# Sprout: Crowd-Powered Task Design for Crowdsourcing

**Jonathan Bragg**
University of Washington
Seattle, WA, USA
jbragg@cs.washington.edu

**Mausam**
Indian Institute of Technology
New Delhi, India
mausam@cse.iitd.ac.in

**Daniel S. Weld**
University of Washington
Seattle, WA, USA
weld@cs.washington.edu

## ABSTRACT

While crowdsourcing enables data collection at scale, ensuring high-quality data remains a challenge. In particular, effective task design underlies nearly every reported crowdsourcing success, yet remains difficult to accomplish. Task design is hard because it involves a costly iterative process: identifying the kind of work output one wants, conveying this information to workers, observing worker performance, understanding what remains ambiguous, revising the instructions, and repeating the process until the resulting output is satisfactory.

To facilitate this process, we propose a novel meta-workflow that helps requesters optimize crowdsourcing task designs and SPROUT, our open-source tool, which implements this workflow. SPROUT improves task designs by (1) eliciting points of confusion from crowd workers, (2) enabling requesters to quickly understand these misconceptions and the overall space of questions, and (3) guiding requesters to improve the task design in response. We report the results of a user study with two labeling tasks demonstrating that requesters strongly prefer SPROUT and produce higher-rated instructions compared to current best practices for creating gated instructions (instructions plus a workflow for training and testing workers). We also offer a set of design recommendations for future tools that support crowdsourcing task design.

### Author Keywords

Crowdsourcing; workflow; task design; debugging.

### CCS Concepts

**•Information systems → Crowdsourcing; •Human-centered computing → Interactive systems and tools;**

## INTRODUCTION

Ensuring high-quality work is considered one of the main roadblocks to having crowdsourcing achieve its full potential [34]. The lack of high quality work is often attributed to unskilled workers, though it can equally be attributed to inexperienced or time-constrained requesters posting imperfect task designs [15, 35]. Often, unclear instructions confuse sincere workers because they do not clearly state the task expectations [12]. In other cases, the task may be clear but complex; here, the lack of guided practice creates a mismatch between worker understanding and task needs [11, 23]. Finally, in many cases, the requesters themselves do not appreciate the nuances of their task, a priori, and need to refine their task definition [21].

Our hypothesis is that explicit or implicit feedback from workers can guide a requester towards a better task design. Unfortunately, existing tools for crowdsourcing fall severely short in this regard. While they often include best practice recommendations to counter variance in worker quality [10] (e.g., gold standard question insertion for identifying under-performing workers, aggregation of redundant labels), they do not provide mechanisms for supporting requesters in effectively defining and designing the task itself, which can mitigate the need for these downstream interventions.

In response, we present a novel meta-workflow that interleaves tasks performed by both crowd workers and the requester (see Figure 1) for improving task designs. SPROUT, our initial prototype, focuses on clarifying the task instructions and ensuring workers follow them, which are difficult [2] and important [12, 15, 23] aspects of task design. SPROUT evaluates a preliminary task design and organizes confusing questions by clustering explanations and instruction edits suggested by crowd workers. SPROUT's dashboard displays these organized confusions, allowing the requester to navigate their own dataset in a prioritized manner. The system goal is to support the requester in efficiently identifying sources of confusion, refining their task understanding, and improving the task design in response.

SPROUT provides additional support for ensuring workers understand the instructions. It allows requesters to embed illustrative examples in the instructions and recommends potential test questions (questions with reference answers that test understanding of the instructions). Upon acceptance by the requester, the instructions and test questions are compiled into *gated instructions* [23], a workflow consisting of an interactive tutorial that reinforces instruction concepts and a screening phase that verifies worker comprehension before commencing work (see Figure 2). Overall, SPROUT provides a comprehensive interface for requesters to iteratively create and improve gated task instructions using worker feedback.

We evaluate SPROUT in a user study, comparing it against *structured labeling* [21], a previous method that is likely to aid requesters in creating instructions [4], while their understanding of the task may be evolving (unassisted by workers).



*UIST '18,* October 14–17, 2018, Berlin, Germany



ISBN 978-1-4503-5948-1/18/10…$15.00

DOI: https://doi.org/10.1145/3242587.3242598

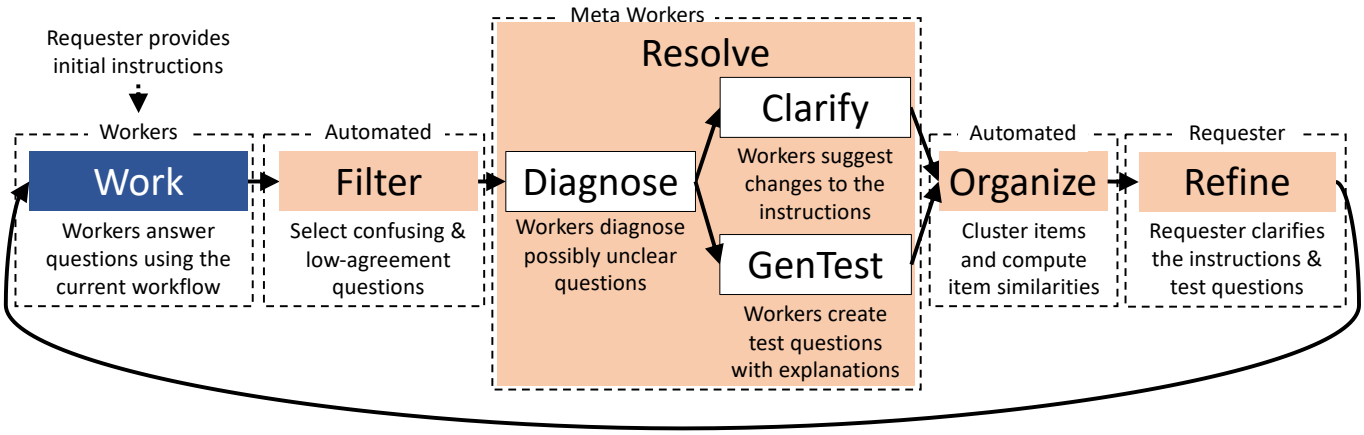


**Figure 1.** SPROUT **is our implemented task-improvement meta-workflow (workflow for improving a task workflow), that interleaves steps where workers answer questions using the base workflow (blue box) and meta steps (orange boxes), where meta-workers diagnose problems and suggest fixes, while** SPROUT **guides the requester to iteratively improve the instructions, add clarifying examples, and insert test questions to ensure understanding.**

Requesters who participated in our study created gated instructions for two different types of labeling tasks—the most common crowdsourcing task type [15]—strongly preferred and produced higher-rated instructions using SPROUT.

In summary, this paper makes four main contributions:

- A novel meta-workflow—combining the efforts of both crowd workers and the requester—that helps the requester create high-quality crowdsourcing tasks more quickly and with substantially less effort than existing methods.
- SPROUT, an open-source tool that implements this workflow for labeling tasks, the most common crowdsourcing task type [15]. SPROUT first has workers suggest changes to the task instructions. It then clusters the suggestions and provides the requester with a comprehensive task-improvement interface that visualizes the clusters for fast task exploration and semi-automates the creation of a gated instruction (training and testing) workflow by suggesting test questions related to the instructions the requester has written.
- A user study with requesters with varying amounts of crowdsourcing experience comparing SPROUT and structured labeling on two different types of labeling tasks. The results demonstrate an overall preference for SPROUT over structured labeling and for the use of worker feedback during task design. Furthermore, requesters using SPROUT produced instructions that were rated higher by experts.
- A set of design principles for future task authoring and debugging tools, informed by our experience building SPROUT, and our observations and discussion with requesters during the user study.

We implement the SPROUT tool as a web application and release the source code for both it and for structured labeling in order to facilitate future research.[1]

## PREVIOUS WORK

We first discuss (1) previous work that characterizes good task design and (2) gated instructions; then, we describe existing tools that help requesters (3) understand user behavior and (4) improve their task design.

### Design Principles for Tasks and Workflows

There is a small but growing body of work elucidating best practices for task design. CrowdFlower, a major crowdsourcing platform, reinforces that tasks should be divided into discrete steps governed by objective rules; they also highlight the importance of clear instructions [9] and test questions [10]. Several studies of worker attitudes also point to task clarity problems as a major risk factor for workers [12, 25, 35]. Furthermore, large-scale analyses have found positive correlations between task clarity features like the presence of examples and task performance metrics like inter-annotator agreement and fast task completion times [15]. Other controlled empirical studies provide further evidence that examples improve task outcomes [35]. Some work has sought to systematically understand the relative importance of various task design features [1, 35], but this work is limited to specific task types and general design principles remain poorly understood.

Emerging understanding of good workflow design suggests that investing in worker understanding is critically important to crowdsourcing outcomes. A large-scale controlled study compared the efficacy of different quality control strategies, concluding that training and screening workers effectively is more important than other workflow interventions [27]. Providing feedback about a worker's mistakes has also been shown to be very helpful in improving their answer quality [11].

These studies demonstrate the strong need for tools like SPROUT to help requesters clarify the task, include illustrative examples, provide training with feedback, and screen workers.

### Gated Instructions

The importance of instructions, training, and screening was also demonstrated by an analysis of several attempts to crowdsource training data for information extraction [23]. SPROUT adopts *gated instruction* from this work (see Figure 2). Gated instructions is a quality control method that uses test questions to ensure that workers have read and understood the instructions. It differs from the common practice of mixing 10–30%

[1] https://crowdlab.cs.washington.edu/task-design.html

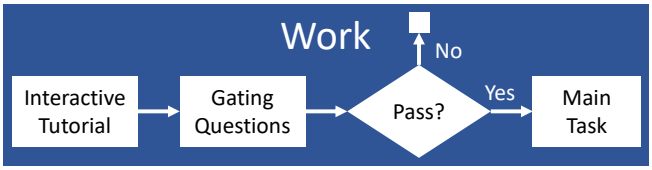


**Figure 2.** SPROUT **runs a gated instruction workflow [23] in the** *Work* **step of the meta-workflow (Figure 1), which ensures workers understand the instructions before starting the main task. Workers who do not pass gating do not continue with the task (indicated by the terminal square). The** *Refine* **step of the meta-workflow updates all parts of this workflow (before the first** *Refine* **step, only the main task is run since the system cannot yet construct tutorial or gating questions).**

gold standard questions into a work stream in the hope of detecting spammers [3,28], since the former is intended to ensure understanding not diligence. It also has advantages over other approaches like instructional manipulation checks [29], which test attentiveness, not understanding; can be gamed [14]; and do not provide training. A gated instruction workflow inserts two phases before the main task: an interactive tutorial, followed by a screening phase consisting of a set of questions workers must pass in order to start work on the actual task.

### Understanding Task Ambiguities and Worker Behavior

Several tools support understanding of task behaviors. CrowdScape provides visualizations to help requesters understand individual worker behaviors and outputs [31]. Noting that experimenting on different versions of task instructions, rewards, and flows is time-intensive, CrowdWeaver provides a graphical tool to help manage the process and track progress [20]. Cheng et al. [5] propose methods for automatically determining task difficulty. Kairam and Heer [17] provide methods for clustering *workers* (rather than questions, as SPROUT does). While there has been more emphasis on understanding worker behaviors, Papoutsaki et al. [30] instead study behaviors of novice requesters designing workflows for a data collection task; they distill several helpful lessons for requesters.

Other tools focus on dataset clustering and understanding. Structured labeling [21] is a tool that produces a clustered dataset with each cluster labeled by a single person akin to the requester. These "structured labels" are a flexible data asset (e.g., can support efficient data relabeling), but are expensive to produce. Revolt [4] outputs structured labels created by the crowd (to save requester effort). While structured labels are flexible, it is prohibitively expensive to run Revolt on large datasets because Revolt asks workers to explain and categorize *every* unresolved data item. Since Revolt's goal is not task improvement, it does not provide a user interface for the requester nor help the requester create good instructions; their evaluation used simulated requesters. In contrast, SPROUT uses the crowd to help requesters create unambiguous instructions (thereby improving task quality) by examining a larger, more diverse subset of the data than previously possible.

### Tools for Task and Workflow Design

SPROUT extends a line of research on tools that support designing and debugging workflows. We are inspired by Turkomatic [22], which proposed having workers themselves decompose and define a workflow to solve a high-level task description provided by a requester. Both systems embody a meta-workflow with crowd workers acting in parallel with the requester. While Turkomatic was only "partially successful" [22], the vision is impressive, and we see SPROUT as diving deeper into the task specification aspect of the greater workflow design challenge. SPROUT also leverages the reusability of instructions across many instances of a task, while Turkomatic considered one-off tasks where reusability is limited. Fantasktic [13] was another system designed to help novice requesters be more systematic in their task instruction creation via a wizard interface, but did not incorporate worker feedback or aid requesters in identifying edge cases like SPROUT. Developed in parallel with our work, WingIt [24] also has workers make instruction edits to handle ambiguous instructions, but does not provide a requester interface and relies on the requester approving or modifying each individual edit (which could be very time-consuming).

Forward-thinking marketplaces, such as CrowdFlower and Daemo [7], already encourage requesters to deploy prototype tasks and incorporate feedback from expert workers before launching their main task. These mechanisms demonstrate the feasibility and value of worker feedback for improving tasks. SPROUT makes this paradigm even more useful with a meta-workflow that produces structured task feedback, does not require expert workers, and enables requesters to efficiently resolve classes of ambiguities via a novel user interface.

## SPROUT: A TOOL SUPPORTING TASK DESIGN

In this section, we present the design of SPROUT, our system for efficiently creating gated task instructions for new tasks. The design decisions for SPROUT are based on previous work and the authors' extensive experience running crowdsourcing tasks, and were iteratively refined through pilot studies with workers and requesters (target users).

SPROUT embodies a feedback loop for task authoring and debugging. First, the requester writes a version of the instructions, which are shown to the crowd on an evaluation set (a small subset of the data) during a *Work* step of the meta-workflow (Figure 1). SPROUT identifies possibly confusing questions during an automated *Filter* step using signals such as low inter-annotator agreement. A different set of (meta)-workers then perform a *Diagnose* step (Figure 3b), where they decide if the question is ambiguous given the current instructions. Immediately after the *Diagnose* step, workers perform either a *Clarify* step (Figure 3c) where they edit the instructions based on their own definition of the task (if they diagnose the question to be ambiguous) or a *GenTest* step where they create a test question with an explanation (if they believe the question has an unambiguous answer). These three steps are implemented as a single, conditional Resolve HIT (Figure 3).

During a subsequent *Organize* step, SPROUT uses these edits and explanations to cluster various items and create item-item similarity scores. These clusters (and closely related items) are exposed in SPROUT's dashboard (Figure 4), which allows the requester to efficiently identify various ambiguities in the previous task design as part of a *Refine* step. The requester improves the instructions and test questions on the basis of this feedback, SPROUT compiles these into gated instructions, and the feedback loop repeats. When the current task design

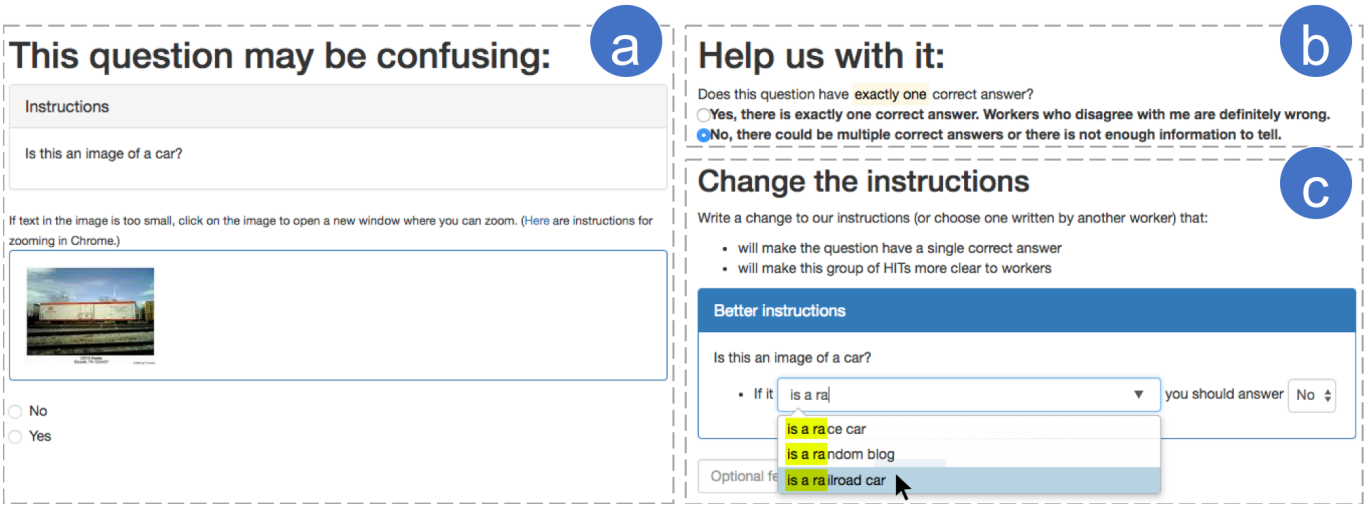


**Figure 3. The Resolve meta-worker HIT primitive, which implements the *Diagnose*, *Clarify*, and *GenTest* steps of the meta-workflow (Figure 1). A worker (a) is shown a question from the base task (here, the *Cars* task) and (b) is asked to perform a *Diagnose* step. If she decides the question is ambiguous (has multiple correct answers) given the current instructions, she then (c) performs a *Clarify* step by adding a rule to the instructions (based on how she might define the task). Workers who decide the question is unambiguous instead perform a *GenTest* step (not pictured) by creating a test question.**

no longer results in worker confusion or the requester ends the process, the final task design is run on the whole dataset.

### Finding and Characterizing Ambiguous Items

SPROUT's *Filter* step identifies possible points of confusion in a *Work* step (run on the requester's current instructions), using either indirect signals (e.g., questions with low inter-annotator agreement) or direct signals (e.g., via a feedback input on the task itself). Possibly confusing questions trigger Resolve HITs, where crowd workers *resolve* potential ambiguities and in the process generate useful metadata for organizing the dataset and creating gated instructions.

**Resolve HIT Part 1**: In the first part of the HIT (Figure 3b), a worker performs a *Diagnose* step by labeling whether the question (Figure 3a) could have multiple correct answers (is ambiguous) or has exactly one answer (is unambiguous). Depending on their response, the worker subsequently performs either a *Clarify* step or *GenTest* step, respectively, in the second part. These subsequent steps take about the same amount of work, so workers tend to perform *Diagnose* steps honestly.

The *Diagnose* step is designed to improve work quality. Our initial design omitted the *Diagnose* step, instead asking workers to perform a *Clarify* or *GenTest* step in the appropriate location of a single form. However, some workers entered *GenTest* justifications in the intended *Clarify* location. Forcing workers to make an explicit initial judgement and dynamically adding a follow-up question helps to reduce these errors.

**Resolve HIT Part 2, *Clarify* Option**: If the worker decides the question is ambiguous, SPROUT elicits a category from the worker (via the text input box in Figure 3c) by having them perform a *Clarify* step in the second part of the HIT. This step consists of adding a clarification bullet to the instructions by describing the nature of ambiguity (category) and deciding how items in that category should be labeled if *they* were the requester (*yes* or *no*). SPROUT ultimately discards worker labeling decisions (since only the requester can make the final determination); their only purpose is to make the HIT feel more natural to workers. The category input field auto-completes to categories previously written by other workers to help workers reduce redundancy and arrive at a relatively small set of categories for future review by the requester.

SPROUT's method of eliciting ambiguous categories by having workers directly suggest edits to the instructions is designed to produce a rich set of categories. Workers in our experiments often entered non-standard categories that function as rich decision boundaries, useful for defining the task acceptance criteria, e.g., workers entered "has the car as the main subject" or "has windshields and seats and wheels" which could help define acceptable car images. Simply asking workers to categorize ambiguities did not produce these types of categories.

SPROUT's *Clarify* step also aims to produce focused text to improve similarity comparisons and clustering results in the next *Organize* step of the meta-workflow. Describing an ambiguity *in the context of instructions that other workers will see* helps keep the text succinct. For example, the first two workers performing *Clarify* steps for the same question entered "should only include photographs or realistic images of birds" and "is a toy bird," and a third worker also entered "is a toy bird" (via auto-complete). These short phrases could all be included directly in the instructions. When asked to explain ambiguities without this context, workers often entered many words unrelated to the actual ambiguity.[2]

**Resolve HIT Part 2, *GenTest* Option**: Workers that decide the question is unambiguous instead perform a *GenTest* step in the second part of the HIT (alternative version of Figure 3c, not pictured), where they create a test question (for use in the gated instruction workflow) by marking the correct answer and providing an explanation. These questions are good candidates

[2]E.g., one worker wrote, "Although the image is of a bird made from legos, it is still an image of a bird. I would think that meets the criteria. However, the instructions are a bit ambiguous and don't say whether it needs to be an actual bird or one depicted in an image."

for testing workers because (1) a worker has a reason for why it is unambiguous and (2) it is likely to help filter workers who do not fully understand the instructions and initially disagreed with that worker, causing the question to be flagged.

For some questions, multiple workers indicated that an item is not confusing by performing *GenTest* steps, but submitted conflicting answers. We believe this is an important source of ambiguity, which likely happens due to differing interpretations of the same instructions. We include all such items in the set of ambiguous items and perform automatic clustering based on *GenTest* step explanations (since *Clarify* step categories are unavailable).

#### Clustering and Determining Related Items

*Organize* is the next meta-workflow step; here, SPROUT uses all worker feedback to organize confusing questions for prioritized exploration by the requester and to determine question relatedness for context. It also maintains information for suggesting test questions to the requester in the *Refine* step. Toward this end, SPROUT creates (1) a two level-hierarchy of ambiguous categories, (2) a priority order for top-level categories, and (3) similarity scores for each item pair.

SPROUT adapts the taxonomization algorithm from Cascade [6] for creating its prioritized hierarchical clusters. Proceeding from the largest categories (auto-completed instruction edits from *Clarify* steps) with the most confusions to the least, SPROUT selects a category to include at the top-level and nests all smaller categories with overlapping items as "related" categories (see Figure 4a). This also creates a natural order for top-level categories, since the more confusing categories are prioritized higher. Note that this is a soft clustering, i.e., an item can appear in multiple categories, which is appropriate for our task, since one item could be confusing for multiple reasons—all such reasons are likely valuable to the requester.[3]

To compute item-item similarity, SPROUT first creates an item embedding. It uses all the text written by workers in the Resolve HITs and takes a TF-IDF-weighted linear combination of word embeddings in that text. Since the amount of text written by workers is relatively small, pre-trained word embeddings based on a Google News Corpus [26] suffice for this task—they capture similarities between semantically related words. SPROUT creates item-item similarity scores using the cosine distance between item embeddings.

#### Requester Dashboard for Improving Task Design

The SPROUT dashboard guides requesters performing a *Refine* step of the meta-workflow to efficiently identify important categories of confusion (Figure 4a), inspect individual items to gain deeper understanding (Figure 4b), and redesign the task (Figure 4c,d) in response.

Following visualization principles of "overview first and details on demand" [32], requesters can view all the top-level categories (largest, most confusing first) in the left column (Figure 4a), and expand categories to inspect individual items as needed. Category size is indicated next to the category name, along with a visualization of the distribution of workers who have answered *yes*, *no*, or *?*. Test question labels from *GenTest* steps are denoted as *yes* and *no*, while *Clarify* step labels are denoted as *?* (SPROUT discards labels from workers editing the instructions). Individual items within each category are represented by a button marked with the item id and colored with the mean worker answer. This compact item representation is inspired by Kulesza et al.'s [21] original structured-labeling implementation. The category and item visualizations use pink, white, and green to represent *no*, *?*, and *yes* worker answers, respectively.

Requesters can view additional details about individual items in the central preview column (Figure 4b), which is accessed by clicking on an item. The top of the preview shows worker responses from the Resolve HITs issued for the item. Below the preview, a panel shows thumbnails of similar items (sorted by descending similarity). These thumbnails are adapted from the original structured labeling implementation [21] for providing context about how to label an item.

The requester's instructions editor (Figure 4c) supports example creation and rich text editing. SPROUT lets requesters format their text using the Markdown markup language, and extends that language to support referencing items as examples using Twitter mention notation (e.g., `@12` will insert a reference to item 12). A preview tab lets the requester preview a formatted version of the text, with referenced items replaced by clickable item buttons.

To make an item into a test question, the requester simply drags it to the test questions panel (Figure 4d). Test questions are used for the gated instruction workflow that ensures that a new worker has understood the task (see next subsection). Clicking on the item in that panel opens a dialog box with a form that lets the requester edit the explanation. The form also provides guidance on best practices for writing test questions [10].

SPROUT suggests improvements to the task's training regimen in the form of test-question recommendations. Each time a requester references an example item using the instructions editor, SPROUT recommends the most similar item (above a minimum similarity threshold) as a potential test question. These questions in general are good candidates for test questions because they are likely to reinforce or test understanding of the examples during the gated instruction workflow. In Figure 4d, the system has recommended an image of a boat carrying cars (item 349) since the requester had previously created an example of cars on a ferry (item 444).

As part of the overall workflow, requesters can quickly see which categories they have (and have not yet) inspected by the presence (or absence) of a checkmark to the left of the category name. SPROUT also provides a measure of the requester's overall progress toward viewing all confusing categories with a progress bar above the categories.

#### Compiling Gated Instructions

As the final part of the *Refine* step, SPROUT compiles the selected test questions into gated instructions [23] (see previous work and Figure 2 for details). SPROUT partitions the test

[3]During pilot experiments, we also tried hierarchical clustering, but found the unprioritized, hard clustering to be less useful and coherent.

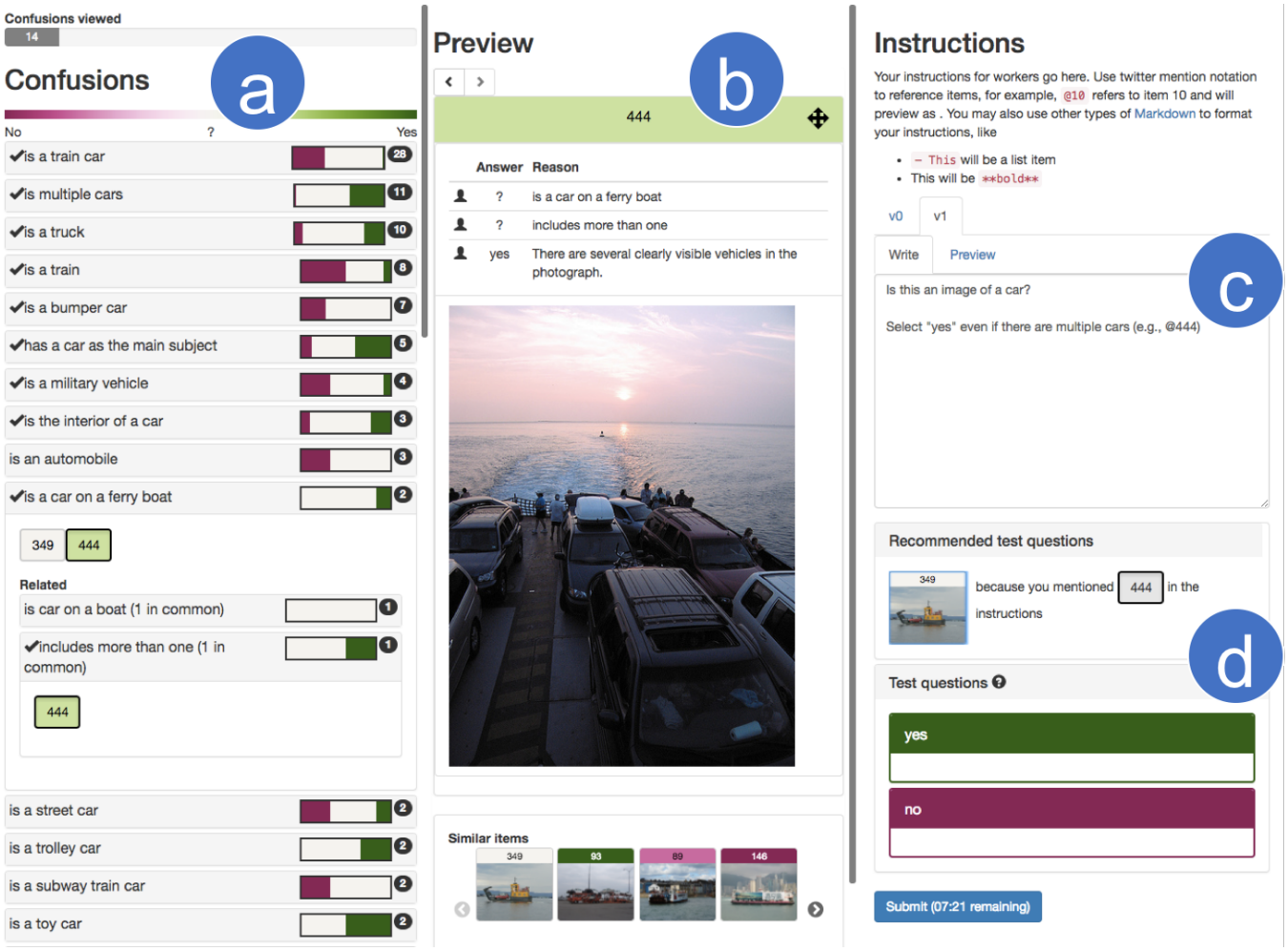


**Figure 4. The SPROUT requester interface during a *Refine* step of the meta-workflow (Figure 1) for the *Cars* task. SPROUT enables requesters to (a) drill down into categories of ambiguous items, (b) view details of items (Item 444 shown), e.g., individual worker feedback (top) and similar items (bottom), (c) edit the instructions in response, and (d) create test questions, possibly from the set of recommended test questions (SPROUT recommended item 349 because it is similar to Item 444—an example the requester provided in the instructions—and thus a good candidate for testing worker understanding).**

questions for each label into two equal-sized sets for constructing the interactive tutorial and the gating questions (ensuring similar label distributions). These sets are subject to a maximum size, which is tunable and limits the duration of gated instruction. SPROUT uses any remaining test questions as gold standard questions to ensure workers remain diligent, using, e.g., a decision-theoretic gold question insertion strategy [3].

## EXPLORATORY USER STUDY DESIGN

We conducted a user study to validate our system design and inform the design of future tools for improving task design. Our evaluation is guided by three primary research questions:

**RQ1**: How useful do requesters find SPROUT's worker-powered interface for improving task design?

**RQ2**: How helpful are SPROUT's task improvement affordances (e.g., worker-powered structured navigation and test question suggestions)?

**RQ3**: How much improved are task designs produced with SPROUT?

Our research questions seek to validate our initial hypothesis that worker feedback helps task design by measuring requester attitudes and behaviors (RQ1, RQ2) and task design quality (RQ3). While it may seem obvious that feedback can help aspects of task design like wording of instructions, it is less clear that it will help other major task design bottlenecks like discovering and defining the nuances of the task. Thus, our evaluation compares SPROUT to structured labeling [21], a strong alternative to worker feedback that embodies best practices for exploring one's dataset and identifying important edge cases, which requesters can then add to the instructions [4].

To answer these research questions, we conducted a within-subjects laboratory study that allowed us to observe requesters using both SPROUT and structured labeling on two different task types, and to survey them on the relative benefits and weaknesses of the approaches. This study design let us get feedback from requesters with varying amount of crowdsourcing expertise, and control for interface condition and task type.

Our experiment design asks requesters to refine their task designs according to their *own* understanding of the concept,

resulting in many different task definitions. We initially attempted to control for different requester concepts by fixing the concept up-front, but could find no way to communicate a fixed concept to requesters without interfering with the experiment. If one fully describes the concept to the requester up front, the requester could simply pass this description on to the workers, obviating much of the tool's benefit (finding ambiguities). We also tried having requesters interact during the experiment with an oracle that answers whether the requester has correctly labeled an item according to a fixed concept. However, in pilot studies, we found this interaction to be overly complex and unnatural. Ultimately, we decided to let requesters specify their own concepts and account for this in our analysis. Previous studies of requester behavior [30] employed a similar design.

Following the user study, we evaluated the quality of the resulting instructions to help answer RQ3. SPROUT's objective and the truest measure of instruction quality is accuracy of the data generated by workers given those instructions. Unfortunately, we could not measure data accuracy because we do not have access to ground truth labels for comparison (each requester has their own, latent concept, which is only partially expressed through the instructions). In order to approximate data quality, we measured instruction quality directly by having two crowdsourcing experts[4] rank the instructions for each task, blind to tool condition, based on the number of ambiguous categories addressed in the instructions.[5] We asked the experts to base their rankings on this criterion, since reducing ambiguities is likely to improve data quality (assuming appropriate quality control measures are employed [23]),

In the rest of this section, we describe how we created the tasks and two requester interfaces, how we recruited participants for our study, and the details of our experimental procedure.

### Labeling Tasks and Requester Interfaces

We selected two classification task types of different complexity: one image and one website (mix of text and images), inspired by prior work [4]. Since prior researchers did not release their tasks, we constructed two new equal-sized datasets:

- The *Cars* image dataset. We obtained this dataset by selecting images from all the ImageNet [16] synsets containing the word *car*, as described in prior work [4].
- The *Travel* website dataset. In order to have a dataset of sufficient size for measuring instructions, we collected a new, expanded version of the dataset created by Kulesza et al. [21] from the DMOZ directory.[6] We found that sampling all pages from the *travel* category resulted in a sufficient number of ambiguous examples that it was not necessary to sample negative examples from other categories.

[4]The experts are researchers (one author, one non-author) who have each posted over 20 different crowdsourcing tasks and authored papers on the topic. Neither expert had previously seen the instructions.

[5]The experts determined categories independently by performing open coding [8] on the instructions.

[6]We used an archived version of DMOZ [33], since the original DMOZ site is no longer active.

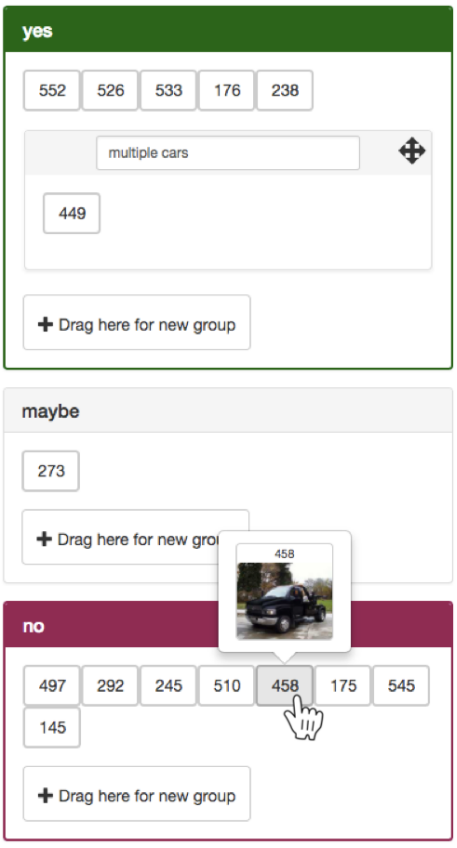


**Figure 5. Our implementation of structured labeling [21], a method for labeling new tasks, which previous researchers have used to construct instructions [4]. Structured labeling supports concept evolution (one's changing definition of the task as one explores more data) by allowing the requester to defer labeling decisions with a *maybe* label, quickly change labels for groups of items (e.g, by dragging the "multiple cars" group to a different label), and name groups for fast recall. Here, a requester is hovering over an item (Item 458 here), which displays a thumbnail preview; a requester can also click on the item to view a larger preview (not pictured) or drag the item to a different label or group.**

Our structured labeling implementation (Figure 5) operates completely independently of worker feedback and adapts key ideas from structured labeling. Requesters can label items by dragging them into *yes*, *no*, or *maybe* sections, organize items in groups within those sections, and name groups for easier recall. Figure 5 shows one requester's use of these sections during the user study. Our structured labeling implementation retains the instructions and test question editors of SPROUT (Figure 4c,d), but provides space for requesters to organize items themselves (Figure 5) in place of the panel with categories created by workers (Figure 4a). We implemented structured labeling as closely as possible to the original paper, since the authors did not release their tool or source code.

We have released the tasks, a library for generating similar tasks, and the source code for SPROUT and structured labeling for use by future researchers.[7] Our requester interface implementations are web applications built on the open-source React front-end library.

[7]https://crowdlab.cs.washington.edu/task-design.html

### Participants

We recruited 11 participants[8] to use SPROUT and structured labeling as requesters, by emailing relevant mailing lists (with IRB approval). Our participants were graduate (10) and undergraduate (1) students at a major research university. Participants ranged in age from 21 to 45 and were balanced in terms of gender (5 male, 5 female, 1 other). Eight of the participants indicated prior experience as requesters, with four reporting having launched 1–3 different tasks and four reporting having launched 4–10 different tasks. Seven of the participants indicated some prior experience as crowd workers, but no participant indicated completing more than 10 HITs. Participants were paid $25 for approximately one hour of their time.

### Procedure

We used a within-subjects experimental design. Each requester used SPROUT and structured labeling to improve the instructions for the two tasks, one per task. We assumed there was no learning effect across tasks, so we fixed the task order as *Cars* followed by *Travel*. We used a Latin square to randomize the order in which each requester encountered the interfaces. Before each task, requesters completed a tutorial and practice task (a dataset of confusing bird images) using the assigned interface in order to ensure they understood the goal and how to use the interface. After each task, requesters completed a brief survey about their experience, and after the last task, they also rated their preferred interface.

To ensure that the study completed within the allotted one hour (including tutorials and surveys), requesters were given 18 minutes per task to create an improved version of the instructions. We selected 300 items for each task as the evaluation set for the instructions, anticipating 300 to be sufficiently large such that (1) it would contain most of the common ambiguities in the task definition and (2) requesters would not be limited by the number of sample items during the experiment.[9]

Requesters were instructed to improve the initial instructions with the goal of having workers produce higher-accuracy answers that agree with the requester's concept. We used simple initial instructions for both datasets: "Is this an image of a car?" and "Is this a website about travel?" Accuracy depends on the task specification, and since each task is underspecified to start, each requester may arrive at a different target concept. Requesters who were unsure what their concept should be were prompted to pretend they were "launching a service for detecting cars" or "launching a website for travel tips." Requesters were also instructed to create at least three test questions that would ensure that workers understand their instructions.

In order to reduce the cost and variability of running the full SPROUT workflow with every requester, we pre-collected worker (and meta-worker) answers for all questions in the evaluation sets by seeding SPROUT with the initial versions of the instructions. These answers were replayed by SPROUT during experiment sessions. We deployed the Resolve HIT to three workers for each of the 300 items in each task, paying $0.05 per image HIT and $0.07 per website HIT (based on a wage of $8 / hour calculated from pilot deployment timings). We limited participation to U.S. workers who had completed at least 100 HITs with an approval rate of at least 95%.

[8]We excluded one participant who was not directly affiliated with the university mailing lists we advertised to, and who had difficulty understanding the objective and procedure of the experiment.

[9]Experimentation during pilot studies showed that 18 minutes was a reasonable upper bound and 300 items was sufficiently large.

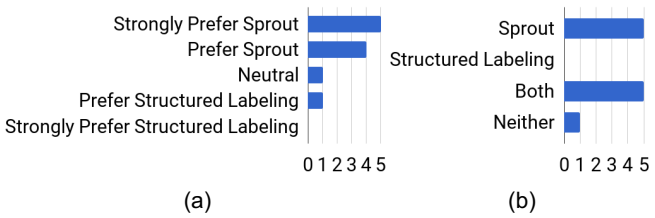


**Figure 6. Requester responses to the questions (a) "Which interface did you prefer for creating instructions?" and (b) "Which interface would you use to create instructions in the future?". Requesters in our study overall preferred SPROUT with worker feedback rather than in structured labeling, but about half still saw uses for both interfaces.**

## RESULTS

### RQ1: Usefulness of Worker Feedback

Figure 6a shows that requesters overall preferred SPROUT over structured labeling. P1 summarized the high-level benefits of worker feedback with SPROUT: "[c]ategorizing the inputs, showing me the cases where there was confusion, etc., made it SUPER easy to identify cases that needed clarification." In contrast, structured labeling didn't provide "a sense of what the data looked like as a whole." Another requester (P4) expressed dismay after switching from SPROUT to structured labeling: "It sucks that you have to start from a completely blank slate. [SPROUT] gave you some more support."

While most requesters preferred SPROUT, about half indicated that they would use both interfaces in the future (Figure 6b), indicating that worker feedback is not always desired. P3 wrote "Before I had any crowd data, I liked [structured labeling] because it let me try doing the task myself...But after I had some preliminary labeled data, I would like to use [SPROUT] to see what kinds of things people were confused about." P2 wanted to use structured labeling to start to avoid being biased by workers when deciding on her task concept. P11 expressed mixed feelings about structured labeling; she liked that it "forced [her] to personally think through what was in the image and what [she] was looking for," but also said that "it was...pretty time consuming to create the different categories."

Requesters did not find the organizational aspects of structured labeling particularly useful and rarely created groups, likely because the instructions editor itself serves as an organizational tool that lets requesters describe categories with example items. Usage might change with more extended interaction; P1 noted "[groups] might be [useful] if [he] were iterating multiple times and wanted to come back...in the future."

The one requester who indicated he would use neither interface in the future (P1) did so because he prefers crafting examples by hand as a domain expert in natural language processing, explaining "[a]lthough SPROUT does help with quickly identifying the confusing cases...to make the instructions concise I typically have to come up with my own examples anyway (so I can reuse the same example sentence for a lot of examples)."

### RQ2: Usefulness of SPROUT's Affordances

Requesters found the recommended test questions particularly useful. One requester (P2), an NLP researcher, wrote "The most helpful features...is (sic) the automatically suggested test questions. The similarity metrics seems (sic) to be working great and the suggested items are great for testing the points I emphasized in the instructions." Overall, a significant fraction of test questions created by requesters were previously recommended by SPROUT (on average, 29% and 20% for the *Cars* and *Travel* tasks, respectively). Requesters reported them useful in surveys, and several created test questions almost exclusively based on recommendations (4/5 test questions by P1 on the *Cars* task and 4/4 test questions by P2 on the *Travel* task). Still, semantically similar items are not always the best test questions; P11 complained, "if I used a ferry with cars on it as an example, it'd just return a boat as a suggestion."

Requesters used the similar items panel infrequently, perhaps due to lack of familiarity with the interface or because SPROUT already recommends the most similar item to each example in the instructions as a test question.

While most requesters looked at individual items first, P9 scanned the text descriptions of the categories and began to edit the instruction text without inspecting items. This type of rapid instruction improvement was made possible by the organized presentation of worker feedback. We did not instruct requesters about such strategies, and other requesters may have learned to use SPROUT more effectively with instruction.

### RQ3: Impact on Task Design Quality

On average, requesters using SPROUT wrote longer instructions, cited more examples, and were rated higher by experts.[10]

Requesters using SPROUT on average wrote longer instructions ($\mu = 1672$ vs. 1110 characters) and cited nearly twice as many examples on the *Travel* task compared to structured labeling ($\mu = 4.6$ vs. 2.6 examples). These results are weakly significant based on a two-sided Welch's t-test ($t = 1.89$, $p = 0.10$; $t = 1.77$, $p = 0.11$). There was no significant difference on the *Cars* task ($t = -0.30$, $p = 0.78$; $t = -0.09$, $p = 0.93$).

Two crowdsourcing experts independently ranked the instructions for each task into five quality buckets (valued 1 to 5), assigning higher values to instructions that mentioned more categories of ambiguous items. Both experts ranked instructions produced using SPROUT higher on the *Cars* task compared to structured labeling ($\mu = 4.0$ vs. 2.8; $\mu = 3.2$ vs. 2.0). These results are weakly significant based on a two-sided Welch's t-test ($t = 1.78$, $p = 0.11$; $t = 1.88$, $p = 0.11$). There was no significant difference on the *Travel* task for either expert rating ($t = 0.0$, $p = 1.0$; $t = 0.0$, $p = 1.0$).

Together, these results suggest that SPROUT helps create more comprehensive instructions. Detailed instructions can elicit higher quality data from workers when combined with proper training and screening methods like gated instructions [23].

[10]Larger samples are needed to establish statistical significance. We excluded P5 from analysis of the *Travel* task, since she was unable to complete the task due to difficulty finding a motivation for the task.

## DESIGN IMPLICATIONS FOR TASK DESIGN

### How to make exploration frictionless?

It is essential that future task design tools help requesters view items with minimal overhead. In structured labeling, even dragging items into *yes* / *no* / *maybe* sections was inefficient for some requesters, who found it faster to scroll through the carousel of item thumbnails. P1 complained that even this carousel had too much friction and the images were too small. A more efficient view might have been a vertical scroll (used by websites like Instagram) with an option to resize images, though factors such as scrolling direction and number of items per page can have subtle effects on performance [18, 19]. One requester (P8) seemed to experience similar friction with SPROUT, repurposing the similar items (bottom of Figure 4b) for quickly retrieving new items (not just similar ones).

SPROUT was designed with the idea that one must explore items before writing instructions. However, one can also view exploration in service of the ultimate goal of creating instructions. From this viewpoint, P10 felt that the instructions editor would be more natural on the left. Other requesters felt that SPROUT could have benefited from more "visual hierarchy" (P8) and "linear process" (P3).

### How much information to show?

Another design decision to consider is how much and when to show information about worker confusions to the requester. P5 felt overwhelmed by the number of categories displayed and began clicking through categories without regard to their prioritization. In contrast, P9 benefited from having all categories displayed initially, as his strategy was to read the descriptions, edit the instructions, and only look at items periodically. Providing the right amount of support for a diverse set of users is a challenging problem for mixed-initiative systems. One possibility is to make the frictionless vertical scroll described above the default mode, and enable the requester to use additional features from SPROUT and structured labeling on demand as they learn what tools most benefit their personal workflow.

### How much initiative to take?

Another possibility is to create an adaptive version of SPROUT that shows the requester a sequence of only the most important and diverse categories, taking into account the set of items the requester has viewed and the instructions she has written up to that point. More knowledge about the space of items considered by the requester so far could also enable smarter suggestions for improvements to the instructions.

While our tool supports test question creation by the requester, it is unclear whether the requester need be the one to do so, once she has written a clear set of instructions with examples. Indeed, our high-level vision (Figure 1) is that workers (or the system) should be able to determine when the requester's input is needed. This suggests that other tedious task components, such as gold questions or task advertisements, might be created automatically by meta-workers, saving requester time. And while P1 wanted SPROUT to help him ensure the distribution of test question labels matched the overall distribution of labels—as recommended by CrowdFlower for gold questions [10]—such tools may not be necessary either.

**How to balance self-organization and worker support?**
While the instructions themselves can be used for organizational purposes, additional support for self-organization in SPROUT could have been helpful. For instance, P9 had trouble recalling and finding a category of items he had previously read (but not opened, so it did not have a check mark). P4 on the other hand wanted to incorporate some of the intelligence of SPROUT into structured labeling by automatically narrowing the set of items classified as *maybe* using structured labeling (e.g., using item vector embeddings), when items in that section are covered by the instructions. Finding a middle ground and making transparent what the workers have done vs. what the requester has done is challenging but could pay dividends.

**How much of the workflow to support?**
Even if one starts out with a binary-classification problem, one can realize down the line that the task might benefit from decomposition or structured output of a different kind. Two of our requesters familiar with crowdsourcing for NLP (P1, P2) wanted to change the task interface, for instance by changing the answer labels, or to break the task into smaller subtasks "since it is easier to write instructions for small tasks." Supporting these aspects of task design are a worthy goal and could benefit from SPROUT's feedback loop; we chose to focus on binary labeling tasks since (a) they are the most common crowdsourcing task type [15] and (b) we wanted to constrain the design space to avoid some of the pitfalls of previous work that tackled broader problems [22].

In addition to supporting more aspects of task design, we believe that future versions of SPROUT could naturally support additional types of tasks. Labeling tasks with more than two answers are possible with minor interface changes, and more generally, we envision SPROUT being useful for any task with many different instances (questions) that share a common design (so that improvement benefits many questions) and have a correct answer (so workers can be tested before beginning the task), for example information-seeking tasks [30].

**How to scaffold the process of learning to design tasks?**
Our requesters were largely unpracticed at writing high-quality crowdsourcing instructions. There is opportunity to incorporate tools for helping requesters—such as guiding them toward effective strategies—into SPROUT. P4 mentioned it would helpful to have templates for common things to say to workers like "use your best judgement." P1 thought better support for task criteria labels would be useful for overall consistency. We agree that these would be great to add, but determining a single set of best practices is difficult. It may also be possible to train crowd workers instead to hone the presentation of the task.

## LIMITATIONS

Several additional types of evaluation would strengthen our findings. Future studies should strive to measure both data accuracy and worker satisfaction resulting from task improvement. More studies are also needed to investigate what happens when instructions become very complex, and to demonstrate that our findings generalize to many types of requesters.

While our study design allowed for controlled observation of requester behavior, we also encountered several experimental challenges. Several requesters experienced difficulty deciding on a motivating concept to help them make labeling decisions, causing large delays (P5) or concept changes just to simplify the task (P11); providing tools to requesters solving their own problems may improve motivation. Future studies could also seek to better control for the amount of time requesters spend on each task (some requesters ended the task early), or the style of instructions (e.g., by providing more requester training). Finally, studying multiple task types was informative but decreased statistical power; future studies could try other experimental designs (e.g., with task type as a blocking factor).

## CONCLUSIONS

To achieve high-quality output from crowdsourcing, one requires diligent workers working on well-designed, and clearly-explained tasks. While there are many papers on identifying diligent workers and substantial research on patterns for task decomposition, our work is perhaps the first tool that helps requesters design effective instructions. Instructions may be less glamorous than some other aspects of crowdourcing, but they have been shown to be deeply important [23].

Furthermore, SPROUT uses a novel method to aid instruction design and debugging: having the crowd evaluate the current design on a sample of data, identifying confusing questions based on disagreement and worker diagnoses, clustering confusion categories based on worker instruction edits, and showing those in an organized and prioritized manner so that a requester can quickly learn the various nuances of their task and its current flaws. SPROUT further aids a requester by providing a natural interface for improving instructions with embedded illustrative examples and recommending test questions for a gated instruction workflow that ensures worker understanding.

Nearly all the requesters in our user study (with varying amounts of crowdsourcing expertise) preferred to use SPROUT (which has worker feedback) over structured labeling, a natural baseline that supports requesters learning about their task themselves rather than through worker feedback. Some requesters felt that structured labeling is a good interface for creating the first set of instructions, but overall they preferred the full power of SPROUT, which makes effective task design more convenient. On average, instructions produced using SPROUT were longer, cited more examples, and were rated higher by multiple crowdsourcing experts. This user study also led to our set of design implications for future task design, and we have released our source code and web-based implementations for further use by requesters and researchers.

In the future, we plan to use the crowd to improve other aspects of task and workflow design, such as task decomposition, and to support task design beyond labeling tasks. For example, we envision crowd workers retrieving task details from requesters as needed and collaboratively developing even better designs with minimal requester effort. We encourage other researchers to continue to explore new ways to leverage and develop worker task design skills, and to build systems that mediate worker-requester communications.

## ACKNOWLEDGEMENTS

We thank Gagan Bansal, Quanze Chen, and Christopher Lin, for their early feedback, and Eytan Adar, Danielle Bragg, Ravi Karkar, Tongshuang Wu, and the anonymous reviewers for their feedback on paper drafts. We also thank the requesters and workers who participated in our study, as well as the external crowdsourcing expert who judged the instructions. This work was supported in part by NSF grant IIS-1420667, ONR grants N00014-15-1-2774 and N00014-18-1-2193, the WRF/Cable Professorship, support from Google, a Bloomberg award, an IBM SUR award, and a Visvesvaraya faculty award by the Government of India to the second author.

## REFERENCES

1. Harini Alagarai Sampath, Rajeev Rajeshuni, and Bipin Indurkhya. 2014. Cognitively Inspired Task Design to Improve User Performance on Crowdsourcing Platforms. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '14)*. ACM, New York, NY, USA, 3665–3674. DOI: http://dx.doi.org/10.1145/2556288.2557155
2. Omar Alonso and Stefano Mizzaro. 2012. Using crowdsourcing for TREC relevance assessment. *Information Processing and Management* 48, 6 (2012), 1053–1066. DOI: http://dx.doi.org/10.1016/j.ipm.2012.01.004
3. Jonathan Bragg, Mausam, and Daniel S. Weld. 2016. Optimal Testing for Crowd Workers. In *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems (AAMAS '16)*. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 966–974. http://dl.acm.org/citation.cfm?id=2937029.2937066
4. Joseph Chee Chang, Saleema Amershi, and Ece Kamar. 2017. Revolt: Collaborative Crowdsourcing for Labeling Machine Learning Datasets. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI '17)*. 2334–2346. DOI: http://dx.doi.org/10.1145/3025453.3026044
5. Justin Cheng, Jaime Teevan, and Michael S. Bernstein. 2015. Measuring Crowdsourcing Effort with Error-Time Curves. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems (CHI '15)*. ACM, New York, NY, USA, 1365–1374. DOI: http://dx.doi.org/10.1145/2702123.2702145
6. Lydia B. Chilton, Greg Little, Darren Edge, Daniel S. Weld, and James A. Landay. 2013. Cascade: Crowdsourcing Taxonomy Creation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '13)*. ACM, New York, NY, USA, 1999–2008. DOI: http://dx.doi.org/10.1145/2470654.2466265
7. Stanford Crowd Research Collective. The Daemo Crowdsourcing Marketplace. In *CSCW '17*.
8. Juliet Corbin and Anselm Strauss. 2014. *Basics of Qualitative Research*. SAGE Publications, Inc.
9. Crowdflower. 2017a. Ideal Jobs for Crowdsourcing. (2017). https://success.crowdflower.com/hc/en-us/articles/202703295-Ideal-Jobs-for-Crowdsourcing Downloaded on 9/17/17.
10. Crowdflower. 2017b. Test Question Best Practices. (2017). https://success.crowdflower.com/hc/en-us/articles/213078963-Test-Question-Best-Practices Downloaded on 9/17/17.
11. Steven Dow, Anand Kulkarni, Scott Klemmer, and Björn Hartmann. 2012. Shepherding the Crowd Yields Better Work. In *Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work (CSCW '12)*. ACM, New York, NY, USA, 1013–1022. DOI: http://dx.doi.org/10.1145/2145204.2145355
12. Ujwal Gadiraju, Yang Jie, and Alessandro Bozzon. 2017. Clarity is a Worthwhile Quality - On the Role of Task Clarity in Microtask Crowdsourcing. In *Proceedings of the 28th ACM Conference on Hypertext and Social Media (HT '17)*. 5–14. DOI: http://dx.doi.org/10.1145/3078714.3078715
13. Philipp Gutheim and Björn Hartmann. 2012. *Fantasktic : Improving Quality of Results for Novice Crowdsourcing Users*. Master's thesis. University of California, Berkeley. http://www.eecs.berkeley.edu/Pubs/TechRpts/2012/EECS-2012-112.html
14. David J. Hauser and Norbert Schwarz. 2016. Attentive Turkers: MTurk participants perform better on online attention checks than do subject pool participants. *Behavior Research Methods* 48, 1 (2016), 400–407. DOI: http://dx.doi.org/10.3758/s13428-015-0578-z
15. Ayush Jain, Akash Das Sarma, Aditya Parameswaran, and Jennifer Widom. 2017. Understanding Workers, Developing Effective Tasks, and Enhancing Marketplace Dynamics: A Study of a Large Crowdsourcing Marketplace. In *43rd International Conference on Very Large Data Bases (VLDB)*. DOI: http://dx.doi.org/10.14778/2735471.2735474
16. Jia Deng, Wei Dong, R. Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 248–255. DOI: http://dx.doi.org/10.1109/CVPRW.2009.5206848
17. Sanjay Kairam and Jeffrey Heer. 2016. Parting Crowds: Characterizing Divergent Interpretations in Crowdsourced Annotation Tasks. In *Proceedings of the 19th ACM Conference on Computer-Supported Cooperative Work & Social Computing (CSCW '16)*. ACM, New York, NY, USA, 1637–1648. DOI: http://dx.doi.org/10.1145/2818048.2820016
18. Diane Kelly and Leif Azzopardi. 2015. How Many Results Per Page?: A Study of SERP Size, Search Behavior and User Experience. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '15)*. ACM, New York, NY, USA, 183–192. DOI: http://dx.doi.org/10.1145/2766462.2767732

19. Jaewon Kim, Paul Thomas, Ramesh Sankaranarayana, Tom Gedeon, and Hwan-Jin Yoon. 2016. Pagination Versus Scrolling in Mobile Web Search. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management (CIKM '16)*. ACM, New York, NY, USA, 751–760. DOI: http://dx.doi.org/10.1145/2983323.2983720

20. Aniket Kittur, Susheel Khamkar, Paul André, and Robert Kraut. 2012. CrowdWeaver: Visually Managing Complex Crowd Work. In *Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work (CSCW '12)*. ACM, New York, NY, USA, 1033–1036. DOI:http://dx.doi.org/10.1145/2145204.2145357

21. Todd Kulesza, Saleema Amershi, Rich Caruana, Danyel Fisher, and Denis Charles. 2014. Structured Labeling for Facilitating Concept Evolution in Machine Learning. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '14)*. ACM, New York, NY, USA, 3075–3084. DOI: http://dx.doi.org/10.1145/2556288.2557238

22. Anand Kulkarni, Matthew Can, and Björn Hartmann. 2012. Collaboratively Crowdsourcing Workflows with Turkomatic. In *Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work (CSCW '12)*. ACM, New York, NY, USA, 1003–1012. DOI: http://dx.doi.org/10.1145/2145204.2145354

23. Angli Liu, Stephen Soderland, Jonathan Bragg, Christopher H. Lin, Xiao Ling, and Daniel S. Weld. 2016. Effective Crowd Annotation for Relation Extraction. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT '16)*. The Association for Computational Linguistics, 897–906. http://aclweb.org/anthology/N/N16/N16-1104.pdf

24. V. K. Chaithanya Manam and Alexander J. Quinn. 2018. WingIt: Efficient Refinement of Unclear Task Instructions. In *Proceedings of the Sixth AAAI Conference on Human Computation and Crowdsourcing (HCOMP '18)*. AAAI Press, 108–116. https://aaai.org/ocs/index.php/HCOMP/HCOMP18/paper/view/17931

25. Brian McInnis, Dan Cosley, Chaebong Nam, and Gilly Leshed. 2016. Taking a HIT: Designing Around Rejection, Mistrust, Risk, and Workers' Experiences in Amazon Mechanical Turk. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI '16)*. ACM, New York, NY, USA, 2271–2282. DOI: http://dx.doi.org/10.1145/2858036.2858539

26. Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *NIPS*. 1–9. DOI: http://dx.doi.org/10.1162/jmlr.2003.3.4-5.951

27. Tanushree Mitra, C.J. Hutto, and Eric Gilbert. 2015. Comparing Person- and Process-centric Strategies for Obtaining Quality Data on Amazon Mechanical Turk. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems (CHI '15)*. ACM, New York, NY, USA, 1345–1354. DOI: http://dx.doi.org/10.1145/2702123.2702553

28. David Oleson, Alexander Sorokin, Greg P. Laughlin, Vaughn Hester, John Le, and Lukas Biewald. 2011. Programmatic Gold: Targeted and Scalable Quality Assurance in Crowdsourcing. In *Human Computation, Papers from the 2011 AAAI Workshop*. AAAI. http://www.aaai.org/ocs/index.php/WS/AAAIW11/paper/view/3995

29. Daniel M. Oppenheimer, Tom Meyvis, and Nicolas Davidenko. 2009. Instructional manipulation checks: Detecting satisficing to increase statistical power. *Journal of Experimental Social Psychology* 45, 4 (2009), 867–872. DOI:http://dx.doi.org/10.1016/j.jesp.2009.03.009

30. Alexandra Papoutsaki, Hua Guo, Danae Metaxa-Kakavouli, Connor Gramazio, Jeff Rasley, Wenting Xie, Guan Wang, and Jeff Huang. 2015. Crowdsourcing from Scratch: A Pragmatic Experiment in Data Collection by Novice Requesters. In *Proceedings of the Third AAAI Conference on Human Computation and Crowdsourcing (HCOMP '15)*. AAAI Press, 140–149. http://www.aaai.org/ocs/index.php/HCOMP/HCOMP15/paper/view/11582

31. Jeffrey Rzeszotarski and Aniket Kittur. 2012. CrowdScape: Interactively Visualizing User Behavior and Output. In *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology (UIST '12)*. ACM, New York, NY, USA, 55–62. DOI: http://dx.doi.org/10.1145/2380116.2380125

32. Ben Shneiderman. 1996. The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations. In *Proceedings of the 1996 IEEE Symposium on Visual Languages (VL '96)*. IEEE Computer Society, Washington, DC, USA, 336–. http://dl.acm.org/citation.cfm?id=832277.834354

33. Gaurav Sood. 2016. Parsed DMOZ data. (2016). DOI: http://dx.doi.org/10.7910/DVN/OMV93V

34. Daniel S. Weld, Mausam, Christopher H. Lin, and Jonathan Bragg. 2015. Artificial Intelligence and Collective Intelligence. In *Handbook of Collective Intelligence*, Thomas W. Malone and Michael S. Bernstein (Eds.). The MIT Press.

35. Meng-Han Wu and Alexander J. Quinn. 2017. Confusing the Crowd: Task Instruction Quality on Amazon Mechanical Turk. In *Proceedings of the Fifth AAAI Conference on Human Computation and Crowdsourcing (HCOMP '17)*. AAAI Press, 206–215. https://aaai.org/ocs/index.php/HCOMP/HCOMP17/paper/view/15943