Crowdsourcing Accurate and Creative Word Problems and Hints

Yvonne Chen[1], Travis Mandel[1], Yun-En Liu[2], and Zoran Popović[1,2]
[1]Center for Game Science, Computer Science & Engineering, University of Washington, Seattle, WA
[2]Enlearn™, Seattle, WA
{evechen, tmandel, zoran}@cs.washington.edu, yunliu@enlearn.org

Abstract

The current state-of-the-art method for generating educational content, such as math word problems and hints, is manual authoring by domain experts. Unfortunately, this is costly, time consuming, and produces content that lacks diversity. Attempts to automatically address the time and diversity issues through natural language generation still do not produce content that is sufficiently creative and varied. Crowdsourcing is a viable alternative - there has been a great deal of research on leveraging human creativity to solve complex problems, such as user interface design. However, these systems typically decompose complex tasks into subtasks. Writing a single word problem or hint is a small enough problem that it is unclear how to further break it down, but also far more complex than typical microtasks like image labeling. Therefore, it is not obvious how to apply these worker improvement methods or which ones are most effective (if at all). We build upon successful task design factors in prior work and run a series of iterative studies, incrementally adding different worker-support elements. Our results show that successive task designs improved accuracy and creativity.

Introduction

For K-12 students, math is an important introduction to logical problem solving, helping them to develop skills that transfer to other fields of learning. Math word problems help students form connections between abstract math concepts and concrete situations. However, the existing pool of educational math content is relatively small and limited in diversity, which can have negative implications on student learning. For example, biases in word choice can lead to word problems that vary in difficulty level for students from different backgrounds (Freedle 2003). Furthermore, without a wide range of diverse examples, students learn to "solve" problems by memorizing patterns rather than truly internalizing concepts (Charles and Silver 1987; Reusser 1988). This can also lead to cheating and gaming tests that reuse the same problem structures, such as the phenomenon where international students with weak English skills can still score high marks on the language portions of standardized tests (Golden 2011). Therefore, if we wish to improve student learning, it is key that we generate many more diverse problems.

However, simply generating more problems is insufficient, as we also need to generate hints to help students when they are stuck. An inexpensive approach is to simply have generic hints that do not refer to elements of the problem, but researchers have shown that contextualized hints referring directly to the text of the word problem are more effective at pointing students in the right direction (Reusser 1988). Hence, we need a way to produce contextualized hints at scale.

Although we need more content, the current state-of-the-art method for generating math content cannot address the problems of scale and diversity. Math word problems and their corresponding hints are manually authored by domain experts, an expensive and time consuming process (Rumble 2001) that produces only a small (and likely biased) corpus of material.

An alternative approach to generating word problems and hints is natural language generation. Prior work explores methods to speed up and automate the production of word problems (Williams, Sandra 2011; Nandhini and Balasundaram 2011; Polozov et al. 2015), but still requires considerable human effort to create ontologies, and a limited ontology vastly limits the kind of problems that can be generated. The system most similar to ours, created by Polozov et al. (2015), generated their problems using a template-based approach that limited creativity and diversity. They found their word problems to be significantly worse than textbook problems across a variety of metrics designed to measure comprehensibility, coherency, and readability. We believe that a cost effective, diverse and scalable approach for creating educational content comes in the form of crowdsourcing platforms, particularly Amazon's Mechanical Turk (MTurk).

Tasks on MTurk typically fall into one of two categories: simple microtasks, like transcribing words from a photo or completing surveys, or complex tasks that can be broken down (Lasecki et al. 2015; Kittur et al. 2011; Little et al. 2010; Kulkarni, Can, and Hartmann 2012; Chilton et al. 2013). However, it is difficult to fit our domain into either of these categories. It is unclear how one would break down the task of writing a single hint or word problem into a collection of microtasks, nor are these tasks large enough to warrant more than one worker writing a hint or word problem together. Furthermore, although these tasks are small, they are complex, with many constraints compris-

ing creativity and mathematical accuracy, and it is unclear what features a task design should have to successfully help a worker write a word problem or hint.

In this work, we carefully adapt techniques from prior research on MTurk task design, such as including questions with verifiable answers (Heer and Bostock 2010; Kittur, Chi, and Suh 2008; Franklin et al. 2011), forcing workers to read instructions (Kittur, Chi, and Suh 2008) and enforced self-review (Di Stefano et al. 2014). Our results show that our successive task designs improved accuracy and creativity, both without incurring higher monetary costs. Hundreds of word problems and hints created in our studies have already been included in a math game played by thousands of students.

Background

Riddlebooks To make the general problem of crowdsourcing educational content more concrete, we focus on crowdsourcing new word problems and corresponding hints for the educational math game Riddlebooks, developed by the Center for Game Science. Riddlebooks is designed to cover many types of word problems, from simple addition and subtraction all the way through to algebra. Its educational approach derives from the highly acclaimed Singapore Math Model system (Kho, Yeo, and Lim 2009). This system teaches math skills by first emphasizing the conceptual relationships between entities in a problem. To solve a problem using the math modeling approach, a student first represents the elements of the word problem in a visual model, turns the model into an equation, then uses the equation to compute the numerical answer. Riddlebooks covers just the first two steps of this process, as its educational focus is on teaching mathematical relationships through the model method. The Singapore math modeling system classifies non-algebraic word problems into one of 8 *model types* and corresponding subtypes. In Riddlebooks, players solve these problems by dragging on-screen blocks and braces to model the relationship between variables (see Figure 1).

Objectives As the math model system targets a wide range of skill levels, we want Riddlebooks to be similarly broad. To achieve this breadth, we must generate an equal sampling of problems for all model types. If we asked workers to write an arbitrary math word problem without specifying further constraints, it is easy to imagine that we would collect a large pool of simple addition problems. Therefore, we need the ability to guide worker behavior towards *mathematical accuracy*, defined as writing problems that fit model types of our choosing. Additionally, we want the produced content to be interesting and engaging for students, and be varied enough to overcome the detrimental learning effects that arise from a lack of diversity, so we want to maximize the *creativity* of worker submissions.

Hinting Framework The Riddlebooks development team created a framework of generic hints for each math model type, based on key points where a student might need help during the process of solving a problem. Whenever

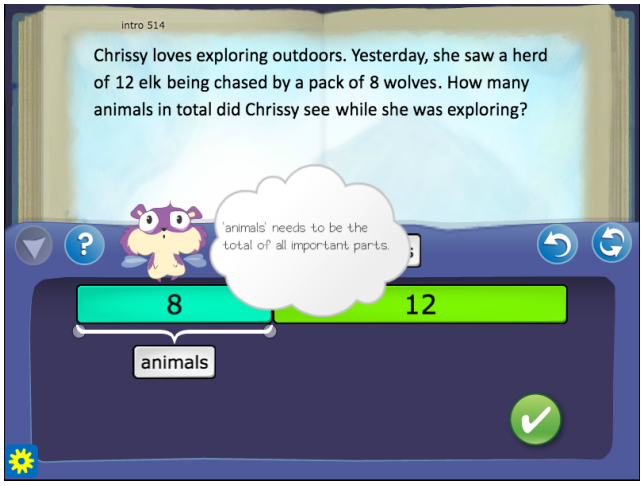


Figure 1: Riddlebooks gameplay screen showing an 'a + b = c' problem and a generic hint. In the correct model, the brace representing c (the sum) should span both bars instead of just the '8' bar.

the student clicks the hint button, the game determines their position in the framework based on the actions they have taken so far, and displays a corresponding hint. Each generic hint leads students towards the next step of the framework. For example, Figure 1 shows an "a + b = c" type addition problem. The player has identified the two elements to be added, but the brace used to represent the summation of the two quantities only spans one of them. Thus, the hint prompts, "'animals' needs to be the total of all important parts." However, as mentioned previously, contextual hints are the most useful to helping students who are stuck. Although the framework sometimes extracts key words from the text to contextualize hints, the result is often stilted. Due to the hundreds of word problems in the game, it is prohibitively expensive for a single person to write a specific, contextual hint for each word problem, let alone one for every single step of the hinting framework. The framework for the previous example of adding two numbers already consists of 10 cases where hints are needed.

Related Work

Automatic Word Problem Generation

Researchers in the field of machine learning and natural language processing have explored various ways to automatically create textual content.

In the domain of math word problems, Nandhini and Balasundaram (2011) explored automatic math problem generation for students with learning difficulties . They implemented two approaches, then evaluated their effect on student performance. A template-based method was simpler to implement, but produced questions with grammatical errors and very similar sentence structures which caused students to memorize solution patterns. A grammar-based method produced problems that better supported conceptual learning, but the approach was complex to implement, requiring

manual authoring of many grammar production rules, which is expensive and can only be done by those with a high level of domain-specific expertise. Furthermore, this method still produced problems of limited diversity and quality, with grammar errors and awkward phrasing present even in their hand-picked example problems.

More closely related to our domain, Polozov et al. (2015) proposed a technique for automatically generating personalized math word problems that fit the Singapore Math model system. After using answer-set programming to generate a logical tree from a set of input predicates, their system fills in primitive sentence templates to produce text for a word problem. Because this text is repetitive, the text is passed through a series of simple post processing steps like replacing nouns with pronouns. However, when compared to textbook math problems that were hand written by experts, these generated problems still rated significantly lower in metrics scoring readability, coherence and logical structuring - this seems to be an inherent flaw with any template based solution. We apply a similar a template approach to our crowdsourced setting as a baseline highly templated, fill-in-the-blanks style design, which was shown to produce the least creative word problems out of our three designs.

Turk Task Design

To improve accuracy, one approach is to ask another set of workers to verify the work of the first set of workers produced and correct minor errors (Bernstein et al. 2015; Ambati, Vogel, and Carbonell 2012). We believe this is undoubtedly an important step before deploying hints in real-world systems. However, in this paper we focus on getting creativity and accuracy as high as possible in the base tasks, so as to minimize the costs associated with future verification steps.

Prior work suggests tasks with verifiable answers can encourage accurate results (Heer and Bostock 2010; Kittur, Chi, and Suh 2008; Franklin et al. 2011). Since our task is open ended and there are many possible responses that would still be correct, we cannot construct a task that is completely verifiable. Instead, we show guidelines against which workers can compare their work.

MTurk's work paradigm does not impose implicit restrictions of expertise on workers. Sometimes, workers will simply lack the requisite skills to complete a task successfully. Posing preliminary tasks, known as qualification tasks, can filter out low quality workers, only retaining those who successfully complete the qualification task as candidates for the actual tasks (Heer and Bostock 2010). However, we are more interested in examining how changes in our task design affect worker submission quality and would like to avoid unnecessarily limiting the set of potential workers. We believe the complex nature of our task already deters dishonest workers and low-quality workers.

Decomposing Complex Tasks

Intuitively, simpler task designs lead to better worker performance (Finnerty et al. 2013). However, some tasks are too complex to be displayed in a simple format. One approach involves the task designer decomposing the complex tasks into discrete units that can be completed by different workers, then combined into a single result (Kittur et al. 2011; Little et al. 2010; Kulkarni, Can, and Hartmann 2012; Chilton et al. 2013). Another involves workers collaboratively deciding how to split up the labor of a complex task (Lasecki et al. 2015). Although our tasks of writing creative word problems and hints according to a set of mathematical constraints are complex, it is not clear how one would decompose such tasks into discrete units, let alone for workers themselves to decide how to structure such a breakdown. Thus, we instead explore how careful task design (Kittur 2010) can improve worker performance.

Study: Writing Word Problems

In our first study, we showed how one can loosen task constraints and add self-checking to achieve more creative and accurate submissions. We deployed three versions of a math word problem writing task, then asked a separate group of workers to rate the creativity of the word problems.

Some inspiration for our task designs come from a MTurk experiment by Di Stefano et al. (2014) which showed that workers who self-reflected on puzzle solving strategies performed better on further puzzles. Similar to self-reflection, we implement a self-checking step so that workers can compare their work against our guidelines. Before submitting their word problem, we prompt workers to check that it is creative and makes sense. We perform this prompting step in each of our designs.

Study Designs

Design 1: Mathlibs Similar to previous natural language generation systems like Polozov et al. (2015), our first design aims to maximize the accuracy of responses, though possibly at the expense of creativity. The most straightforward approach to maximizing accuracy is by requiring workers to follow a template. Our main inspiration comes from the popular game MadLibs. In MadLibs, one first thinks of words that correspond to some part of speech, such as nouns, verbs or adjectives. Then, these words are inserted into fixed blanks in a paragraph to produce an amusing story. Likewise, our first "mathlibs" design is similar in that we restrict workers to filling in blanks in a template.

We created a representative template for each subtype of the first 7 Singapore Math Model types, resulting in a total of 41 templates. Templates consisted of ordered textboxes that correspond to some part of speech. Filling in the boxes produces a word problem guaranteed to be mathematically accurate, exactly matching one of the Singapore Math Model types (but not necessarily grammatically accurate or coherent) (Figure 2). To enforce plot consistency, certain text boxes are grouped. If a worker fills in one member of the group, the contents of the others are automatically populated with the worker's entry. To introduce the setting and characters of their word problem, workers were allowed to add a short, freeform introduction before the text of the word problem.

Design 2: Freeform Word problems created using the first system seem likely to be formulaic, so we explored methods

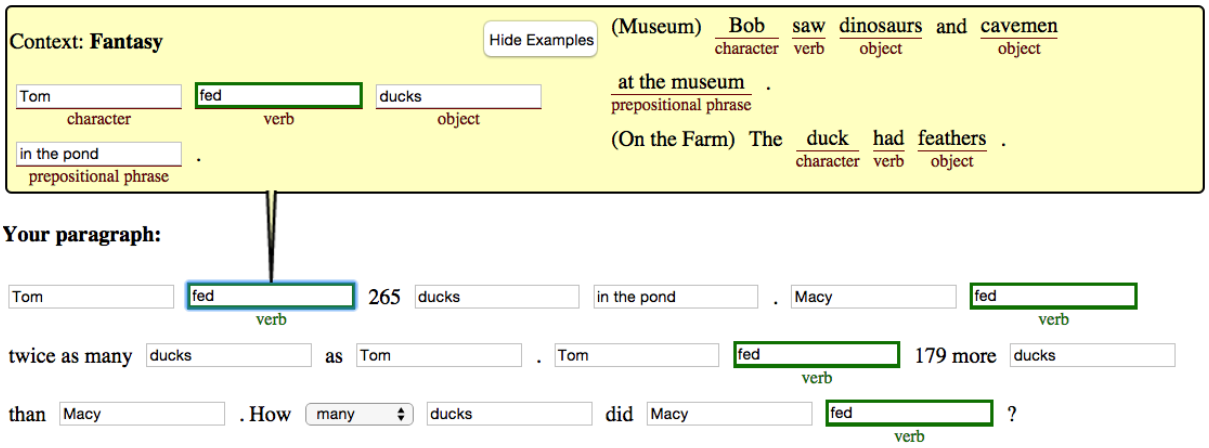


Figure 2: The first design, "mathlibs" of a system for MTurk workers to write word problems. The structure is highly constrained to fit a particular math model, which limits creativity.

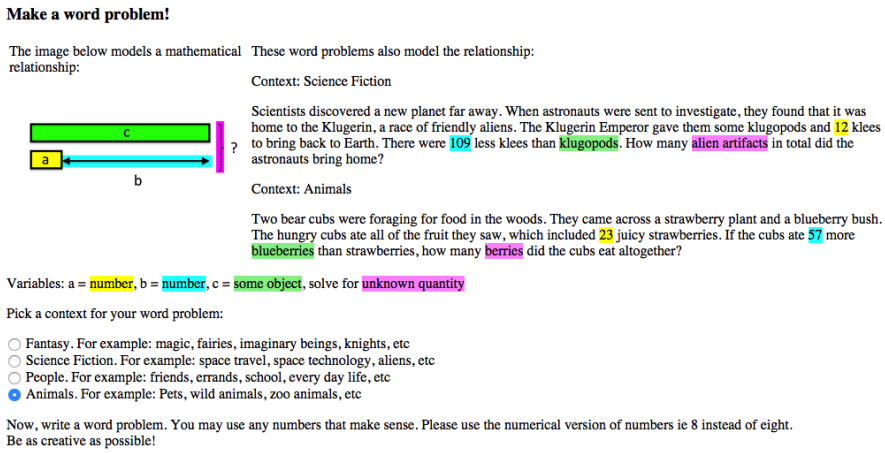


Figure 3: The second system design, "freeform". Workers can enter anything into a blank textbox for submissions, which is intended to increase creativity compared to mathlibs problems. Structural constraints are replaced with a visual diagram.

to improve the creativity of the produced problems without compromising accuracy. While mathlibs enforce constraints through forced adherence to a template, here we suggest the constraints through a visual representation of the underlying math model that should structure a worker's word problem. Our hope was that giving workers the freedom to write arbitrary full sentences would increase the possibility of more interesting problems, in contrast to the highly constrained space of possible mathlibs submissions. We call this system "freeform".

Figure 3 shows the MTurk UI for this design. Workers are given a single blank text box in which to write their word problem. We also display two sample word problems written to fit the displayed model, providing a starting point and plot inspiration. The model and sample problems are color coded to emphasize the connection between model entities and plot entities. Workers can choose from 4 contexts for the plot of their word problem: Fantasy, Science Fiction, Animals, and People. These contexts tie into the existing art and UI elements of Riddlebooks. At the end of the task, workers are shown a popup with their final submission. In addition to questions about grammar and creativity, they are also asked to verify that their problem fits the given math model.

Design 3: Tagged A major concern is that the second design might now be too unconstrained, allowing workers to submit word problems that did not match the displayed math model. Our third approach explores a two-step process that combines the freeform hint writing step with a tagging step to reinforce the math model constraints and impose another layer of self-checking (Figure 4). After writing the word problem, workers now also have to tag parts of their word problem to show that it fits the displayed math model. We hypothesize that this step either encourages workers to be more careful from the outset because they know about the second tagging step, or that while tagging their problems, they notice and correct errors. We anticipate minimal loss of creativity because workers can still write full sentences. We

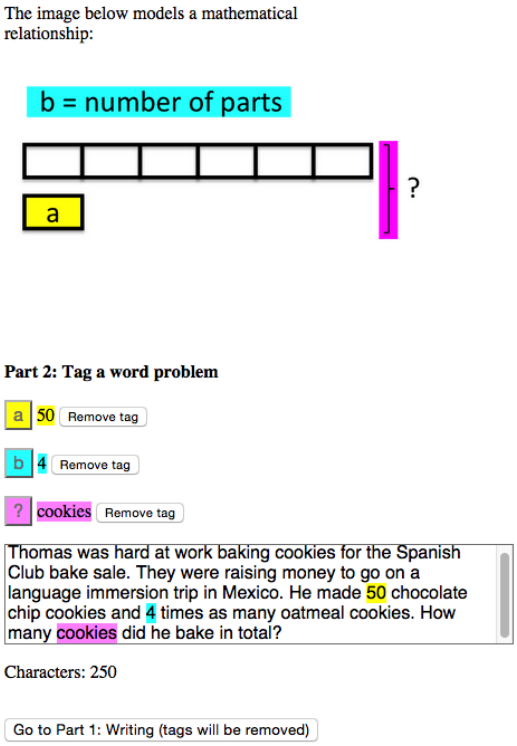


Figure 4: Our third system design, "tagged," introduces an additional tagging step which asks workers to verify the accuracy of their submission against the visual math model. We expect self-checking to increase accuracy.

will refer to this system as "tagged".

In this second tagging step, workers see a series of buttons color-coded to match parts of the model diagram and example word problems. They are asked to tag sections of their word problem with the color that corresponds to the appropriate part of the model diagram. Workers can switch freely between the problem writing and tagging steps to make changes. At the end of the task, workers were also shown a popup and asked to verify the accuracy of their work.

Methodology

We did not restrict workers by any demographic information. The only requirement for completing our task was a task approval rate of 98% or higher across at least 1000 completed tasks, for at least some filtering. We made this choice because we wished to study effects across a wider space of workers.

The last column of Table 2 lists payment per word problem in the three task versions. Payment increased between the first and second versions to speed up responses, and decreased between the second and third versions to see if task completion speed would remain at a reasonable level despite lower payment. However, the effects of payment on task completion time are not a focus of this paper. Furthermore, we do not anticipate these payment differences to have a significant effect on work quality, as suggested by some prior work (Marge, Banerjee, and Rudnicky 2010; Mason and Watts 2010; Mao et al. 2013).

Analysis

We first compare the accuracy rate of word problems that were created using the freeform and tagged systems. The first author manually coded each problem into two categories, based on whether or not its underlying mathematical structure fit the displayed math model. This rating process was not blind. We did not perform ratings on mathlibs problems, as they are all 100% accurate due to the underlying template structure. We performed a chi-square test of independence to examine the relationship between problem accuracy in the freeform and tagged systems. As expected, tagged problems were found to be significantly more accurate at 52.95% than freeform problems at 44.28% ($\chi^2(1, N = 1250) = 8.96, p = 0.0028$) (Figure 2). This suggests tagging caused workers to engage more deeply in our task and pay more attention to our requirements, leading to higher performance.

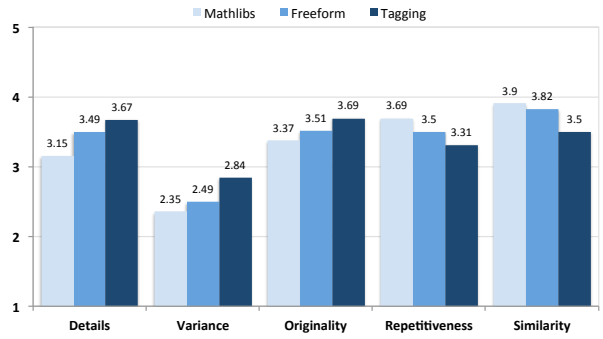


Figure 5: Mean creativity ratings for each of the three systems. Ratings were scored on a scale of 1 to 5. Higher values of repetitiveness and similarity are considered worse based on our questions (see Table 3).

Upon further analysis, we realized several prolific workers wrote many problems in the freeform and tagged versions of the experiment, meaning their personal accuracy ratings could possibly mask any changes in overall accuracy due to interface improvements. We define prolific workers as those who wrote 10% or more of the total word problems in a version. As shown in Figure 6, this cutoff only excludes outliers: two in the tagged system, and just one in the freeform system. We performed another analysis excluding these workers, and our results still show a significant improvement from 31.24% for freeform problems to 60% for tagged problems ($\chi^2(1, N = 950) = 77.65, p < 0.0001$). Notice that the tagged problems' accuracy increased after excluding prolific workers, while freeform problems' decreased; we suspect this is due to some prolific workers being more accurate and others inaccurate, as opposed to differences in the task structure. Our accuracy results are summarized in Table 2.

We also conducted an experiment to analyze the rela-

Details	Z = -2.36, 0.0483
Variance	Z = -1.39, ns
Originality	Z = -1.49, n.s
Repetitiveness	Z = 1.88, n.s
Similarity	Z = 0.78, n.s
N = 233	

(a) Mathlibs vs. freeform

Details	Z = 1.51, n.s
Variance	Z = 1.99, ns
Originality	Z = 0.96, n.s
Repetitiveness	Z = -1.34, n.s
Similarity	Z = -2.13, n.s
N = 209	

(b) Freeform vs. tagged

Details	Z = 3.63, p <0.0008
Variance	Z = 3.08, p <0.0059
Originality	Z = 2.75, p <0.0165
Repetitiveness	Z = -3.08, p <0.0059
Similarity	Z = -2.88, p <0.0110
N = 238	

(c) Tagged vs mathlibs

Table 1: Results from analyzing creativity ratings on word problems created using the three systems. Pairwise analyses were conducted with the Steel-Dwass All-Pairs method.

tive creativity of word problems generated using each system. MTurk workers, different from the set who wrote word problems, first read five word problems chosen from one of the three systems, and which were written for one of three randomly selected math model types. They then answered five 5-point scale questions about the creativity of that group. Since the structure of the mathlibs system included hard-coded numeric values, which could artificially limit perceived creativity, we replaced those with randomly generated numbers. Because our data was non-parametric, we performed a Kruskall-Wallis test for statistical significance for all conditions, then used the Steel-Dwass All Pairs method to examine pairwise differences between systems. Results are shown in tables 1a, 1b, and 1c. As expected, word problems created using the tagged system were significantly more creative compared to those written using the mathlibs system. Although there was only one significant comparison between the mathlibs and freeform systems (1a), Figure 5 shows that the mean creativity ratings for the freeform system are all better than those of mathlibs.

We expected the mathlibs problems to be significantly less creative than problems in the other two systems, but the absolute differences seem small. We were particularly surprised that the creativity metrics for the mathlibs system are already quite high (Details and Originality were above 3.00 on average) despite the use of a single restrictive template. We suspect that the freeform introductions before main mathlib text encouraged higher detail and originality ratings. If this is indeed the case, then mathlib creativity scores are somewhat misleading, as we are primarily concerned with improving creativity in the completely templated main body of the word problem. It is unclear how to separate these two notions of creativity without complicating the rating task. However, we still think the statistically significant improvement across all five metrics shows the more freeform systems are able to produce more creative output.

It is possible that the example word problems we showed caused some workers to use them as the basis for their own problems, resulting in a less diverse result set and thus diluting the impact of tagging on worker performance. We also hypothesize that more data would increase the statistical significance of this data. Although we did not emulate every aspect of Polozov et al.'s system (2015), results suggest that our tagged system may produce more creative problems than templated approaches.

Version	**# Tasks**	**Overall**	**Without Prolific**	Price per Task
Mathlibs	392	100%	100%	$0.50
Freeform	531	44.28%	31.24%	$0.75 - $1.00
Tagged	489	52.95%	60.0%	$0.50 - $0.15

Table 2: Tagged problems were more accurate than freeform problems, even when excluding work from prolific workers. The last column lists prices paid to workers per task.

Question	**Scale**
On average, how detailed are these word problems?	Simple - Detailed
Comparing the word problems to each other, how varied are they?	Similar - Varied
On average, how original are these word problems?	Unoriginal - Original
Reading these word problems one after another, how repetitive do they seem?	Distinct - Repetitive
Comparing the word problems to each other, how similar are they?	Different - Similar

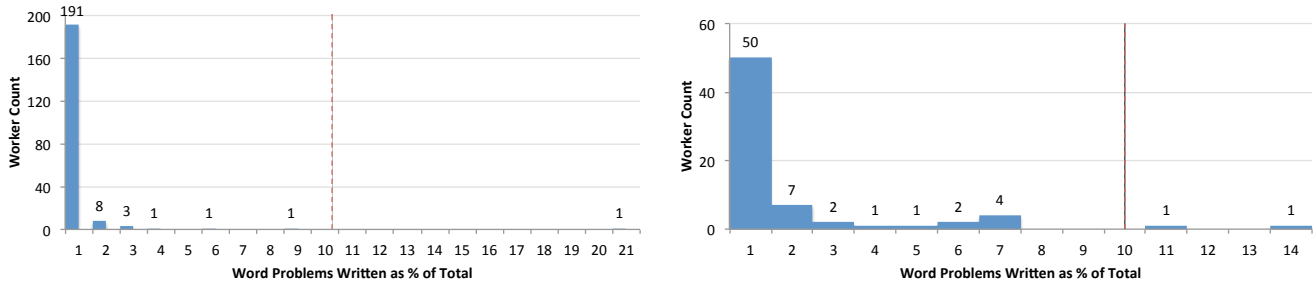
Table 3: Questions asked about the creativity of word problems.

Note that although our results indicate the tagged system generates the most creative output, the accuracy is considerably worse than the perfect accuracy attained by the mathlibs system. However, as we feel that more creative word problems are key to improving student learning, we believe this is a good tradeoff in our setting. Additionally, it is much easier to boost accuracy after the fact (e.g. with verification tasks) than it is to boost creativity. Thus, we use the highly templated mathlibs system primarily as a baseline.

Overall, our analysis suggests that adding the additional self-check step in the tagged system helped improve accuracy over the freeform system and creativity over the mathlibs system, while trending in creativity improvements over the freeform system. While Stefano et al. (2014) showed the effectiveness of self-reflection on problem solving strategies, here we show its effectiveness on improving output in constrained problem writing.

Study: Writing Hints for Word Problems

After developing systems for generating word problems, we needed a process to fill out contextualized hints that replace the generic hints in the hinting framework. Since the framework is seeded with an initial set of generic hints, this set of



(a) Freeform

(b) Tagged

Figure 6: Histograms for word problems written per worker, expressed as a percentage of total word problems written.

hint writing tasks can be seen as rewriting a generic hint in a more contextualized manner. For example, a generic hint of "This problem has things being added together. What are they?" could have the contextualized hint of "Which two groups of students are you adding together?". To analyze the effect of various design constraints on worker accuracy, we deployed two versions of a math hint writing task on MTurk. Each task involved workers writing a hint for one of the math word problems collected in the previous study.

Study Designs

Design 1: Baseline Some inspiration for our first hinting task design comes from prior work. Mao et al. showed that one of the major concerns of MTurk workers is that their work does not meet requester standards. Low quality work can lead to rejected submissions, which reduces a worker's approval rating and restricts the pool of tasks they can complete. To alleviate this concern, a set of guidelines is displayed against which workers can compare their work. Additionally, before completing our task for the first time, workers are required to go through a tutorial. Prior work by Kittur et al. (2008) suggests that forcing workers to process content can increase their work quality.

Likewise, our baseline design (Figure 7) shows a checklist of our guidelines as the worker writes their hint. At the top of the page, workers see a math word problem. Then, they view two diagrams that visually represent the mathematical relationships between elements of the word problem. One diagram represents a student's incorrect diagram, and the other represents the next step in the hinting framework. Below this, workers see a generic hint corresponding to a step in the Riddlebooks hinting framework. Finally, workers write a custom hint that contains the same level of detail as the generic hint, but refers to story elements of the word problem. Workers can hover their mouse over any task element to display a tool tip explaining its purpose.

Design 2: Faded Tutorial Compared to writing word problems, the process of creating a hint here requires more steps, so it is crucial that the worker understands every part of the process. Thus, the tutorial is of utmost importance. Prior work suggests that fading in answer choices on survey questions reduces satisficing (Kapelner and Chandler 2010), so we think requiring workers to step through the tutorial might prompt them into reading instructions more carefully and thus further enforce our guidelines. Our second design Figure 7 modifies the first page of the tutorial so that task elements fade in one after the other. At first, only the first task element, the word problem, is shown. After the worker hovers over it to read the explanation tooltip, the second task element, the incorrect diagram, appears, and so on. We also modified the tutorial example hints to fit closer to the tone of the generic hint. Since our criteria for rating accuracy was the same for both versions, we do not think this change had a significant effect on worker performance.

Methodology

As before, we did not restrict MTurk workers by any demographic information. The only requirement for completing our task was a task approval rate of 95% or higher across at least 1000 completed tasks, which was more permissive than our previous choice of 98%. We made this change to explore effects across a larger worker space. Workers were paid $0.05 per submission.

Analysis

Version	Tasks	Overall	w/o Prolific	Dates Run (2016)
Baseline	2138	50.0%	42%	Feb 12 - Feb 25
Faded Tutorial	684	84.0%	58%	Apr 5 - Apr 12

Table 4: Accuracy results for each version of the hint writing task, overall and with excluding prolific workers.

Accuracy was determined this time by whether the hint was appropriate for the generic hint's position in the framework (see Background section) and whether it was contextualized (i.e. referred to story elements of the word problem).

We randomly sampled 100 hints from each design. The first author manually rated them as accurate or inaccurate according to a set of guidelines: the hints had to apply correctly to the error situation, not give away the answer, and refer to story elements in the word problem. We did not consider hints to be inaccurate based on grammatical or spelling errors. The rating was blind between versions. Examples of each error type, drawn directly from worker submissions, are

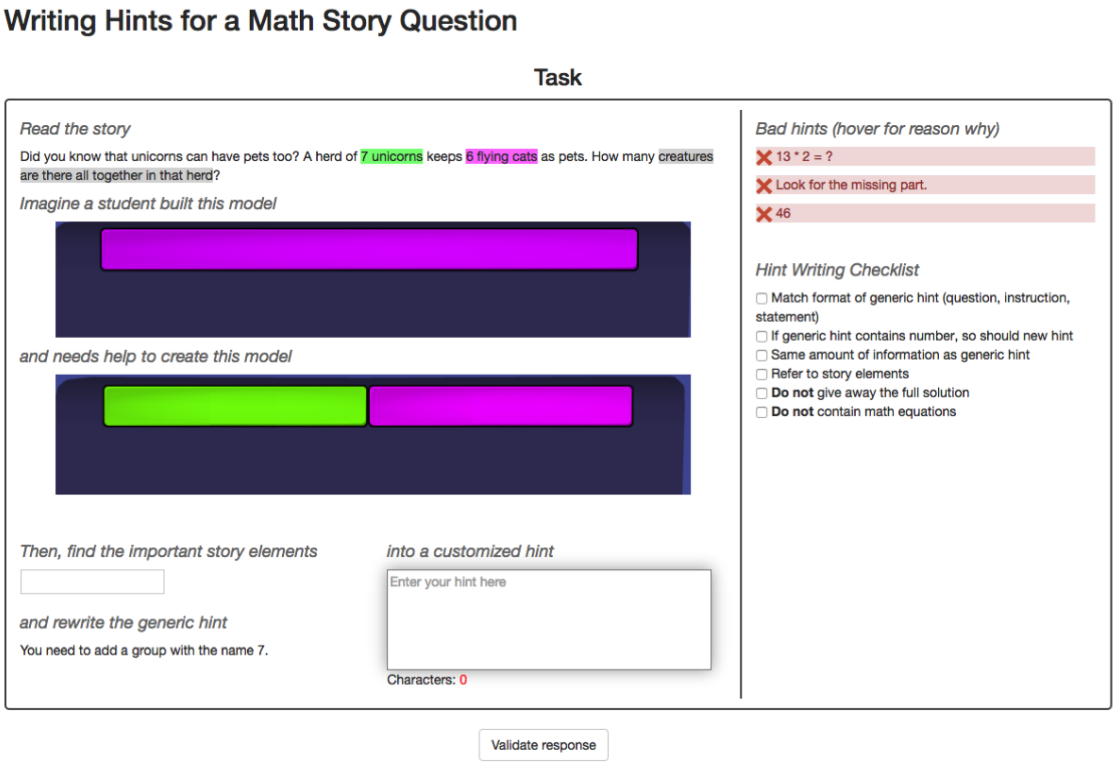


Figure 7: The layout of our baseline hinting task.

shown in Table 5. Results of a chi-square test for independence show that the accuracy improvements of hints written in each successive version (Table 4) are statistically significant $\chi^2(1, N = 2822) = 9.30, p = 0.00229$ from 50.0% in the baseline version to 84% in the faded tutorial version.

Although this difference is very large, the result does not take prolific workers into account. One major concern is that the difference in accuracy between versions could be simply due to getting lucky (or unlucky) from a few very good (or very bad) prolific workers. In this case, we did observe a single worker who completed almost 80% of the second version's hints. Therefore, we rated accuracy again after excluding workers who had produced at least 10% of the hints in a particular task version. This is a reasonable cutoff as it only excludes outlier workers (Figure 8), three in the first version and one in the second. The resulting accuracy rates are 42% in the baseline version versus 58% in the faded tutorial version ($\chi^2(1, N = 2822) = 53.04, p < 0.0001$). Although a smaller change compared to the initial analysis, this is still a relatively sizeable boost in accuracy, which suggests that the faded tutorial does in fact improve accuracy.

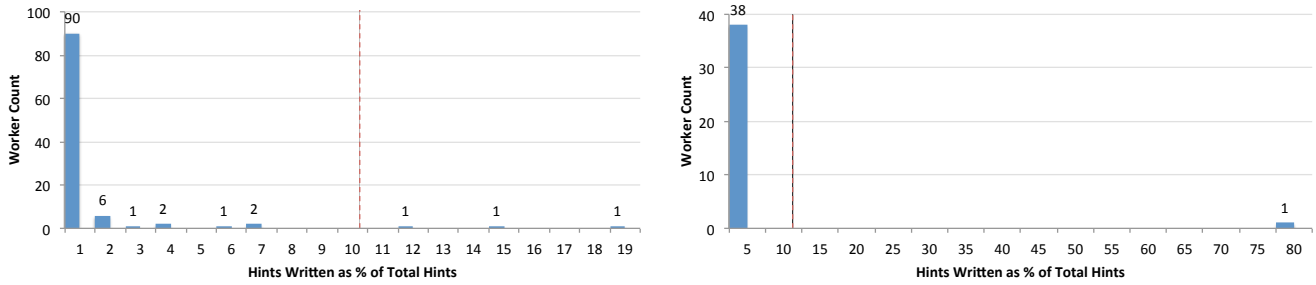
Our results show that a faded, step by step tutorial can improve work quality, likely by encouraging workers to pay more attention to instructions. Kapelner and Chandler (2010) showed previously that an approach of fading in content can cause survey respondents to pay closer attention to answer choices, and our results show that this technique can induce better task performance by making workers pay more attention to the instructions. This faded tutorial approach could also easily generalize to many other creative, constrained tasks.

Limitations and Future Work

In our word problem writing task, all tasks for a given model type had the same example problems, shown prominently to the workers as they completed the task. Some workers could have based their work off the examples, resulting in lower overall creativity in the submissions. Future work could explore approaches that better encourage diversity, such as displaying randomly selected examples from a larger, varied set or displaying contrasting examples of accurate versus inaccurate word problems (Cheng and Bernstein 2015).

The text of word problems and hints should flow naturally before being shown to students. Although we considered creativity and mathematical accuracy in our rating metrics, we did not judge the grammatical accuracy and coherence of worker submissions. Improving these through crowdsourcing is an area of future work. If we limit our word problems to only those written by native English speaking workers, we hypothesize that they are already more comprehensible and grammatically correct than problems produced by natural language generation systems, but further study is necessary to be conclusive.

Our results also show that some workers will dominate the results space by completing high volumes of tasks. Some can be relied on to produce consistently good work, while others cannot. Although prior work has already investigated measures to prevent dishonest workers from ruining results (Kit-



(a) Baseline

(b) Faded tutorial

Figure 8: Histograms for hints written per worker, expressed as a percentage of total hints written. We define prolific workers as workers who wrote 10% or more of the total hints in a version.

Error	Incorrect Hint	Reason
Missed instructions	"Find the total of the 2 parts"	Doesn't refer to any parts of the word problem.
Doesn't match the generic hint	"89 HORSES THE KING"	Horses and a king were both mentioned in the word problem, but the hint doesn't convey the desired info.
Gives away the answer	12 elk + 8 wolves = 20 animals	Although contextual, this hint provides too much information compared to the generic hint.

Table 5: Examples of common errors noticed in the incorrect hints written by MTurk workers.

tur, Chi, and Suh 2008; Kulkarni, Can, and Hartmann 2012), there is little exploration on the effect of positive prolific workers, who are important for tasks where the primary objective is high volumes of accurate work, rather than a wide sampling of participants. To gain insight into the minds of these individuals, we reached out to the prolific worker who completed 80% of the second version's hints (with an accuracy rate of 92%) and asked what motivated them to work. They replied:

> *I like HIT's [human intelligence tasks] that involve writing. The price was right for the work... I like HIT's that are not brainless*

This anecdotally suggests that prolific high-quality workers can be motivated by factors outside of pure financial incentives, leading to interesting questions for settings where a minority of workers complete the majority of work. For example, if one designs a system that relies heavily on output from Turk, and a single worker is the driving force behind keeping the system going, what happens to the system if they decide to stop working?

Another future direction is automatic selection of the best hints for each student. In our studies, we have produced a large set of hints, but it is unclear which of these hints to actually show students. In this paper we manually evaluated hints based on mathematical accuracy, but we truly want to show each student the hint that will best maximize their long-term learning. One promising option is to treat hints as actions in a reinforcement learning framework, with reward derived from the measured learning and engagement of each student. This should allow us to automatically determine the best path of hints for a particular student through the vast space of hints generated through MTurk, thus resulting in more personalized and targeted educational experiences. We believe this is a rich area of future research.

Conclusion

Large quantities of diverse content are important to math education. We explore the application of crowdsourcing to the new setting of math word problems and hints, we examine various design factors to improve worker creativity and submission quality, showing how one can adapt techniques from prior work to improve worker accuracy and creativity. Hundreds of word problems and hints produced by workers through our systems have been augmented into Riddlebooks, which has been played by thousands of students in a recent algebra challenge. Additionally, results from both studies suggest further avenues of research into designing tasks that attract individual prolific workers, which can lead to higher volumes and success rates of task completion in crowdsourcing work from other domains.

Acknowledgments

This work was completed with support from the Center for Game Science, Office of Naval Research (ONR) grant N00014-12-C-0158, the Bill and Melinda Gates Foundation grant OPP1031488, the National Science Foundation (NSF) BIGDATA grant No. DGE-1546510, the Hewlett Foundation grant 2012-8161, Adobe, and Microsoft. This work was also supported by the NSF Graduate Research Fellowship grant No. DGE-1256082. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the NSF. We would also like to thank all the Mechanical Turk workers who completed our tasks.

References

Ambati, V.; Vogel, S.; and Carbonell, J. 2012. Collaborative Workflow for Crowdsourcing Translation. In *Proceedings of*

the ACM 2012 Conference on Computer Supported Cooperative Work, CSCW '12, 1191–1194. New York, NY, USA: ACM.

Bernstein, M. S.; Little, G.; Miller, R. C.; Hartmann, B.; Ackerman, M. S.; Karger, D. R.; Crowell, D.; and Panovich, K. 2015. Soylent: A Word Processor with a Crowd Inside. *Commun. ACM* 58(8):85–94.

Charles, R. I., and Silver, E. A. 1987. The Teaching and Assessing of Mathematical Problem Solving. *Research Agenda for Mathematics Education Series* 3.

Cheng, J., and Bernstein, M. S. 2015. Flock: Hybrid Crowd-Machine Learning Classifiers. In *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing*, CSCW '15, 600–611. New York, NY, USA: ACM.

Chilton, L. B.; Little, G.; Edge, D.; Weld, D. S.; and Landay, J. A. 2013. Cascade: Crowdsourcing Taxonomy Creation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13, 1999–2008. New York, NY, USA: ACM.

Di Stefano, G.; Gino, F.; Pisano, G.; and Staats, B. 2014. Learning by Thinking: Overcoming Bias for Action through Reflection. *Harvard Business School Working Paper Series* 58(14-093).

Finnerty, A.; Kucherbaev, P.; Tranquillini, S.; and Convertino, G. 2013. Keep It Simple: Reward and Task Design in Crowdsourcing. In *Proceedings of the Biannual Conference of the Italian Chapter of SIGCHI*, CHItaly '13, 14:1–14:4. New York, NY, USA: ACM.

Franklin, M. J.; Kossmann, D.; Kraska, T.; Ramesh, S.; and Xin, R. 2011. CrowdDB: Answering Queries with Crowdsourcing. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data*, SIGMOD '11, 61–72. New York, NY, USA: ACM.

Freedle, R. O. 2003. Correcting the SAT's Ethnic and Social-Class Bias: A Method for Reestimating SAT Scores. *Harvard Educational Review* 73(1):1–43.

Golden, D. 2011. China's Test Prep Juggernaut. [2016-04-22].

Heer, J., and Bostock, M. 2010. Crowdsourcing Graphical Perception: Using Mechanical Turk to Assess Visualization Design. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, 203–212. New York, NY, USA: ACM.

Kapelner, A., and Chandler, D. 2010. Preventing Satisficing in Online Surveys. *CrowdConf 2010*.

Kho, T. H.; Yeo, S. M.; and Lim, J. 2009. *The Singapore Model Method for Learning Mathematics*. Marshall Cavendish Int (S) Pte Ltd.

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

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

Kittur, A. 2010. Crowdsourcing, Collaboration and Creativity. *XRDS* 17(2):22–26.

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

Lasecki, W. S.; Kim, J.; Rafter, N.; Sen, O.; Bigham, J. P.; and Bernstein, M. S. 2015. Apparition: Crowdsourced User Interfaces That Come to Life As You Sketch Them. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, CHI '15, 1925–1934. New York, NY, USA: ACM.

Little, G.; Chilton, L. B.; Goldman, M.; and Miller, R. C. 2010. TurKit: Human Computation Algorithms on Mechanical Turk. In *Proceedings of the 23Nd Annual ACM Symposium on User Interface Software and Technology*, UIST '10, 57–66. New York, NY, USA: ACM.

Mao, A.; Kamar, E.; Chen, Y.; Horvitz, E.; Schwamb, M. E.; Lintott, C. J.; and Smith, A. M. 2013. Volunteering Versus Work for Pay: Incentives and Tradeoffs in Crowdsourcing. *First AAAI Conference on Human Computation and Crowdsourcing*.

Marge, M.; Banerjee, S.; and Rudnicky, A. I. 2010. Using the Amazon Mechanical Turk for transcription of spoken language. In *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*, 5270–5273.

Mason, W., and Watts, D. J. 2010. Financial Incentives and the Performance of Crowds. *SIGKDD Explor. Newsl.* 11(2):100–108.

Nandhini, K., and Balasundaram, S. R. 2011. Math Word Question Generation for Training the Students with Learning Difficulties. In *Proceedings of the International Conference & Workshop on Emerging Trends in Technology*, ICWET '11, 206–211. New York, NY, USA: ACM.

Polozov, O.; O'Rourke, E.; Smith, A. M.; Zettlemoyer, L.; Gulwani, S.; and Popović, Z. 2015. *Personalized Mathematical Word Problem Generation*, volume 2015-January. International Joint Conferences on Artificial Intelligence. 381–388.

Reusser, K. 1988. Problem solving beyond the logic of things: contextual effects on understanding and solving word problems. *Instructional Science* 17(4):309–338.

Rumble, G. 2001. The Costs and Costing of Networked Learning. *Journal of Asynchronous Learning Networks* 5(2).

Williams, Sandra. 2011. Generating Mathematical Word Problems. *Question Generation: Papers from the 2011 AAAI Fall Symposium*.