To Stop or Not to Stop? A Case Study of an Early intervention for Data Collection

Yu-Chun Yen, Hidy Kong, Jingning Zhang, Yu Wu, and Qian Cheng
Department of Computer Science
University of Illinois, at Urbana-Champaign
{yyen4, hkong6, jzhng117, yuwu4, qcheng4} @illinois.edu

ABSTRACT
Deciding the sample size of a qualitative survey is a critical issue in crowdsourcing technology. Insufficient feedback might result in the loss of diversity among possible answers while too much feedback results in a waste of time and money. In this paper, we address the problem in paid crowdsourcing technology. We examine the data saturation pattern over time when collecting feedback about a Kickstarter project page. Our main contribution is twofold. First, we built an easy-to-use website to let project creators post a simplified version of their Kickstarter page in Amazon Mechanical Turk. Secondly, we designed an interactive visualization page allowing project creators to check saturation level according to the current feedback pool. Our system takes the advantage of its generalized framework that could be applied to any feedback systems that cannot provide the guidelines for deciding the appropriate sample size.

Categories and Subject Descriptors
D. H.5.3 [Information Interface and Presentation]: Group and Organization Interfaces -- Evaluation/methodology

General Terms
Crowdsourcing, Data saturation

Keywords
Keywords are your own designated keywords.

1. INTRODUCTION
Crowdsourcing has become a near-ubiquitous technology allowing a requester to have scalable and diverse feedbacks from crowd workers in an efficient way. Various platforms are designed to leverage crowdsourcing technology for collecting subjective feedback. FeedbackArmy[1] claims to receive desired quantity of usability test results for website creators in two minutes. UsabilityHub[2] allows users to upload their designs or mockups and generate the survey page for them so that they can get critiques from real people. Some platforms ask requester to find critics by inviting their friends though social network site such as Facebook, while others find anonymous paid workers from crowdsourcing platform such as Amazon Mechanical Turk. Amazon Mechanical Turk [3] is a famous platform that provides a marketplace for the requesters to publish tasks and for workers to find work opportunities. Beyond spreading tedious works such as image labeling or text translation, requesters are now expecting to collect diverse feedback from crowd workers. Due to the divergent background of different workers, requesters have a chance of acquiring popular impression from a reasonable size of crowd workers.

The number of feedback is a crucial issue when requesters solicit for subjective feedback. Too little feedback would result in missing a portion of viewpoints from general public while too much feedback results in a waste of time and money. However, guidelines for determining the sample sizes for qualitative survey are virtually nonexistent. In current crowdsourcing platforms, systems try to alleviate such problems by asking requesters to decide the number of feedback by themselves. However, users do not have a good sense of what the appropriate number of feedback is. Our first experiment shows that the variance of desired number of subjective feedback among different people is incredibly large, which confirms the difficulty of determining the number of feedback. Most people claim that they want to get as much feedback as possible to collect all the possible ideas generated by the public. However, the more feedback the requesters collect from paid workers, the more money they have to pay. Even for those requesters who invite their friends for free answers, more data requires a longer data collection time. To overcome this problem, we designed a novel system to minimize the number of feedback while acquiring representative results from the crowd workers.

Minimizing the number of feedback is a popular issue in paid crowdsourcing technology. Requesters are not pleased when they pay for undesired feedback. Moreover, meaningless feedback disturbs data analysis by adding noise to the result. Detecting the reliability of individual feedback is one approach to reduce undesired expense [4][5]. Researchers have investigated various approaches to avoid spammer so that requesters can reject meaningless feedbacks. In addition to passively removing undesired data, actively stopping the collecting of new data is another approach to minimize the payment. Guest G. [6] *et al* present the idea of data saturation when determining the sample size for qualitative interviews. They define the concept of "saturation" as the point at which no new information or themes are observed in the data. Similarly, we define "data saturation" as the point at which new feedback provides no additional information compared to earlier feedback. Our second experiments show that, for some survey questions, the point of saturation occurs before the end of data collection, indicating that the data that comes after the saturation point wastes time and money to some degree.

We observed the pattern of data saturation over time by studying the feedback of Kickstarter projects collected from Amazon Mechanical Turk. We put a simplified Kickstarter project page along with survey questions on Mechanical Turk. Workers who selected the HIT had to answer three questions based on the project page provided in the HIT. The main contribution of this work is four-fold. First, we designed an interface for a Kickstarter project creator to easily post his project page on Amazon Mechanical Turk. Second, we visualized the data saturation pattern

for project creator in multiple viewpoints. Third, the saturation analysis not only provides the degree of saturation at each point but also gives a summary of feedback among crowd workers. Finally, our system can be applied to any feedback system, particularly those with a paid system. Requesters can make a decision to stop collecting new data early once they feel satisfied by existing data.

To the best of our knowledge, there has not been prior research examining data saturation throughout data collection process using crowdsourcing technology. We therefore designed our study to answer three research questions. *RQ1* How do people decide the number of subjective feedback when they collect data from MTurk? *RQ2a*: Is there a saturation pattern on sequential feedback data? *RQ2b*: Does the user-defined number match the point of saturation? *RQ3*: Can we have a visualization that helps the users decide when to stop?

This paper is organized as follows. We start with the description of two experiment settings used to answer proposed research questions. We then present a system that automatically generates the simplified Kickstarter page and posts it on MTurk to receive feedback from workers. A case study on a manual coffeemaker Kickstarter project is presented. The saturation pattern on collected feedback are visualized in multiple viewpoints and shown in the result. Finally, the paper concludes with an analysis of our results, and some discussion of future work.

2. RELATED WORK

In recent years, crowdfunding platforms such as Kickstarter have become more and more popular. At the first glance, this grassroots approach of fundraising may sound nice and easy. In reality, conducting a successful campaign requires a lot of skill and effort. People spend thousands of dollars in preparation for advertising material. Even after the campaign starts, people still need to spend lots of time interacting with supporters. However, despite all these money and effort, half of the campaigns still fail. In order to solve this problem, researchers have tried to use various ways to predict the success rate of a project. A common approach is to use machine learning algorithms. By training on past successful projects, the algorithm can give a decent estimate of the success rate. Moreover, the trained model also sheds light upon how people should organize their project page and phrase the texts [16][17]. Other researchers have taken the social approach. By closely monitoring the reaction from social media right after the campaign launched, researchers can roughly tell whether the project can succeed or not. [18]

Ever since Amazon Mechanical Turk platform launched, researchers began to notice the problem of low work quality. Due to the nature of crowdsourcing, workers come from a wide range of demographic groups equipped with various levels of skills and motivations. Even with the same worker qualification limit, the content generated from two workers can differ drastically. How to weed out the less useful inputs and extract meaningful information from the large pool of worker feedback is a question researchers have been trying to answer for years. One of the most common method people have used is the majority rule [12]. The answer provided by the most workers will be taken as the most important answer. However, this approach is far from perfect. Researchers have tried to improve this method in a few different ways. Crowd workers have different levels of expertise and sometimes the tasks may also have various level of difficulty. Researchers have used probabilistic approaches to take these variance into consideration and outperform the traditional majority vote heuristic [5]. In addition, since the majority vote method usually requires a certain level of redundancy to confirm answers, this approach may incur unnecessary high cost by collecting too many similar feedbacks. Researchers have tried to take algorithmic approach to minimize the total cost of the tasks [11]. Others have been using crowd workers to detect redundant information [12].

One way to efficiently present redundant information is clustering. By binding similar ideas into clusters, we can reduce a large set of opinions into a few representative ideas. Clustering appears to be a natural solution for viewing crowdsourcing feedbacks, which usually come in large quantities and with high level of redundancy. Most clustering algorithms are based on some sort of similarity measures. The measure choice usually has a heavy influence over the formation of the clusters [13]. Traditionally, people measured the similarity between pairs through some feature comparisons. The more features a pair shares, the more similar the pair is. Researchers have also proposed using other hidden pattern to measure similarity [14]. If a pair of items always acts in the same way, we may say they are similar to each other even though they share few features in common. Other researchers have taken a more straightforward approach of using humans to evaluate similarity between items [15]. All these methods help to generate a concise view of the crowd workers' opinions.

3. EXPERIMENT

In this section, we first study the degree of variance on deciding the number of HITs in MTurk. Second, we publish a Kickstarter project page along with 5 survey questions. We will examine the saturation pattern within collected data.

3.1 Decision on the number of HITs

For the first research question, we conducted a scenario-based survey through MTurk. The instruction for the task was as follows.

Kickstarter is a new way to fund creative projects. (You can learn more about it here if you're not familiar with it.https://www.kickstarter.com/hello?ref=footer) ***Imagine you are launching a waterproof keyboard project on Kickstarter****. Now you want to ask questions on MTurk so that you can get the general impression of the project from the crowd.*

We then asked them to decide the number of feedbacks that they thought would be sufficient for a given question. We collect 30 HITs for each question from MTurk with the qualification restriction of workers having greater than 95% acceptance rate. Question in each HIT is selected from the following list:

- *Provide one scenario in which this product would be useful. Please describe the scenario in detail.*
- *In what situation do you think this product would not be a good product?*
- *Add a feature that will convince you to buy the product?*
- *Which feature in the Kickstarter page do you think the most informative to attract people's attention.*

There are two assumptions to support this experiment. First, workers are the actual people who give answers. We assume they are the people best known the quality of individual answer and then estimate the number of answers they would be satisfied by receiving such answers. Second, imagine being a Kickstarter project creator does not need expert knowledge than asking workers to imagine themselves as designers. Workers are at some degree qualified to answer such questions.

3.2 Feedback on Kickstarter project

To test whether the saturation point occurs before the end of data collection process, we posted a Kickstarter project about a manual

coffeemaker product on Mechanical Turk and asked the questions provided in the Section 3.1. We collected 30 HITs for the coffeemaker project, and will regard answers given for the same question as a feedback pool. Therefore, for each question, we got a feedback pool of size 30. We will further examine the saturation pattern within each feedback pool.

For the last questions, we aimed to extract the most informative features in a Kickstarter page. According to the characteristic of working style on MTurk, a task should be a micro-task. Workers are not expected to pay attention on an individual HIT for a long time. Therefore, reading the whole project page would break the rule of micro-task and might decrease the quality of collected feedbacks. In addition, since we are only looking for the general impression rather than a holistic judgment on the product, it was not necessary for workers to read the whole page.

The first three questions were asked to collect diverse feedback from crowd workers. We published these 30 HITs at the same time. For each feedback pool, we further apply our algorithm to detect the saturation point independently.

4. IMPLEMENTATION

4.1 Workflow of Proposed System

Our proposed system is constructed with two functionalities: HIT posting and feedback visualization. The HIT posting functionality allows requesters to post their ideal HITs on Amazon Mechanical Turk through our user interface with their unique Amazon Web Service (AWS) credentials. After successfully posting their HITs, the requesters can track their HITs using the identity number the system provides them. The feedback collection functionality provides a dynamic HITs information tracking and results in a visualization for the users to decide whether they should stop collecting the HITs for the batch or not. The general workflow of the system is shown in Figure 1.

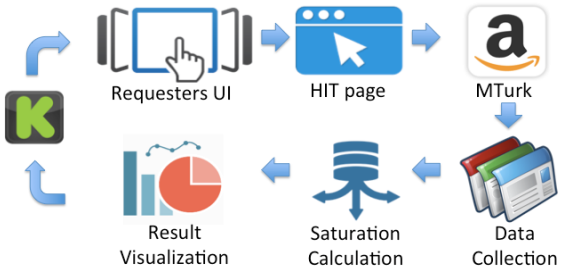

Figure 1 Workflow of proposed system

The HIT posting page was developed with boto, a python interface supporting Amazon Web Service APIs. In the boto.mturk package, APIs are provided to manage activities on Amazon Mechanical Turk, including connection, HIT layout design, HIT creating and tracking, price managing and notification. Several major APIs were applied to implement our system:

connection.MTurkConnection() allows providers to connect to Amazon Mechanical Turk with their AWS access keys. This is one of the essential functions for requesters to communicate with Mturk.

create_hit() is a key function that publishes HITs on Amazon Mturk, either on the real Mturk or on SandBox. We created a survey HIT file that stored the questions, URLs and control parameters. Then we were able to decide to publish the HITs on SandBox or real Mturk using control flags.

get_assignment() is the function that obtains HIT completion information in real time. It returns the answers for each question posted, completion time, worker identity information and so forth. Calling this function periodically helped us update the dynamic feedback.

Posting HITs

In the posting HITs functionality, AJAX technology is applied for the front-end development, while PHP is used to pass variables and call system functions at the server-end.

The design of the user interface is shown in Figure 2. On this page, we allow the requester to post surveys for his Kickstarter project by simply copying the URL of the original project page. The requester also has to enter the HIT title, description, and keywords that will be displayed on the created HIT page. The requester can set control parameters about how the HIT will be published by entering the reward amount, number of HITs, time allotted, expiration date, and approval delay, as he would if he were to post directly on Amazon MTurk.

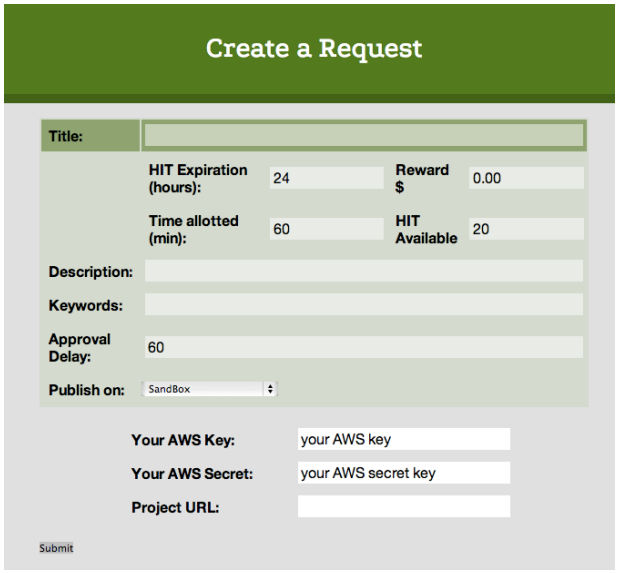

Figure 2 User interface for requesters to post their HITs.

The Kickstarter project page is simplified by removing redundant and less informative sections from the original page. We first crawl the project page by passing the URL provided the worker to Scrapy. Then we parse the HTML document and save the necessary information such as the project title in a .json file. Our HTML file automatically reads the json when it is loaded to produce a simplified HIT page shown in Figure 3. This simplified page is shown to the workers in the HIT, and a link to the original Kickstarter project page is provided in case the worker wants to get more information on the project.

Figure 3 Simplified Kickstarter project page in the HIT

4.2 Feedback Visualization

The system offers an interface that reports the feedback from the crowd to the requester. Instead of showing tables and numbers, we chose to use several interactive visualizations to help the requester explore responses. We present the feedback saturation pattern for a single question in one page. Users can easily choose one question from a list of questions to visualize and can also switch between questions to compare them.

There are four major goals of the visualization feedback:

1. Display the distribution of feedback clusters
2. Present the transition of saturation level
3. Demonstrate the evolvement of major feedback clusters.
4. Replay intermediate statuses of the entire feedback process

In the following paragraphs, four major functional modules are described along with their contributions to accomplish the goals above.

4.2.1 Clustering Algorithm

Our clustering algorithm is based on the k-means clustering algorithm. The input of the clustering algorithm is the pairwise similarity score of each pair of answers within an answer pool. We put the first answer in the first cluster as a starting point. Then for each new answer we receive, we calculate the average of the similarity scores between the new answer and all the answers in the first cluster to find the similarity score for that cluster. After finding the cluster average similarity scores for all the clusters, we place the new answer in the cluster with the highest similarity score if the similarity score is greater than the threshold score of 50%. If the highest similarity score is lower than the threshold, we create a new cluster and place the new answer in that cluster. This algorithm worked fairly well for finding the appropriate cluster when there were more than three answers in the cluster. However, we wanted to ensure that clusters with two answers had higher similarity scores since the clustering was based on only one similarity score. So we added an additional step of dividing all the clusters with only two answers and finding a new cluster for those answers with a higher threshold of 80%.

4.2.2 Cluster Distribution Graph

The cluster distribution graph is located at the bottom of the interface. Each circle in the view represents an answer from the crowd for the question. All the circles are grouped into clusters, corresponding to the results of clustering algorithm. Clusters are evenly spaced and circles within a cluster are organized with a pack layout. We chose to use spatial mapping to convey the concept of response clusters.

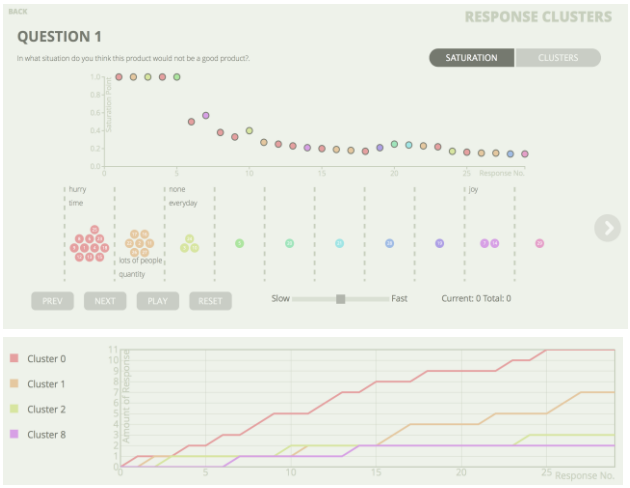

Figure 4 Feedback visualization interface

As described above, we also extract the keywords for each response cluster. Keywords for one cluster are listed within its area. The keywords are alternately placed at the top or bottom of the cluster as shown in Figure 4 to avoid the overlapping of keywords of adjacent clusters. If the answer is gathered as free form text, the full content of the answer is displayed when the cursor hovers over its circle. This gives the user the access to the raw data from MTurk.

4.2.3 Saturation Graph

The saturation graph aims at presenting the transition of saturation level over time. Each circle in the graph represents one answer, and its height shows the overall saturation level by the time that answer came in. The x axis is the number of responses, starting from the first to the last. Since the responses are sorted by time, this axis is also the time axis. The typical time axis spaced by a unit of time such as seconds, minutes, or hours is not suitable here because the actual time between the answers is irregular and the graph may be too sparse. The y axis is the saturation level, with a maximum value of 1.0 as defined. This is a typical dot graph layout. Since there is a new saturation level as each response comes in, circles are used to maintain consistency.

A shortcoming of typical dot graphs is the difficulty to read values accurately because the circles are far away from the axes. One solution is to increase the granularity of ticks on the axes. However, the numbers on the axes will be less readable due to the limited space. To balance this trade-off, a horizontal line and a vertical line will appear when the requester hovers over a circle, in order to pinpoint the circle's value on both axes. The exact values will also be displayed.

Different from the MTurk dashboard, this visualization feedback enables the requester to access all intermediate states of the feedback process. When the requester hovers over a circle, all the answers that were received after the selected answer will fade out on both Saturation and Distribution graphs. This technique provides a quick manual access to an intermediate state. We chose the hover event as a trigger to make this operation as quick and simple as possible so that different states can be compared with the least effort.

All the circles are filled with the same color as in the Cluster Distribution Graph. It not only enhances the consistency between graphs but also provides more information to help requester identify the trend of clusters.

4.2.4 Cluster Evolvement Graph

The Cluster Evolvement Graph depicts the growth of response clusters over time. The user can toggle between the Saturation Graph and this graph. The x axis marks the number of answers like the x axis of the Saturation graph. The y axis is the amount of responses in one cluster. Different from Saturation Graph, which visualizes the response set as a whole, this graph is for comparing different clusters. Therefore, each cluster is represented by one line in this graph, which matches the cluster color. Line was selected to represent a cluster to convey the concept of trend and evolvement.

Line graph also has the same accuracy problem as the dot graph. However, it is harder for the user to pick the point they want to explore on a line with a mouse. Therefore, instead of actively display a value for a point, we chose to display the axis grid to passively provide a measure to the user.

Another problem is the overlapping of the lines. The first cause of the overlap is the large number of clusters. As a solution, we reduced the number of lines by eliminating all the clusters that end up with only one answer, which actually eliminated a majority of the lusters. All the remaining clusters had more than one answer in them and were considered as major clusters. The second cause over the overlaps lies in the nature of line graph with multiple lines. Some lines will inevitably overlap since they have the same values. Therefore, a legend was added to help identify the lines as shown in Figure 4. More importantly, the boxes in the legend act as toggle buttons for the requester to hide or show certain clusters. The fill of a legend box indicates the visibility of the corresponding line.

4.2.5 Process Replay Animation

All three graphs are not limited to displaying the final results as they can all be animated to replay the entire feedback process from the first answer till the last. The user can also adjust the replay speed and skip through the responses manually. All the graphs will be updated as the animation is played. These functions give the requester a full access to all the intermediate states of the process and decide when to stop gathering responses.

All four of the modules above are designed to accomplish the four goals proposed at the beginning of this section.

The overall design is aimed at clarity and simplicity. A mono color scheme is used for all the background components such as axes and buttons. In contrast, more diverse colors are used on data points and lines to make them to stand out from the background. All the interactions are designed to be intuitive and understandable for everyone.

5. RESULT

5.1 Number of HITs estimated by the workers

The workers provided a wide range of estimated number of HITs and various reasons for deciding the number of HITs to post on MTurk. Although we provided three different questions for workers, the distribution of the number of HITs were similar for all three questions regardless of the question contents. It indicates that workers do not take the size of possible answers into account but just make a general guess about the number when they decide to receive subjective feedback. Due to the similar distribution on estimated number of Hits for all three questions, we combine all the results regardless of different questions. We removed irrelevant

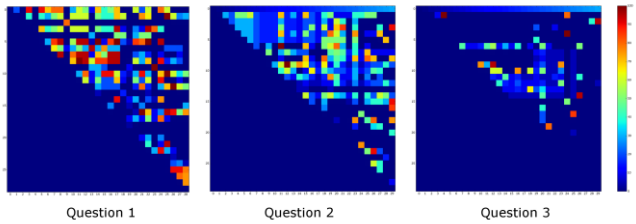

Figure 5 Heatmaps of pairwise similarity scores for three questions

answers before analysis. Four workers had misunderstood the question 'how many HITs would you post to MTurk' as 'how many answers would you ask from each of the workers per HIT' and answered in the range of one to five with the reason that it will tire out the workers if more answers were required. We discarded these answers since they did not answer the question that was actually asked. The remaining answers were categorized into three categories: 10-50 HITs, 100-500 HITs, and 1000+ HITs.

Table 1 Number of HITs estimated by workers

Est. Range of HITs	#Workers
<10	10
10-50	15
50-100	0
100-500	31
500-1000	0
1000	8
10000	2

Reasons for choosing the number of HITs

The first group, who answered that they would post ten to fifty HITs, was concerned about the tradeoff between the quantity of the answers and the effort that is required to go through each of the answers. They were worried that an important idea might get overlooked if there were too many answers. One worker said, "*I think 50 is a good sample size--not so large that the important points get lost, but big enough for many points to be brought up*."

The second group was more concerned about the tradeoff between the amount of information they could gather and the expense they have to pay to get the claimed number of answers. They avoid the expense becoming a "*sinkhole of money*," but still wanted to have a "*good sample group to see what the general consensus was from users*." The third group, which had less concern about the money, wanted to ensure a good sample of all the ideas even if it meant investing a little more money for it. A worker who estimated a thousand HITs reported, "*Such a simple task on MTurk wouldn't cost much, so I feel I could easily afford 1000 answers. In addition, I feel that 1000 is a good set to get some quality thoughts, at least initially*." Assuming that each HIT costs 10 cents, a thousand HITs would cost a hundred dollars. It is reasonable to invest a hundred dollars if it could help fundraise successfully since many projects have a goal amount that exceeds ten thousand dollars.

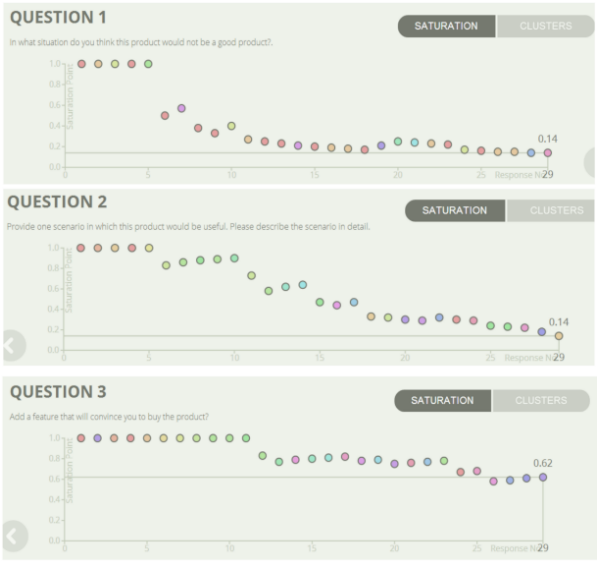

Figure 7 The saturation level graph for each of the three questions

Each group had legitimate reasons for choosing the number of HITs. However, most of them did not have a concrete idea of why they chose the number of HITs they did. Most workers used the words 'I think' or 'I feel' to support their decisions rather than backing up the number of HITs with a more objective theory or data. This shows that the number of HITs on MTurk can be arbitrary, and the requester might not know the appropriate number of HITs at the time of posting the request.

5.2 Simplified Kickstarter Project Page

In response to the question '*What was the most informative part of the Kickstarter project*,' 31% of the workers answered '*Video*.' They also vote for the *Image*/*Animation* feature provided in the description page. We notice that nearly all projects under *Design* category provide one promotional videos and various number of images. Some projects even provide more than ten images in their pages. To make our generated simplified project page more consistent in structure, we choose promotional video for the campaign, project title along with its essential information such as short description, goal amount, and creator's name as the elements of simplified Kickstarter page.

5.3 Feedback on Kickstarter Project

5.3.1 Pairwise Answer Similarity

We collected 30 answers for each questions listed in Section 3.1; the answers are used to answer RQ2 by checking for the existence of the saturation point and the general pattern of the answers. To acquire the saturation pattern, we calculate the saturation score at each data point. The saturation score come from two-stage process. First, find the similarity score for each pair of answers within the target answer pool. In this work, the similarity scores were manually coded by our group using the following rubrics.

0%: The answers have no common idea and describe completely different concepts

25%: The two answers present different concepts about a similar idea.

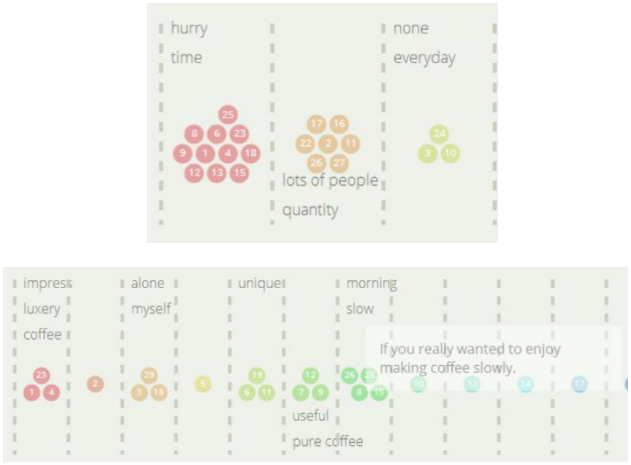

Figure 6 The main idea clusters generated from crowd workers for Question 1 (top) and Question 2 (bottom)

50%: The two answers share the same idea, but one answer provides additional information that does not relate to the common idea.

75%: The two answers share the same idea, but one answer provides additional information that relates to the common idea.

100%: The two answers have, and only have, the exact same idea.

A heat map of the pairwise similarity scores for each question is presented in Figure 5. Red blocks indicate answer pairs are coded 100% identical to each other in terms of given concept and blue blocks indicate answer pairs that are 0% similar to each other.

Figure 5 shows that the answers for Question 1 were mostly similar to each other while answers for Question 3 were most distinctive. This is not surprising since Question 1 asked about a scenario that the product would not be useful in, and the scenarios are mostly based on the shortcoming of the product. The product had distinctive shortcomings, such as the small quantity of coffee that could be made at one time and the time consuming process of hand dripping coffee. Thus the majority of the scenarios for Question 1 covered those issues.

On the other hand, Question 3 asked for an additional feature of the coffee maker that would convince the worker to buy the product. A wide range of possible features was available and the additional feature was related to worker's individual preferences rather than the characteristics of the coffee maker. Thus answers to Question 3 were rarely similar to each other.

5.3.2 Saturation Level

The saturation level marks how close the answer set is to being completely saturated. In theory, a completely saturated answer set already contains every possible idea and the incoming answers would not contain any new information. Therefore, 1.0 means that the answer set is not saturate at all, and 0.0 indicates complete saturation of the answer set.

The saturation level was derived by calculating the ratio of the new answers to the total number of answers in the set. An answer was considered as new when the average of similarity score of its top three similar answers was less than 50%. It indicates that we cannot find at least three answers provide the same ideas as the new data. So if we kept on getting new answers, saturation level would rise till it reaches 1.0 as the answer set moved away from complete saturation. Conversely, the saturation level would get lower if we

got more answers that repeated the ideas from previous answers as the answer set moved towards complete saturation.

So based on the similarity scores shown in Figure 6, saturation level for the final answer set for Question 1 was expected to be lower than the saturation level for the final answer set for Question 3.

This resulting saturation pattern for each question was presented in Figure 6. The graphs in Figure 6 mark the saturation level as each answer comes in. The x-axis indicates the answer received at that time, and the y-axis indicates the saturation level after that answer is added to the answer set. All the graphs start at the saturation level of 1.0 as each answer is new at the beginning of the data collection. The saturation level of both Question 1 and 2 drop after the fifth answer is added to the answer set while the saturation level for Question 3 remains at 1.0 till the 11th answer.

The saturation level of Question 1 drops dramatically till answer 11, which can be seen by the steep negative slope in the top graph in Figure 6. After receiving answer 11, the saturation level does not change much and only fluctuates a little bit till the end of the collection. This figure answers RQ2 and RQ3 well since it clearly shows that the answer set has reached a saturation point before the end of the collection, and it notifies the requester that it would be safe to stop the batch around answer 14. Contrastingly, the saturation graph for Question 3 shows a slow decline, and the saturation level of the entire answer set is 0.62, which is very high compared to Questions 1 and 2's ending saturation level of 0.14. Question 2 is the middle ground where the saturation level declines steadily till the end of the collection. Whether the batch should be stopped before the last answer or not in this example depends largely on the requester's preference.

Overall, dynamic notification would be useful for requests such as Question 1 and Question 2 where the answer set eventually reaches a saturation point and the system can notify the requester to stop the batch since the likelihood of the incoming answers carrying sufficient new information is small. On the other hand, Question 3 is a type of open question that might never reach a saturation point due to the wide range of possible answers. In this case, the requester might want to define the number of idea clusters that he wants to obtain from the answers instead of defining a saturation level to end upon.

5.3.3 Idea Clusters

After manually assigning the similarity scores for each of the answer pairs, we clustered the answers using the clustering algorithm mentioned previously in the Implementation section. Then we manually extracted the keywords for all the clusters that contained more than one answer.

Figure 7 shows the three main idea clusters for Question 1 that asked the workers to find a situation in which the coffee maker would not be good product. The largest cluster was about situations when people were in a hurry and do not have sufficient time to manually make coffee. This addresses one of time-consuming nature of the coffee maker. Based on this feedback, the requester might consider adding an alternative way to making coffee for people in a rush to make the product more suitable for all situations.

The second cluster addresses the small amount of coffee that the coffee maker can make at one time. This is especially problematic when combined with the slowness of the coffee maker and will make the product almost useless in settings such as meetings or conferences. The third group simply claimed that the product would be useful in all situations. The answers in this cluster show the difficulty of extracting keywords automatically as the answers seem to be saying the exact opposite things on the surface level: one answer said "no such situation" and another said "it is suitable in all situations." Ironically, 'no situation' and 'all situations' are referring to the same idea since the question asked for a situation in which the product would *not* be good. So the first answer is actually saying that there is no situation for which the product is *not* good.

The main clusters for Question 2 indicated that the product useful when one want to be unique, is making coffee for oneself, and want to enjoy making coffee slowly on a day off or Sunday morning. This presents an interesting point as it shows that the drawback of a product in some situations can also become merit of the product in other situations. Question 3 had a total of 21 clusters showing a wide variety of the answers. Some workers answered that there is no feature they would like to add. Others answered easier and faster preparation time, more space, availability in different colors, and warranty in case the glass breaks.

Looking at the keywords and the answers in each cluster, we concluded that the main clusters were good indicators of the product's strength and weaknesses. The keywords also provided a good summary general impression of the product without having to read through the individual answers. This would especially be useful for the requesters who want to have sufficient data, but are concerned about the time and effort that would be needed to sort out all the answers. The clusters also allow the requester to stop the batch after receiving a certain number of ideas, rather than waiting for the answer set to reach a saturation point.

5.4 Comparison of the Workers' Estimate and Saturation Point

In the first experiment, we looked at what the MTurk workers thought was an appropriate number of HITs for feedback on a Kickstarter project. The result showed that they did not consider the range of possible answers that differed based on the question, and they gave similar number of HITs for each of the question. However, experiment 2 showed that the appropriate number of HITs differs greatly based on the questions asked. Also, the median for the estimated number of HITs was in the hundreds, while the result based on the saturation point showed that Question 1 only required around 15 answers to gather the main ideas. Thus the workers' estimate number of HITs were almost tenfold the actual number of HITs that were needed. The dynamic notification of saturation level would be extremely useful in this case as it would save time and money for the requester. The actual number of HITs that was needed for Question 3 is vague as it does not reach a saturation point. However, the visualization of the clusters would be a good indicator of the range of ideas that are mentioned in the answer set and might help the requester decided when to stop the batch.

6. CONCLUTION

Crowdsourcing technology offers a promising approach to receive diverse feedback within a reasonable short time. More and more systems are designed to mediate the need between feedback providers and consumers. However, guidelines for determining the sample sizes are virtually nonexistent. When it comes to paid workers, the trade-off between the cost and the saturation of information is important. In this study, we propose a system to check data saturation pattern during the data collection process. We found that early saturation point exist in some feedback pool. Therefore, it is valuable to automatically notify the requester to stop collecting new data based on the feedback. Moreover, we provide the requester with multiple views of the idea distribution among

collected feedback. As a result, the requester can view the saturation level along with the summary of the collected feedback.

7. FUTURE WORK

We see at least three points that can be refined through future work. First, we should examine the variance of the estimated sample size among real visual designers or project creators. Their estimations on the number of HITs will reflect the real problem. If crowdsourcing becomes a promising approach to receive critiques for real designers, sample size will be an inevitable problem. Second, the pairwise similarity scores of the answers are calculated manually in our present work. We should incorporate natural NLP Toolkit to get real-time similarity score so that we could provide real time notification of data saturation. Finally, the usefulness of our system has not been evaluated from real users. We should do a usability test on visual designers or project creators in our next stage.

8. REFERENCES

[1] Feedback Army http://www.feedbackarmy.com/

[2] Usability Hub https://usabilityhub.com/

[3] Amazon Mechanical Turk https://www.mturk.com/mturk/welcome

[4] David R. Karger, Sewoong Oh, and Devavrat Shah. Iterative Learning for Reliable Crowdsourcing Systems. NIPS, 2011.

[5] J. Whitehill, P. Ruvolo, T. Wu, J. Bergsma, and J. Movellan. Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. NIPS, 2009.

[6] Guest, G., Bunce, A., & Johnson, L. (2006). How many interviews are enough? An experiment with data saturation and variability. Field Methods, 18, 59-82.

[7] Michael D Greenberg, Bryan Pardo, Karthic Hariharan, and Elizabeth Gerber. Crowdfunding support tools: predicting success & failure. In CHI'13 Extended Abstracts on Human Factors in Computing Systems, pages 1815–1820. ACM, 2013.

[8] Tanushree Mitra and Eric Gilbert. The Language that Gets People to Give: Phrases that Predict Success on Kickstarter. In CSCW'14. ACM

[9] Vincent Etter, Matthias Grossglauser and Patrick Thiran. Launch Hard or Go Home. In COSN'13. ACM

[10] V. S. Sheng, F. Provost, and P. G. Ipeirotis. Get another label? improving data quality and data mining using multiple, noisy labelers. KDD, 2008.

[11] David R. Karger, Sewoong Oh, and Devavrat Shah. Iterative Learning for Reliable Crowdsourcing Systems. NIPS, 2011.

[12] W. Willett, S. Ginosar, A. Steinitz, B. Hartmann, and M. Agrawala. Identifying Redundancy and Exposing Provenance in Crowdsourced Data Analysis. IEEE Transactions on Visualization and Computer Graphics, 2013.

[13] A. Strehl, J. Ghosh, and R. Mooney. Impact of Similarity Measures on Web-page Clustering. AAAI, 2000

[14] H. Wang, W. Wang, J. Yang, P. S. Yu. Clustering by Pattern Similarity in Large Data Sets. SIGMOD, 2002.

[15] M. Gordon. User-Based Document Clustering by Redescribing Subject Descriptions with a Genetic Algorithm. Journal of the American Society for Information Science, 1991.

[16] T. Mitra, E. Gilbert. The Language that Gets People to Give: Phrases that Predict Success on Kickstarter. CSCW, 2014.

[17] M. Greenberg, B. Pardo, K. Hariharan, E. Gerber. Crowdfunding Support Tools: Predicting Success & Failure. CHI, 2013.

[18] V. Etter, M. Grossglauser, P. Thiran. Launch Hard or Go Home! COSN, 2013