity

Steps: Steps to perform a task are missing, ambiguous, or possibly do not exist.
Words: Workers are not able to understand the process description due to vocabulary, mechanics, etc.
Wrong: Steps described to perform the task are wrong, or lead to results that don't make sense.

Output Ambiguity

Entity: By following the process with given input leading to multiple outputs (e.g., Who was president in January 2017?)
Exception: By following the process with given inputs leading to abnormal results or no result.
Units: When the units for work that need to be submitted by the worker are not specified. For instance, if the requester asks a worker to find the average temperature of Chicago in 2016, should the data be entered in Celsius or in Fahrenheit?
Format: When the requester does not specify the format of output, the format that the results should be submitted in will be ambiguous. For instance, if the requester asks a worker to send the address of a Walmart store near ORD airport, then in which format should the worker send the address?
Precision: When describing a quantity, how precisely the quantity should be described.

WingIt

WingIt is a system we developed to facilitate high quality answers to adhoc tasks, even when the instructions are sloppily written and the requester's availability to clarify them is limited. We assume that a requester who offloads a modest job (e.g., a few hours) to crowd workers is busy, so any time spent perfecting the instructions or answering inquiries from workers will offset the value. Therefore, we shift the burden to workers (with compensation). As such, our focus is on the worker's experience.

For simplicity, we narrow the focus of our implementation (and evaluation) to information retrieval tasks consisting of a text prompt (e.g., *"Find the color of the house at the following address."*) and a single parameter (e.g., *"123 Bug St, Pillville"*).

When a worker encounters an ambiguity, they can ask for

clarification in the form of a question ("Q&A") or a direct revision to the instructions ("Edit"). Either way, the worker must use their intuition to form a best guess of what they *think* the requester wanted. Thus, when asking a question ("Q&A"), they must also propose a possible answer. If the requester agrees with the worker's best guess (i.e., proposed answer or edit), they can simply click to confirm it, thus minimizing the requester's effort.

When a worker encounters an ambiguity while attempting to perform a task, WingIt provides one of the following options:

1. Ask the requester a question, propose a possible answer, and wait for confirmation from the requester before submitting. ("Synchronous Q&A")
2. Ask the requester a question, propose a possible answer, and proceed assuming the answer reflects the requester's intentions. The requester can respond at a later time. ("Asynchronous Q&A")
3. Edit the question, and wait for confirmation from the requester before submitting. ("Synchronous Edit")
4. Edit the question, and proceed assuming the edit reflects the requester's intentions. The requester can respond at a later time. ("Asynchronous Edit")

For the synchronous cases ("Sync __"), the worker is promised a response within 3 minutes. The requester can either confirm or revise the answer or edit, after which it becomes part of the specification for all workers.

For the asynchronous cases ("Async __"), as well as the synchronous cases, the worker's answer or edit becomes part of the specification for all other workers going forward, until and unless the requester reviews it and disagrees. In that case, any results received in the meantime are discarded. This is the risk a requester must accept in order to avoid the interruptions entailed by the synchronous cases.

In all cases, workers are offered a bonus if their suggested clarification (answer or edit) matches the requester's intent. On the surface, this might resemble a payment for mind-reading. However, our past experiences with Mechanical Turk (as well as homework assignment specifications) suggest that workers (like students) can often work through imperfect specifications by using context and common sense assumptions about what would constitute a reasonable task. Our experiments with WingIt (presented later in this paper) validate this.

Synchronous Q&A

In synchronous Q&A, when workers are working on a task, if they find ambiguities in the task or if any information to complete the task is missing, they can ask the requester in the form of a Question and Answer (Q&A). In our model, we encourage workers to find ambiguities in the task instructions by paying a bonus. When a worker asks a Q&A, the requester will receive the Q&A in an email stating that a worker is working on the task and he/she has some questions related to the task along with the guessed answer. The requester will have a choice either to approve the Q&A if it is a valid Q&A or modify the answer to make it correct. The updated Q&A will be sent back to the worker and the worker can continue to work on the task.

Figure 1 shows an example of a task asking workers to find the release date of a Harry Potter movie. When the worker has a question, s/he will post the question along with the best-guess answer and wait for a short period of time (say 3 minutes) to get reply from the requester. When a worker submits a Q&A, the requester will receive an email saying that worker has questions related to the task instructions and has also provided their best guess to the question. The requester will review the Q&A and either approve the Q&A without any modifications if the worker's guess is correct or modify the Q&A when the worker's guess is incorrect. All the Q&A that are approved by the requester will be shown in the Q&A section along with the time stamp of when it was approved by the requester. When a new worker works on this task, she will read the task and then go through the Q&A section to get more details about the task before working on the task.

Synchronous Edit

In Synchronous Edit, workers will have the power to edit the instructions when they find a problem with them. When a worker modifies the instructions, the requester will get an email stating the instructions have been changed by a worker. If the modified instructions are in line with the requester's needs, the requester will approve the modifications and a notification is sent to the worker saying that the modifications are approved. These updated instructions will be available to all workers who work on this task in the future. If the edited instructions are not in line with what requester needs, then the requester will revise the instructions with the correct information. Synchronous Edit helps improve the quality of instructions, and when a new worker takes up the job, it will take less time to read the instructions rather than reading the Q&A.

Figure 2 shows an example of the task finding the release date of a Harry Potter movie. When the worker finds ambiguities within the task instructions, s/he will edit the original instructions to resolve all the ambiguities that were found in the task instructions. Once the worker modifies these instructions, s/he will wait for a short period of time (3 minutes) to get a response from the requester. The requester will receive an email notification with the updated instructions. The requester will either approve or improve these modified instructions. Worker will receive a notification saying that the requester has approved the modifications. Other workers of the same task will receive a notification saying that an update to these instructions are suggested by the requester and showing the difference between the current instruction and the modified/new instruction. Workers will have a choice either to accept or reject the modified instructions. Workers will receive a bonus for accepting the modified instructions.

Asynchronous Q&A

This is similar to Synchronous Q&A, except that instead of waiting for a response, workers may continue with the task, assuming their answer is correct. This will be more useful in

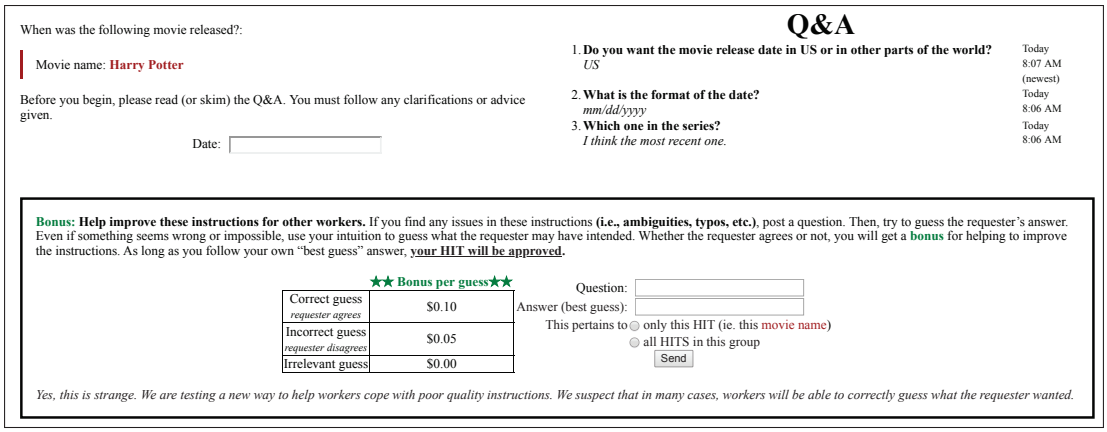

Figure 1: The Q&A user interface

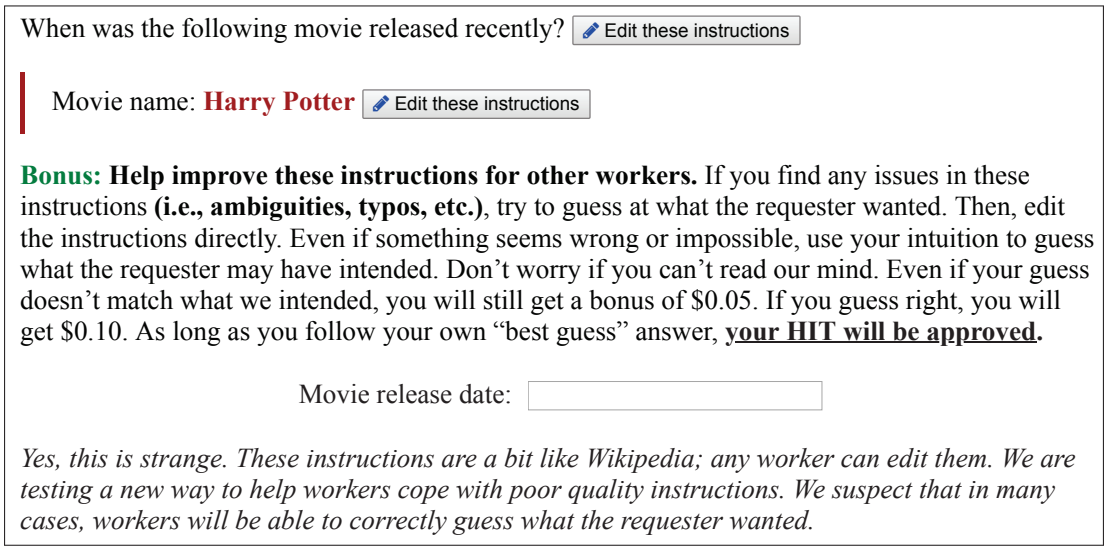

Figure 2: The Edit user interface

the cases where the requester cannot afford to respond to the worker's question within the short time frame.

The interface for Asynchronous Q&A is similar to Figure 1. When a worker finds an ambiguity within the task instructions, s/he posts the question and the best guess that resolves the ambiguity. In Asynchronous Q&A, workers' questions and best guessed answers will be posted in the Q&A section directly. Workers can assume that their guess is correct and continue working on the task.

Asynchronous Edit

This is similar to Synchronous Edit, except that instead of waiting for a response, workers may continue with the task assuming their edit to the instructions is acceptable.

The interface for Asynchronous Edit is similar to Figure 2. When a worker finds an issue with the task instructions, s/he can modify the instructions. These modifications will be notified to any other workers who are working on the same task, similar to Synchronous Edit. Other workers will have a choice either to accept or reject the modified instructions. Workers will be paid a bonus for accepting the modified instructions.

The main advantage with Synchronous cases is that the worker will get a response to their questions in real-time and we believe that this will increase confidence levels of workers when working on the task. The disadvantage is that the worker needs to wait until requester replies back to the question and this waiting would increase the cost per hit that the requester needs to pay for the worker.

Study Design

In this section, we describe experiments we conducted to investigate the benefit of WingIt compared to traditional crowdsourcing. We conducted our experiments on Amazon's Mechanical Turk (AMT) platform. We describe the task that workers were asked to perform, specific hypotheses driving our study, and various analysis methods we employed.

Experiment Design

Evaluating the WingIt method required a set of tasks that (a) contained a small number of bona fide ambiguities, (b) spanned the full range of ambiguity types that we knew to be applicable to web search tasks, (c) had objective ground truth answers available, and (d) fit within a consistent overarching task structure (i.e., textual instruction prompt plus a single textual parameter). We know of no such existing dataset, nor any way to procure one through ecologically valid means. Therefore, we produced a new set of tasks according to those specifications. In our study, we planted a total of 65 ambiguities in the thirty-four tasks that comprise of all the different types of ambiguities.

There were five experimental conditions, including the four methods described in the introduction ("Sync Q&A", "Async Q&A", "Sync Edit", "Async Edit") and a naïve baseline, which provided no means of seeking clarification. Workers were randomly assigned to one of these treatments when they first viewed the task (whether previewing or accepted). To the extent possible, they were prevented from seeing any of the other conditions. The experiments followed a between-subject design.

Each worker is allowed to work on a maximum of one HIT in each HIT type. In our work, we are only interested in understanding the effect of Q&A and Edit methods; hence, we do not differentiate workers based on geographic location, prior experience, and personal characteristics. We paid $0.40 per HIT, plus a bonus of $0.10 for a correct guess, $0.05 for an incorrect guess, and $0.00 for irrelevant guesses. Each worker is allowed to work on at most one instance of the task. A total of 304 distinct workers have participated in our study.

In Synchronous Q&A, when a worker asks a question along with a guessed answer, the requester either approves it with a single click when the guessed answer is correct or corrects the answer according to the question, and then approves it. In Synchronous Edit, when a worker modifies the task, the requester can see the modifications in the task instructions in the form of colored text showing the difference between the original text and the modified text. This will help the requester give quick feedback to the worker. When the modification is valid, the requester approves it with a single click or corrects the task based on workers' modification and approves it. We (authors) acted as requesters in this study.

Research Questions and Hypotheses

Our study addressed the following research questions:

Q1: How does the choice of Sync vs. Async affect performance?

Q2: How does Edit vs. Q&A affect performance?

Q3: Can the crowd identify and resolve ambiguity according to the expectations of requester?

H1: Accuracy will be higher with WingIt.

We compare workers' results with our results, and label them as correct or incorrect accordingly. To understand whether WingIt performs better in terms of accuracy compared to Naïve, we conducted a chi-square test for the results of the task with WingIt and Naïve. Figure 3 shows the treatments and corresponding accuracy of results. From Figure 3, it is clear that there is a difference in the accuracy of results with different treatments. The order of the results' accuracy from highest to lowest is Sync Edit, Sync Q&A, Async Q&A, Async Edit, and Naïve. To check the significance of the difference, we conducted a chi-square test on the accuracy of results. There was a significant association between the WingIt and the accuracy of results, $\chi^2(4) = 26.09$, $p < 0.001$. For pairwise comparisons between treatments, we conducted chi-square posthoc analysis with Bonferroni correction. The posthoc analysis showed that there is no significant difference between the results of Sync Q&A and Sync Edit, and also that there is no significant difference between the results of Naïve, Async Q&A, and Async Edit.

Accuracy of Sync Q&A and Sync Edit is higher than that of Async Q&A and Async Edit due to the presence of the requester to validate the Q&As and Edits made by the worker. When a worker makes a correct guess in Async Q&A and Async Edit, it helps other workers who are working on the same task in different instances to submit quality results. However, if they make a wrong guess, it will pollute other workers and force them to submit wrong results. We observed that the beneficial effect of correct guesses nullified wrong guesses in Async Q&A and ASync Edit and hence its accuracy is similar with Naïve.

H2: Completion time will be higher with WingIt.

Sync Q&A and Sync Edit will have a higher completion time compared with Async Q&A and Async Edit as workers need to wait for a reply from the requester when they submit a question or edit the instruction. Naïve should have less completion time compared to WingIt. Figure 4 shows the workers' completion time for different treatments. The Kruskal-Wallis test results suggest that workers' completion time is significantly affected by the treatment, $H(4) = 16.84$, $p = 0.002$.[1] Comparison of mean ranks between the treatment groups showed that workers' completion time is not significantly different with Async Q&A, (difference = 1.44), Async Edit, (difference = 10.66) or Sync Edit (difference = 5.42) compared to Naïve. However, workers' completion time is significantly higher with Sync Q&A than with Naïve (difference = 64.65). In all cases, the critical difference ($\alpha = 0.5$ corrected for the number of tests) was 57.93. The reason for Sync Edit having a lower completion time than Sync Q&A might be that the time taken by the requester to approve or modify the task instruction in Sync Edit is less than that of Sync Q&A, due to the colored text in Sync Edit. Async Q&A and Async Edit having similar completion times to Naïve may be due to the fact that the time taken to edit the text or write Q&A is considerably less compared to the time taken to submit the results. Naïve has a similar completion time to Async Q&A and Async Edit as

[1] One-way ANOVA was not used due to non-homogeneous variance

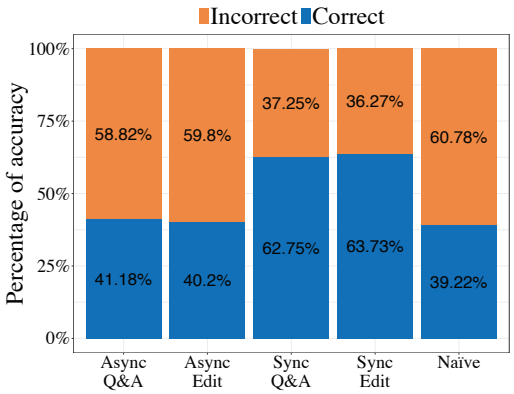

Figure 3: Accuracy vs Treatment

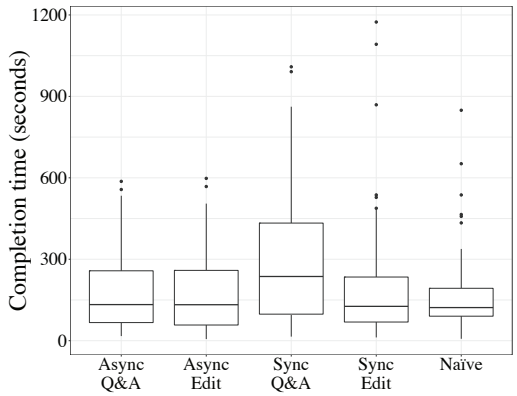

Figure 4: Workers' completion time vs Treatment

the workers might have taken more time to resolve ambiguity in Naïve compared to Async Q&A and Async Edit.

H3: Accuracy will be higher with Sync.

There was a significant association between the Sync and Async treatments and the accuracy of results, $\chi^2(1) = 20.77, p < 0.001$. This seems to represent the fact that based on the odds ratio, Sync produced 2.5 times more accurate results compared with Async. We observed that workers are able to guess most of the ambiguities in the task instructions, but they were not able to guess the correct answer from the set of available answers. This is the reason for Sync to produce a higher accuracy in results compared with Async and baseline.

H4: Completion time will be higher with Sync.

Workers in Sync need to wait for some time to get a reply from the requester when they post a question or change the task instructions. A Wilcox test showed that there is a significant difference in the completion time with Sync and Async. The completion time of Async ($Median = 136$) was significantly less than Sync ($Median = 173$), $W = 17630$, $p < 0.05, r = -0.132$.[2]

H5: Number of edits/Q&A will be higher with Sync.

We hypothesize that workers will tend to ask more questions when the requester is present. To verify our hypothesis, we conducted a t-test on the number of edits/Q&As in Sync vs Async. On average, the number of edits/Q&As in Sync ($Mean = 2.68, SE = 2.14$) is higher than in Async ($Mean = 1.94, SE = 2.19$). The difference was significant $t(133.92) = -1.98, p < 0.05, r = 0.17$.

H6: Completion time will be lower with Edit.

Editing the task to fix the ambiguity would consume less time compared to writing a question and an answer to the question. A Wilcox test showed that there is a significant difference in the completion time with Edit compared to Q&A. The completion time of Edit ($Median = 139.50$) was significantly less than that of Q&A ($Median = 191.5$), $W = 23642, p < 0.05, r = -0.118$.

H7: Number of edits will be lower than number of Q&A.

Workers can fix multiple ambiguities in the task with a single edit. However, they need provide a Q&A for each of the ambiguity they identify. Hence, number of edits will be lower than number of Q&A. However, Wilcox test showed that there is a significant difference between the number of edits and the number of Q&A. The number of edits ($Median = 3$) was significantly higher than the number of Q&A ($Median = 1$), $W = 1519.5, p < 0.001, r = -0.301$. When we checked our experiment data, we found that many workers with edit method are trying to paraphrase the given task in different possible ways which is not possible with the Q&A method. So, the number of edits are higher than number of Q&A.

H8: Requester will receive more Q&A than Edits.

Workers need to ask separate questions for each ambiguity that they identify in the task, but they can fix all the ambiguities with a single modification of the task. So, we hypothesize that the requester will receive more Q&A than Edits. A Wilcox test showed that there is a significant difference in the number of edits and the number of Q&A that the requester receives. The number of edits ($Median = 3$) was significantly higher than the number of Q&A ($Median = 2$), $W = 324, p < 0.01, r = -0.385$. In Sync Q&A, we observed that most of the questions are related to the ambiguities in the task; however, in Sync Edit, we observed that workers tend to edit the tasks more, such as paraphrasing the text, correcting the grammar, and changing the vocabulary. Such edits are not possible with Q&A. Hence, the total number of edits are significantly higher than Q&A.

[2]Student's t-test was not used due to non-homogeneous variance

H9: The number of questions/edits the requester receives depends on the number of ambiguities in the task.

To test whether there is any dependency between the number of edits/Q&A the requester receives to the total number of ambiguities in the task, we have built a linear model with the number of ambiguities in the task as the predictor and the number of questions/edits the requester receives as the outcome. The number of ambiguities in the task significantly predicted the number of Q&As ($b = 0.78$, $\beta = 0.29$, $t = 2.64$, $p < 0.05$) and edits ($b = 0.97$, $\beta = 0.62$, $t = 1.56$, $p < 0.05$) the requester received in Sync Q&A and Sync Edit respectively. The beta value indicates that as the number of ambiguities in the task increases, the number of Q&As/edits the requester receives also increases. Hence, the number of questions/edits the requester receives depends on to the number of ambiguities in the task.

Discussion and Future Work

Our study shows that there is no significant difference between the Q&A and Edit methods in the accuracy of results. Accuracy of results can only be improved by the presence of the requester. We found that requesters who spend a small amount of time in validating the workers' Q&A's/edits would improve the quality of results significantly. WingIt utilizes the guessing ability of workers to guess the requesters' needs. The amount of effort required by the requester is greatly reduced from simply approving the workers' Q&A/edits with one click or by making small changes to the workers' Q&A/edits. WingIt provides a way to share the knowledge of solutions to unclear instructions with other workers. Over time, when all the ambiguities are cleared, we believe that WingIt can produce 100% accuracy of results.

Why are workers not able to guess all the ambiguities in a task?

We planted a total of 65 ambiguities. Workers were able to guess 13 of the ambiguities with Async Q&A, 10 with Async Edit, 25 with Sync Q&A, and 24 with Sync Edit. An examination of the responses provided by workers suggests some possible reasons for the low accuracy of the guesses.

- Workers may use units common in their locale, if the expected units are not specified. Example: When prompted for the price of an Apple iPhone 6S, four workers in the US and India entered prices in USD and INR, respectively.
- Workers may trust supplementary facts presented by search engines above the results (e.g., Google Now "cards"), without scrutinizing the relevance or timeliness of the information. Example: When prompted for the release date of the *Power Rangers* movie in the USA, some incorrectly entered March 22, 2017. At that time, we found that a Google search results for "Power Ranger" included a supplementary panel with that same date, which was the correct release date for Indonesia but not (at the time) labelled as such. We have no direct evidence of the workers' searches; this is one possible explanation for this mistake.

We studied the significance of WingIt with information search tasks. However, this can be extended to other kind of tasks. We believe that the results we have obtained with information search tasks are parallel with other kinds of tasks. In the future, we would like to build a large data set of ambiguous tasks that other researchers can utilize to build systems like WingIt that can help workers to resolve ambiguities in the task and improve their quality of work. When we have a large data set, we can study the ability of WingIt to identify each type of ambiguity.

Conclusion

This paper presented WingIt, a new approach for resolving ambiguities in task instructions. Comparing WingIt to a traditional approach demonstrates that WingIt can improve the quality of results and decrease the effort of requesters in creating comprehensive instructions that leave no room for subjective assessment. Our results show that there is no difference in the accuracy of results between Edit and Q&A methods. We also found that workers are more interested in asking questions or making edits to the tasks in the presence of a requester. The number of questions/edits the requester receives depends on the number of ambiguities in the task.

References

Alagarai Sampath, H.; Rajeshuni, R.; and Indurkhya, B. 2014. Cognitively inspired task design to improve user performance on crowdsourcing platforms. In *CHI '14: Proceedings of the 32nd Annual ACM Conference on Human Factors in Computing Systems*, 3665–3674. New York, NY, USA: ACM.

Alonso, O., and Baeza-Yates, R. 2011. Design and implementation of relevance assessments using crowdsourcing. In *ECIR '11: Proceedings of the 33rd European Conference on Information Retrieval*, 153–164. Dublin, Ireland: Springer.

Basu, S., and Christensen, J. 2013. Teaching classification boundaries to humans. In *AAAI '13: Proceedings of the 27th AAAI Conference on Artificial Intelligence*, 109–115. Bellevue, Washington, USA: AAAI Press.

Berg, J. 2016. Income security in the on-demand economy: findings and policy lessons from a survey of crowdworkers. *Comparative Labor Law and Policy Journal* 37(3).

Bernstein, M. S.; Little, G.; Miller, R. C.; Hartmann, B.; Ackerman, M. S.; Karger, D. R.; Crowell, D.; and Panovich, K. 2010. Soylent: A word processor with a crowd inside. In *UIST '10: Proceedings of the 23rd Annual ACM Symposium on User Interface Software and Technology*, 313–322. New York, NY, USA: ACM.

Bernstein, M. S.; Brandt, J.; Miller, R. C.; and Karger, D. R. 2011. Crowds in two seconds: Enabling realtime crowd-powered interfaces. In *UIST '11: Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*, 33–42. New York, NY, USA: ACM.

Bragg, J.; Mausam; and Weld, D. S. 2015. Learning on the job: Optimal instruction for crowdsourcing. In *ICML '15: Proceedings of the ICML Workshop on Crowdsourcing and Machine Learning*, 17–21. New York, NY, USA: PMLR.

Bushnell, D. S. 1990. Input, process, output: A model for evaluating training. *Training & Development Journal* 44(3):41–44.

Chang, J. C.; Amershi, S.; and Kamar, E. 2017. Revolt: Collaborative crowdsourcing for labeling machine learning datasets. In *CHI '17: Proceedings of the 35th Annual ACM Conference on Human Factors in Computing Systems*, 2334–2346. New York, NY, USA: ACM.

Fowler Jr, F. J. 1992. How unclear terms affect survey data. *Public Opinion Quarterly* 56(2):218–231.

Gadiraju, U.; Yang, J.; and Bozzon, A. 2017. Clarity is a worthwhile quality: On the role of task clarity in microtask crowdsourcing. In *HT '17: Proceedings of the 28th ACM Conference on Hypertext and Social Media*, 5–14. New York, NY, USA: ACM.

Gaikwad, S. N. S.; Whiting, M. E.; et al. 2017. The daemo crowdsourcing marketplace. In *CSCW '17: Proceedings of the 20th ACM Conference on Computer Supported Cooperative Work and Social Computing*, CSCW '17 Companion, 1–4. New York, NY, USA: ACM.

Grady, C., and Lease, M. 2010. Crowdsourcing document relevance assessment with mechanical turk. In *CSLDAMT '10: Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, 172–179. Stroudsburg, PA, USA: ACL.

Gutheim, P., and Hartmann, B. 2012. Fantasktic: Improving quality of results for novice crowdsourcing users. Master's thesis, EECS Department, University of California, Berkeley. [Online; accessed September 1, 2017].

Ikeda, K.; Morishima, A.; Rahman, H.; Roy, S. B.; Thirumuruganathan, S.; Amer-Yahia, S.; and Das, G. 2016. Collaborative crowdsourcing with crowd4u. *Proceedings of the VLDB Endowment* 9(13):1497–1500.

Ilgen, D. R.; Hollenbeck, J. R.; Johnson, M.; and Jundt, D. 2005. Teams in organizations: From input-process-output models to imoi models. *Annual Review of Psychology* 56:517–543.

Ipeirotis, P. G.; Provost, F.; and Wang, J. 2010. Quality management on amazon mechanical turk. In *HCOMP '10: Proceedings of the ACM SIGKDD Workshop on Human Computation*, 64–67. New York, NY, USA: ACM.

Khanna, S.; Ratan, A.; Davis, J.; and Thies, W. 2010. Evaluating and improving the usability of mechanical turk for low-income workers in india. In *ACM DEV '10: Proceedings of the 1st ACM Symposium on Computing for Development*, 12:1–12:10. New York, NY, USA: ACM.

Kittur, A.; Smus, B.; Khamkar, S.; and Kraut, R. E. 2011. Crowdforge: Crowdsourcing complex work. In *UIST '11: Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*, 43–52. New York, NY, USA: ACM.

Kittur, A.; Nickerson, J. V.; Bernstein, M.; Gerber, E.; Shaw, A.; Zimmerman, J.; Lease, M.; and Horton, J. 2013. The future of crowd work. In *CSCW '13: Proceedings of the 16th ACM Conference on Computer Supported Cooperative Work*, 1301–1318. New York, NY, USA: ACM.

Kittur, A.; Chi, E. H.; and Suh, B. 2008. Crowdsourcing user studies with mechanical turk. In *CHI '08: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 453–456. New York, NY, USA: ACM.

Kulkarni, A.; Can, M.; and Hartmann, B. 2012. Collaboratively crowdsourcing workflows with turkomatic. In *CSCW '12: Proceedings of the 15th ACM Conference on Computer Supported Cooperative Work*, 1003–1012. New York, NY, USA: ACM.

Le, J.; Edmonds, A.; Hester, V.; and Biewald, L. 2010. Ensuring quality in crowdsourced search relevance evaluation: The effects of training question distribution. In *CSE '10: Proceedings of the SIGIR 2010 Workshop on Crowdsourcing for Search Evaluation*, 21–26. New York, NY, USA: ACM.

Lilly C., I., and M. Six, S. 2013. Turkopticon. https://turkopticon.ucsd.edu/. [Online; accessed September 1, 2017].

Little, G.; Chilton, L. B.; Goldman, M.; and Miller, R. C. 2010. Turkit: Human computation algorithms on mechanical turk. In *UIST '10: Proceedings of the 23rd Annual ACM Symposium on User Interface Software and Technology*, 57–66. New York, NY, USA: ACM.

Marshall, C. C., and Shipman, F. M. 2013. Experiences surveying the crowd: Reflections on methods, participation, and reliability. In *WebSci '13: Proceedings of the 5th Annual ACM Web Science Conference*, 234–243. New York, NY, USA: ACM.

Salehi, N.; Teevan, J.; Iqbal, S.; and Kamar, E. 2017. Communicating context to the crowd for complex writing tasks. In *CSCW '17: Proceedings of the 20th ACM Conference on Computer Supported Cooperative Work and Social Computing*, 1890–1901. New York, NY, USA: ACM.

von Ahn, L., and Dabbish, L. 2004. Labeling images with a computer game. In *CHI '04: Proceedings of the 22nd Annual ACM Conference on Human Factors in Computing Systems*, 319–326. New York, NY, USA: ACM.

Wu, M.-H., and Quinn, A. J. 2017. Confusing the crowd: Task instruction quality on amazon mechanical turk. In *HCOMP '17: Proceedings of the 5th AAAI Conference on Human Computation and Crowdsourcing*, 206–215.