Scalable and Explainable Automated Scoring for Open-Ended Constructed Response Math Word Problems

Scott Hellman and Alejandro Andrade and Kyle Habermehl
Pearson
Boulder, CO
{scott.hellman, alejandro.andradelotero,
kyle.habermehl}@pearson.com

Abstract

Open-ended constructed response math word problems ("math plus text", or MPT) are a powerful tool in the assessment of students' abilities to engage in mathematical reasoning and creative thinking. Such problems ask the student to compute a value or construct an expression and then explain, potentially in prose, what steps they took and why they took them. MPT items can be scored against highly structured rubrics, and we develop a novel technique for the automated scoring of MPT items that leverages these rubrics to provide explainable scoring. We show that our approach can be trained automatically and performs well on a large dataset of 34,417 responses across 14 MPT items.

1 Introduction

Math word problems are a common question type in both formative and summative mathematics assessment. In a math word problem, the prompt describes a scenario and asks the student to calculate some value or construct some mathematical expression pertaining to that scenario. Such problems assess both the student's ability to carry out mathematical computation and reasoning as well as their ability to apply their knowledge in determining how to solve a mathematical problem.

Automated assessment of closed constructed-response (CR) math problems is straightforward, although complexities arise due to the variety of possible representations for a given mathematical expression. Examples of automated assessment systems for closed CR items include m-rater (Fife, 2017) and MathQuery (Streeter et al., 2011). In contrast, open-ended CR math problems are difficult to automatically score, since responses to open CR items combine mathematical expressions with prose explanations. And if a problem asks students to both compute a value and explain their computation, that introduces the complexity of partial credit; in the dataset we consider in this work, score ranges for items vary between 0–2 and 0–4. Even for humans, these sorts of items, with partial credit and open-ended responses, are time-consuming to score (Stankous, 2016).

Automated assessment of CR items outside of mathematics is now common, thanks to the achievements of researchers in the areas of Automated Essay Scoring (AES) and Automated Short Answer Scoring (SAS). The reliability of AES systems is often comparable to that of humans (Shermis and Burstein, 2003, 2013), and the same is true for SAS systems (Butcher and Jordan, 2010). Given that MPT items are themselves CR items, this suggests that such approaches could also be used for MPT; research in this area is promising, but sparse (Erickson et al., 2020; Cahill et al., 2020).

How mathematical expressions are encoded in response text is a key attribute of a given MPT dataset. In this work, we use data generated by a writing environment that allows students to enter mathematics using a math editor tool. Any math written in this tool is represented in the final response text as Content MathML (an XML-based specification for the representation of mathematics). As students can also write math outside of the math editor, the dataset that we consider in this work contains math represented both in MathML and in plain text, often within the same response.

Given this set of challenges, our interest is in creating an *explainable* predictive model for MPT. Such a model would be able to differentiate, for example, between a response that received a 1 out of 3 because it contained the correct final answer without showing work, and a response that received a 1 out of 3 because it contained correct reasoning but incorrect computations. A model that successfully achieved this would be useful both for students, as they would better understand why their responses received their assigned scores, as well as for test administrators, as the explanations would build trust in the validity of the model's scoring.

This paper is structured as follows. We begin with a discussion of related work and a detailed description of our task. We introduce a novel scoring model that uses the rubric's structure to provide explainable scoring for MPT, and show how our model can be automatically trained. We then present experimental results that show the effectiveness of our approach, and conclude with a discussion of the present and future work.

2 Related Work

There is a substantial literature around the automated scoring of non-mathematical CR items. Work on AES dates back to the 1960s (Page, 1966), and modern-day AES systems involve a wide variety of approaches, including linear regression (Larkey, 1998), random forests (Hellman et al., 2019), and neural networks (Taghipour and Ng, 2016; Dong et al., 2017; Riordan et al., 2017). Short answer scoring is also relevant, as our MPT responses tend to be only a few hundred characters long. For SAS, many systems involve paraphrase detection, or some similar notion of semantic similarity to reference answers (Leacock and Chodorow, 2003; Tandalla, 2012; Ramachandran et al., 2015; Kumar et al., 2017).

While much work has been done on AES and SAS, as well as around the automated *solving* of math word problems (e.g. Kushman et al. 2014; Huang et al. 2016; Wang et al. 2017; Xie and Sun 2019), work around the automated *scoring* of math word problems is more limited. Livne et al. demonstrate a system that successfully uses instructor-provided reference answers to automatically score responses to closed CR math word problems (Livne et al., 2007). Lan et al. present a system that predicts scores by embedding multi-step math responses using a bag-of-expressions model, a bag-of-words approach designed to capture mathematical features (Lan et al., 2015). Once embedded, they use a combination of clustering and limited human scoring to score all responses. However, while their items were open CR math word problems, any prose in student responses was ignored by the scoring system.

Some systems do attempt to grapple with the full complexity of open CR math word problems. Kadupitiya et al. present a system that can score CR math word problems for summative assessments whose responses contain both prose and math (Kadupitiya et al., 2017). Their system assumes that all math is encoded as MathML, and prose is handled by estimating the semantic similarity of response phrases to known reference phrases. Erickson et al. (Erickson et al., 2020) investigated the effectiveness of random forests, XGboost, and LSTMs for scoring formative open CR math problems with only plain text responses, and follow-up work has shown that transformer-based approaches can also perform well on this task (Baral et al., 2021; Shen et al., 2021).

As mentioned above, we expect that many real-world MPT datasets will include responses that contain math represented both as plain text and as MathML. To the best of our knowledge, Cahill et al. is the only published work that attempts to score these sorts of responses (Cahill et al., 2020). In their work, they extract plain text math from student responses using regular expressions, and then use the m-rater (Fife, 2017) math scoring system to evaluate the correctness of this extracted math. They then build a feature space that includes binary features indicating whether certain rubric elements were covered by the student response. By training machine learning models on this feature space, they create models with interpretable features. This process requires knowledge of the rubric during training. Our work differs from Cahill et al. in that the model that we introduce relies *only* on features that are aligned with the rubric, and produces scores that are inherently explainable. Furthermore, it requires no knowledge of the rubric during training. We also evaluate our approach across a wider variety of items with more responses per item.

3 Open Constructed Response Math Word Problems

The dataset we use in this work is proprietary, so we have adapted an item from the GSM8K dataset [1] (Cobbe et al., 2021) as an illustrative example, shown in Table 1. In this example, the prompt establishes a scenario and asks the student to compute a value related to that scenario. The rubric defines three binary components that a response can achieve, which defines the score range for this item to be from 0 to 3. Finally, the example response shows a typical mixing of MathML and prose.

[1]Dataset located at `https://github.com/openai/grade-school-math/tree/master/grade_school_math/data`.

Example Prompt	Albert is wondering how much pizza he can eat in one day. He buys 2 large pizzas and 2 small pizzas. A large pizza has 16 slices and a small pizza has 8 slices. If he eats it all, how many pieces does he eat that day?
Example Rubric	1 point for correct computation (48 pieces). 1 point for correct modeling of the number of slices for the large pizza $(2 * 16)$ and the small pizza $(2 * 8)$. 1 point for correct modeling of the total number of slices (32 + 16).
Example Response	He eats 32 from the largest pizzas because <math> <apply><eq/> <apply> <times/> <cn>2</cn> <cn>15</cn> </apply>32</apply> </math>. He eats 16 from the small pizza because <math> <apply> <eq/> <apply> <times/> <cn>2</cn><cn>8</cn> </apply>16</apply> </math>. He eats 48 pieces because <math><apply> <eq/> <apply> <plus/>< cn>32</cn> <cn>16</cn> </apply>48</apply> </math>.
Normalized Response	He eats 32 from the largest pizzas because 2*15=32. He eats 16 from the small pizza because 2*8=16. He eats 48 pieces because 32+16=48.

Table 1: Example item and response. Adapted from the GSM8K dataset (Cobbe et al., 2021)

We are focused on word problems that ask the student to construct some mathematical equation and/or compute some number, as well as to provide the work and reasoning that they used in coming to their answer. For some items, this explanation is required to be prose, while for others the chain of mathematical expressions that led to the answer can suffice.

Each item has a rubric composed of some number of *computation*, *modeling*, and *reasoning* components, each of which is worth one point. Computation components generally refer to the presence of a correct final answer, modeling components to showing the correct mathematical derivation of the final result, and reasoning components to an explanation of why those steps were taken. A given rubric may not include all three of these components, and may also define multiple components of a given kind. The final score of a response is the sum of these binary component scores. Note that even if a rubric does not require a prose explanation, the student may still include prose in their final response.

The characteristics of the dataset used in this work are shown in Table 2. A critical aspect of our dataset is that MPT problems are, in general, quite difficult for students to answer correctly. For some items, more than 70% of student responses received a score of 0. This is an expected feature of our dataset, as math word problems are known to be substantially harder for students to solve than conventional math problems (Cummins et al., 1988).

Student responses are written in an environment that supports the entry of both plain text in a conventional text field and of math via a math editor. Critically, arbitrary text input is allowed in the math editor, to support the presence of variables in the student answer. While the expectation is that students will use this math editor to write the relevant mathematical expressions, and write the rest of the response outside of the math editor, in practice students often write prose inside of the math editor and math expressions outside of the math editor. Thus, we cannot look only at the MathML in a response to identify the mathematical statements produced by the student, and we cannot look only at the plain text to identify their explanations and supporting arguments. Because of this, we believe the best way to score MPT responses is by converting them to a normalized form.

This normalization process consists of three steps: first, we convert mathematical terms in the response into their symbolic equivalents, e.g. "eight" to "8", or "plus" to "+". Next, we need to account for prose written in the math editor. We identify MathML containing chains of variables being multiplied together that appear to spell out English words. When such a chain is found, it is removed from the MathML and converted to plain text by preserving the order of the variables and removing the multiplication operators. This replaces the variables in the MathML by their corresponding plain text word. Finally, we transform all remaining MathML into plain text by taking the in-order traversal of the expression tree defined by the MathML.

item	grade	domain	response count	mathml %	char count	score range	sp 0 %
1	7	algebra	4095	18.1	167	0-2	37.2
2	high school	algebra	2634	26.4	166	0-2	77.4
3	high school	algebra	2472	47.3	109	0-2	92.6
4	high school	algebra	2362	34.4	97	0-2	81.8
5	high school	algebra	5701	28.3	202	0-4	70.0
6	4	arithmetic	1266	72.5	87	0-2	50.0
7	5	geometry	1596	62.7	125	0-3	32.7
8	7	algebra	1665	29.6	198	0-3	41.4
9	7	algebra	1085	32.7	95	0-2	39.9
10	7	algebra	838	30.8	91	0-3	70.5
11	high school	algebra	1581	19.7	294	0-2	63.5
12	high school	algebra	1495	22.2	294	0-4	72.5
13	6	algebra	6018	36.0	212	0-3	54.6
14	6	arithmetic	1609	29.5	259	0-4	40.1

Table 2: Dataset summary. Mathml % is the mean percentage of characters in a response occurring inside of MathML spans. Char count is the mean number of characters. Sp 0 % is the percentage of responses at scorepoint 0.

4 Explainable Scoring

As outlined in Section 3, the rubrics for our MPT items are highly structured. We leverage this structure to create a new approach to the automated scoring of MPT items by essentially codifying the rubric in a machine-understandable way. The close alignment of our model with the rubric produces predictions that are inherently explainable.

Rules form the core building block of our approach. Rules encode short mathematical expressions and the transformations required to convert them into other lexically distinct but semantically identical forms. For example, a rule encoding "2 + 3" could generate "3.0 + 2" as an alternative form. These alternate forms account for different mathematical properties, principally commutativity and conversion between floats and integers (for whole numbers). To account for variables, we also allow single letters to serve as operands in our expressions.

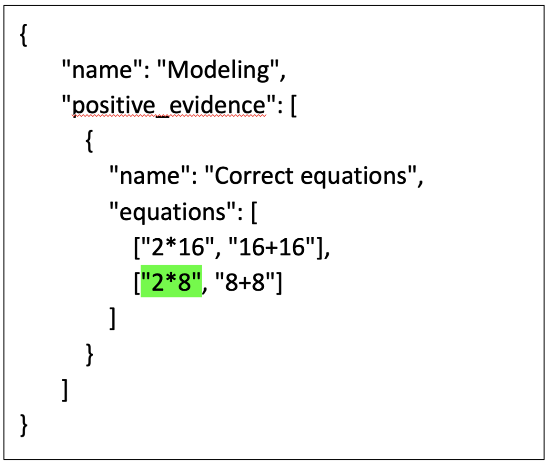
To determine if a rule is present in a student response, we first extract all mathematical text from the normalized text of the response. This is to prevent superfluous words from obscuring the underlying mathematics. See Figure 1b for an example. Then, if any of the forms of a rule are present as a substring of the extracted math, that rule is considered to be present in the response.

The amount of prose in a response is highly item-dependent. To account for items where prose is important, we also include the ability to write regular expressions as rules. Such a rule is found in a response if its constituent regular expression has at least one match in the response.

Assembling these rules into a form that can automatically score responses is done as follows. We define a *group* to be a list of rules, and we consider a group to be present in a response if any of its constituent rules are present. This allows us to capture mathematics that are equivalent under the rubric but not captured by the lexical transformations of our rules, for instance, "2 * 16" and "16 + 16" could be two valid ways of writing an expected expression.

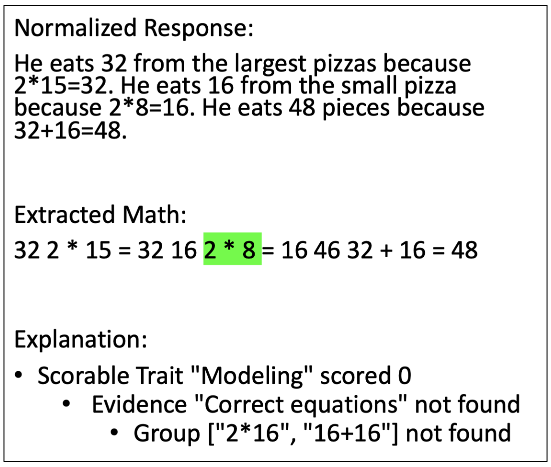
We then create *evidence* out of these groups. Evidence is a list of groups, and we consider evidence to be present in a response if *all* of its constituent groups are in the response. This allows us to capture rubric elements that require the student to cover multiple areas. For example, if a student needs to show two distinct values to achieve a Computation component, we can capture this notion by constructing evidence with two groups, one for each of those two distinct values.

Finally, to mirror the structure of the rubric components, we collect evidence into *scorable traits*. A scorable trait contains lists of positive and negative evidence. If any positive evidence and no negative evidence is present in a response, then the scorable trait scores a 1. Otherwise, it scores a 0. We include this concept of negative evidence to account for misconceptions and other incorrect mathemat-



```
{
    "name": "Modeling",
    "positive_evidence": [
      {
        "name": "Correct equations",
        "equations": [
          ["2*16", "16+16"],
          ["2*8", "8+8"]
        ]
      }
    ]
}
```

(a) Example Scorable Trait



Normalized Response:

He eats 32 from the largest pizzas because 2*15=32. He eats 16 from the small pizza because 2*8=16. He eats 48 pieces because 32+16=48.

Extracted Math:

32 2 * 15 = 32 16 2 * 8 = 16 46 32 + 16 = 48

Explanation:

- Scorable Trait "Modeling" scored 0
  - Evidence "Correct equations" not found
    - Group ["2*16", "16+16"] not found

(b) Example Response

Figure 1: An example scorable trait is shown in Figure 1a. This scorable trait captures a modeling component from the example shown in Table 1. This scorable trait is composed of one piece of positive evidence, which in turn consists of two groups. The first detects if the student found a correct equation for the number of slices for the large pizza. The second detects if the student found a correct equation for the number of slices for the small pizza. Figure 1b shows the normalized response from Table 1, alongside the math extracted from the response. The highlighted characters indicate where in the response the rules from the Scorable Trait were found. The automatically generated explanation of the score is also shown.

ics that can prevent a student from receiving full credit on a rubric component. For example, if an item asked the student to compute 4 divided by 2, the student could incidentally compute the correct value by subtracting 2 from 4.

We construct a number of scorable traits corresponding to the number of components in the rubric, and the final predicted score for a response is the sum of the individual binary trait scores. Because we know exactly which rules, groups, evidence, and scorable traits were found or not found when scoring, we can automatically construct an explanation of our predicted scores. See Figure 1 for an example of a scorable trait and the score and explanation it produces.

5 Automated Discovery of Rules

Given the hierarchy of rules, groups, evidence, and scorable traits described above, one approach to developing a scoring model would be to define all of these elements manually. While manually constructed models perform well (per our experiments below), requiring manual effort to construct a scoring model prevents the adoption of this approach at any scale larger than a small handful of items. Thus, we would like to automate this process. However, our model is not differentiable, so approaches such as stochastic gradient descent can not be used.

Simulated annealing is a highly flexible optimization technique that makes few assumptions about the objective function being optimized (Kirkpatrick et al., 1983). When applied to our modeling task, simulated annealing maximizes the performance of a model by iteratively adding or removing rules. If a change increases the model's training set performance, we keep it. Otherwise, the change is stochastically accepted with a probability based on a temperature variable and the difference in performance between the new and previous states. As the procedure continues over many iterations, the temperature is slowly reduced according to a cooling schedule. The result of this is a process that initially makes many random changes, but that tends towards only making changes that maximize the performance of the model as the temperature decreases.

In practice, we evaluate the performance of our models using both accuracy and the unweighted average recall (UAR), and so we optimize against both of these metrics during the annealing process. That is, our goal is to maximize the following function:

$$S(\theta) = \lambda * \mathrm{UAR}(\hat{y}_\theta) + (1 - \lambda) * \mathrm{Acc}(\hat{y}_\theta)$$

where θ corresponds to the model parameters, i.e., the rules, groups, and evidence of the model, $\hat{y}_\theta$ to the predictions of the current model on the training set, and λ is a hyperparameter that controls the relative importance of UAR versus accuracy.

To use simulated annealing, we must define the ways in which an existing model can be altered to generate a new model. We begin by building a set of candidate rules. Candidate math expressions are generated by identifying sequences of alternating operands and operators in the math extracted from a response. In this work, we consider sequences of up to six operands. Once these expressions have been identified, we rank them according to their information gain. We keep the top n expressions as our set of candidate rules for use in annealing.

When humans craft manual rules, they are able to write regular expressions. Automatically determining useful regular expressions in full generality is beyond the scope of this work, but providing our automated rules with some ability to reason about prose writing is important. For this reason, we consider all words in the responses, again rank by information gain, and then keep the top m as regular expression rules (that ultimately will match if the given word is present in the response).

When annealing our rules, we allow for four transformations:

1. Add a rule to a group.
2. Remove a rule from a group.
3. Replace a rule with a new rule.
4. Move a rule from one group to another group.

We initialize our model to have a number of scorable traits equal to the maximum score for the item, and create a user-defined number of empty evidences and groups for each trait. To improve final model performance, we use random restarts during training. That is, we perform k simulated annealing runs, and keep the model with the best training set performance as our final trained model.

To avoid overfitting to our training data, we also include two regularization terms in our objective function. The first term, $R(\theta)$, penalizes the model by the total number of operands used by all rules. The second term, $E(\theta)$, penalizes the model for the number of non-empty evidences used by the model. Our final objective function is

$$S'(\theta) = S(\theta) + \gamma * (\alpha * R(\theta) + \beta * E(\theta))$$

where α, β, and γ are hyperparameters that control the relative and overall regularization strength.

6 Experiments

To the best of our knowledge, there is no publicly available dataset that features open CR math word problems with a large number of student responses per item. For example, the GSM8k Dataset used in Table 1 has only one response per item. Therefore, we use our own proprietary dataset of MPT items for our experiments. This dataset consists of 14 items covering algebra, arithmetic, and geometry, targeting grade levels from fourth grade to high school. The scoring scales for these items range from 0–2 to 0–4. See Table 2 for detailed per-item information.

Our primary goal is to evaluate the performance of our rules-based model, both with manually crafted rules and automatically learned rules. The manual rules used in these experiments were crafted by human experts, who were allowed to view only a randomly sampled subset of the responses for each item. Responses used in this way during rule creation were also used for hyperparameter search for the simulated annealing approach, but were excluded from the dataset used in the final experiments. The response counts in Table 2 correspond to the counts used in our final experiments.

We perform a grid search for the cooling rate, number of iterations to run annealing for, and the overall regularization strength γ. Our pool of candidate rules consists of the top 500 expressions and top 50 words. We spend 1000 iterations at each temperature, create 3 positive evidences and 1 negative evidence for each trait, allow up to 10 groups per evidence, and set $\alpha = 0.0025$, and $\beta = 0.01$. We use a geometric cooling schedule, and perform 5 random restarts. These settings are based on values that were found to work well during initial development. We use 5 stratified and randomized train/test splits when performing this hyperparameter search, with 25% of the data in the test split.

Prior work has found that traditional AES approaches can work well for MPT, such as random forests (Erickson et al., 2020) and recurrent neural networks (Cahill et al., 2020). For this reason, we compare our rules-based scoring to three other conventional approaches: fine-tuned DistilBERT (Sanh et al., 2019), character n-gram random forests, and word n-gram random forests.

For both random forest models, we use regression random forests with 100 trees, and 33% of the features considered at each split. We keep all n-grams that occur in more than 5% of documents

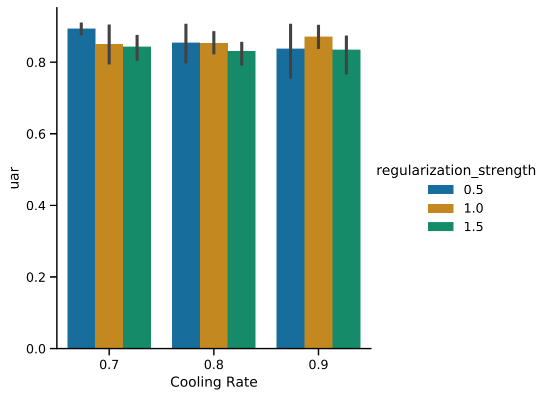
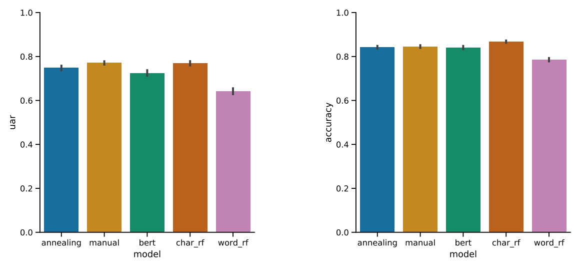


Figure 2: Mean UAR achieved by annealing on Item 6 at various regularization strengths and cooling rates at 25,000 annealing iterations. We also evaluated performance at 50,000 and 75,000 iterations, but performance across all three settings was similar, so we show only the results for 25,000 for clarity. Error bars are 95% bootstrap confidence intervals.



(a) Overall UAR (b) Overall Accuracy

Figure 3: Mean UAR and accuracy of each approach, averaging over all items and folds. Error bars are 95% bootstrap confidence intervals.

and in fewer than 95% of documents. For character n-grams, we consider n-grams ranging from 3 to 6 characters long. For word n-grams, we consider n-grams ranging from 1 to 4 words. We use scikit-learn's implementations of random forests and count vectorizers (Pedregosa et al., 2011).

For the DistilBERT model, we finetune all layers using the Adam optimizer. We use a learning rate of 2e-5, a weight decay of 0.01, and train for 4 epochs. The training data is further split into a final training set and an evaluation set; we evaluate model performance on the evaluation set after each epoch, and we evaluate our final test-set performance on the model that achieved the best evaluation set performance. DistilBERT uses wordpiece tokens (Wu et al., 2016) with a 512 token context window. All of our responses fit within this window; the longest response in our dataset is 501 tokens long. Our DistilBERT fine-tuning utilizes Hugging Face (Wolf et al., 2020).

For both random forests and DistilBERT finetuning, all hyperparameters not mentioned here were left at their default values.

For each item, we create 30 stratified and randomized train/test splits, with 25% of the data in the test split, and train and evaluate all models on these splits. We evaluate model performance using both accuracy and the unweighted average recall (UAR). In our operational scoring, poor performance at any scorepoint can rule out the use of a model, and UAR captures this by considering the impact of poor performance at rare and common scorepoints to be equivalent. For all regression models, we generate final score predictions by rounding the model output to the nearest whole number.

7 Results and Discussion

The results of our hyperparameter grid search for simulated annealing are shown in Figure 2. We see that performance is quite robust across all hyperparameter settings tested. Best performance is achieved by annealing for 25,000 iterations, with a cooling rate of 0.7 and a regularization strength of 0.5. These are the settings that we use for simulated annealing in the other experiments described in this section.

The mean UAR and accuracy of each model, averaging over all items and folds, is shown in Figure 3. Focusing on UAR, we see that the random forest using word n-grams performs noticeably worse than the other approaches. Character n-gram random forests and manually crafted rules perform well. Finally, we see that our annealing-based approach to automatically constructing rules performs slightly worse than the manually crafted rules, but slightly better than the DistilBERT model.

When we compare accuracy trends, we see that our rules-based approaches perform no better than DistilBERT. This is due to performance at the lowest scorepoints - these tend to be common (and thus prominent in the calculation of accuracy), but the rules-based approaches tend to have slightly lower recall at the lowest scorepoint. This is not seen in the UAR figures because the rules-based models tend to perform slightly better on the higher (and rarer) scorepoints.

In Figure 4, we show the mean UAR of each model for all items. Our discussion here will focus on items 2 and 10; these items were chosen as examples where the annealing approach performs

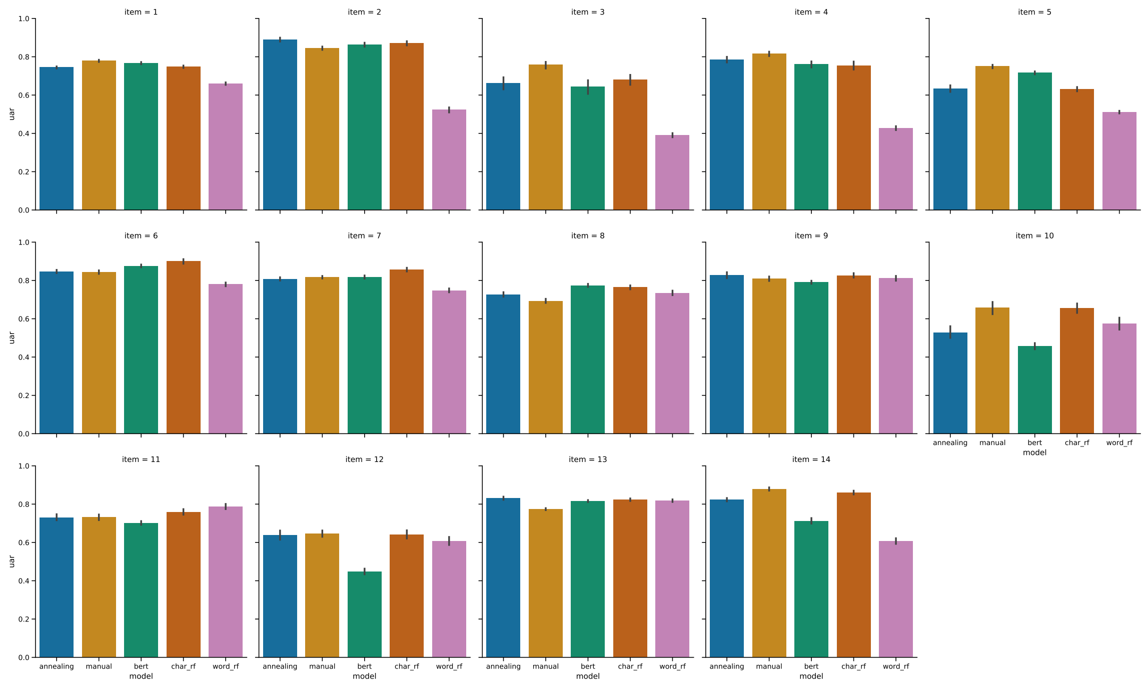


Figure 4: Mean UAR per item, averaging over all folds. Error bars are 95% bootstrap confidence intervals.

very well and very poorly, respectively.

For Item 2, our annealing approach performs the best out of all models. This item describes the improvement in average speed of two athletes over the course of a training regimen, and asks the students to calculate at what week of training their average speeds will be equal. The rubric contains a computation component, requiring the students to calculate the correct week, and a modeling component, requiring students to show their work in calculating their answer. The annealing process successfully constructs evidence both for identifying when the correct answer is present, and for identifying work that supports that correct answer.

In contrast, for Item 10, the annealing process performs quite badly. Item 10 asks students to calculate the speed of a real car based on the performance of a scale model of that car. The rubric contains one computation component, for the correct final speed, as well as two modeling components, one for proper unit conversion and one for correctly scaling the speed to the full-size car. The manually crafted rules perform comparably to the character n-gram random forest for Item 10, indicating that is possible for our rules-based approach to perform relatively well on this item. However, our manual rules for this item make extensive use of regular expressions, both to capture information about units and to capture notions such as the student stating in prose that they multiplied by the scaling factor. These sorts of sophisticated regular expressions are not captured by our current candidate rule generation process.

The relatively lackluster performance of the DistilBERT model is surprising, given the dominance of transformer-based approaches in many areas of NLP. However, there is a substantial literature detailing how both recurrent and transformer-based neural models can struggle with mathematics (Huang et al., 2018; Cobbe et al., 2021; Hendrycks et al., 2021). This literature, in combination with our results here, suggests that fine-tuning off-the-shelf neural models is not a particularly powerful approach for MPT scoring.

In light of these results, we conclude that our rules-based approach enables explainable automated scoring of MPT items without sacrificing performance, at the cost of requiring manual effort in designing the rules. However, we also have found that a simulated annealing-based approach to automatic rule creation can produce explainable models that are almost as effective as manually crafted rules, allowing for scalable and explainable MPT scoring.

8 Conclusion and Future Work

We have presented a novel, explainable approach to scoring MPT items via handcrafted rules that performs well, and have shown that such rules can be automatically discovered through simulated annealing.

While our model is able to provide explanations of its scores, generating explanations is only the first step in the full explainability process. Explanations are of limited utility without the ability to convey model explanations to stakeholders such as test takers or test administrators. Determining how best to use the explanations produced by our models is an important area of future work.

Our approach is heavily reliant on the assumption that the final score of a response is the sum of multiple binary components. For MPT items that are not structured in this way, it is unlikely that our approach would work well on its own, although it could possibly be combined with other approaches. We are actively investigating how best to extend our approach to more rubric types.

The success of our annealing process ultimately relies on our ability to generate useful candidate rules. While our current process works well, we have seen that for some items, we need to be able to construct more sophisticated rules. Determining how to improve the generation of our candidate pool is another promising area for future work.

The dataset we used in this work is mainly composed of algebra problems. While we do have some geometry and arithmetic items, how well our approach can generalize to other MPT item types is an area of future work. In particular, our items do not cover calculus, trigonometry, or other areas that require students to extensively reason about functions.

Acknowledgements

We would like to thank Alicia Bouy for her assistance in constructing the manually-crafted rules, and Lee Becker and Joshua Southerland for their feedback during the writing process.

References

Sami Baral, Anthony F Botelho, and John A Erickson. 2021. Improving Automated Scoring of Student Open Responses in Mathematics. In *Proceedings of The 14th International Conference on Educational Data Mining (EDM21)*, page 9, Paris, France.

Philip G. Butcher and Sally E. Jordan. 2010. A comparison of human and computer marking of short free-text student responses. *Computers & Education*, 55(2):489–499.

Aoife Cahill, James H Fife, Brian Riordan, Avijit Vajpayee, and Dmytro Galochkin. 2020. Context-based Automated Scoring of Complex Mathematical Responses. In *Proceedings of the Fifteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 186–192, Seattle, WA, USA → Online. Association for Computational Linguistics.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168.*

Denise Dellarosa Cummins, Walter Kintsch, Kurt Reusser, and Rhonda Weimer. 1988. The role of understanding in solving word problems. *Cognitive Psychology*, 20(4):405–438.

Fei Dong, Yue Zhang, and Jie Yang. 2017. Attention-based Recurrent Convolutional Neural Network for Automatic Essay Scoring. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 153–162, Vancouver, Canada. Association for Computational Linguistics.

John A Erickson, Anthony F Botelho, Steven McAteer, Ashvini Varatharaj, and Neil T Heffernan. 2020. The automated grading of student open responses in mathematics. In *Proceedings of the Tenth International Conference on Learning Analytics & Knowledge*, pages 615–624.

James H Fife. 2017. The m-rater™ Engine: Introduction to the Automated Scoring of Mathematics Items. Technical Report ETS RM–17-02.

Scott Hellman, Mark Rosenstein, Andrew Gorman, William Murray, Lee Becker, Alok Baikadi, Jill Budden, and Peter W. Foltz. 2019. Scaling Up Writing in the Curriculum: Batch Mode Active Learning for Automated Essay Scoring. In *Proceedings of the Sixth (2019) ACM Conference on Learning @ Scale*, L@S '19, pages 1–10, New York, NY, USA. Association for Computing Machinery.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *NeurIPS*.

Danqing Huang, Jing Liu, Chin-Yew Lin, and Jian Yin. 2018. Neural math word problem solver with reinforcement learning. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 213–223, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Danqing Huang, Shuming Shi, Chin-Yew Lin, Jian Yin, and Wei-Ying Ma. 2016. How well do Computers Solve Math Word Problems? Large-Scale Dataset Construction and Evaluation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 887–896, Berlin, Germany. Association for Computational Linguistics.

J. C. S. Kadupitiya, Surangika Ranathunga, and Gihan Dias. 2017. Assessment and Error Identification of Answers to Mathematical Word Problems. pages 55–59. ISSN: 2161-377X.

Scott Kirkpatrick, C Daniel Gelatt Jr, and Mario P Vecchi. 1983. Optimization by simulated annealing. *science*, 220(4598):671–680.

Sachin Kumar, Soumen Chakrabarti, and Shourya Roy. 2017. *Earth Mover's Distance Pooling over Siamese LSTMs for Automatic Short Answer Grading*. Pages: 2052.

Nate Kushman, Yoav Artzi, Luke Zettlemoyer, and Regina Barzilay. 2014. Learning to Automatically Solve Algebra Word Problems. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 271–281, Baltimore, Maryland. Association for Computational Linguistics.

Andrew S. Lan, Divyanshu Vats, Andrew E. Waters, and Richard G. Baraniuk. 2015. Mathematical Language Processing: Automatic Grading and Feedback for Open Response Mathematical Questions. In *Proceedings of the Second (2015) ACM Conference on Learning @ Scale*, L@S '15, pages 167–176, New York, NY, USA. Association for Computing Machinery.

Leah S. Larkey. 1998. Automatic essay grading using text categorization techniques. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval - SIGIR '98*, pages 90–95, Melbourne, Australia. ACM Press.

Claudia Leacock and Martin Chodorow. 2003. C-rater: Automated Scoring of Short-Answer Questions. *Computers and the Humanities*, 37(4):389–405.

Nava L Livne, Oren E Livne, and Charles A Wight. 2007. Can Automated Scoring Surpass Hand Grading of Students' Constructed Responses and Error Patterns in Mathematics? *MERLOT Journal of Online Learning and Teaching*, 3(3):12.

Ellis B Page. 1966. The imminence of grading essays by computer. *The Phi Delta Kappan*, 47(5):238–243.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12(Oct):2825–2830.

Lakshmi Ramachandran, Jian Cheng, and Peter Foltz. 2015. Identifying Patterns For Short Answer Scoring Using Graph-based Lexico-Semantic Text Matching. In *Proceedings of the Tenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 97–106, Denver, Colorado. Association for Computational Linguistics.

Brian Riordan, Andrea Horbach, Aoife Cahill, Torsten Zesch, and Chong Min Lee. 2017. Investigating neural architectures for short answer scoring. In *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 159–168, Copenhagen, Denmark. Association for Computational Linguistics.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. In *5th Workshop on Energy Efficient Machine Learning and Cognitive Computing - NeurIPS 2019*, volume abs/1910.01108.

Jia Tracy Shen, Michiharu Yamashita, Ethan Prihar, Neil Heffernan, Xintao Wu, Ben Graff, and Dongwon Lee. 2021. MathBERT:A Pre-trained Language Model for General NLP Tasks in Mathematics Education. In *Math AI For Education Workshop*.

Mark D. Shermis and Jill C. Burstein, editors. 2003. *Automated essay scoring: A cross-disciplinary perspective*. Lawrence Erlbaum Associates, Inc., Mahway, NJ.

Mark D. Shermis and Jill C. Burstein, editors. 2013. *Handbook of automated essay evaluation: Current applications and new directions*. Routledge, New York.

Nina V Stankous. 2016. Constructive response vs. multiple-choice tests in math: American experience and discussion. In *2nd PAN-AMERICAN INTERDISCIPLINARY CONFERENCE, PIC 2016 24-26 February, Buenos Aires Argentina*, page 321.

Lynn Streeter, Jared Bernstein, Peter Foltz, and Donald DeLand. 2011. Pearson's Automated Scoring of Writing, Speaking, and Mathematics. Technical report.

Kaveh Taghipour and Hwee Tou Ng. 2016. A Neural Approach to Automated Essay Scoring. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1882–1891, Austin, Texas. Association for Computational Linguistics.

Luis Tandalla. 2012. Scoring Short Answer Essays. Technical report.

Yan Wang, Xiaojiang Liu, and Shuming Shi. 2017. Deep neural solver for math word problems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 845–854.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le

Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144.

Zhipeng Xie and Shichao Sun. 2019. A Goal-Driven Tree-Structured Neural Model for Math Word Problems. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, pages 5299–5305, Macao, China. International Joint Conferences on Artificial Intelligence Organization.