Math Word Problem Generation with Mathematical Consistency and Problem Context Constraints

Zichao Wang [1] **Andrew S. Lan** [2] **Richard G. Baraniuk** [1,3]

[1] Rice University [2] University of Massachusetts Amherst [3] OpenStax

{zw16,richb}@rice.edu, andrewlan@cs.umass.edu

Abstract

We study the problem of generating arithmetic math word problems (MWPs) given a math equation that specifies the mathematical computation and a context that specifies the problem scenario. Existing approaches are prone to generating MWPs that are either mathematically invalid or have unsatisfactory language quality. They also either ignore the context or require manual specification of a problem template, which compromises the diversity of the generated MWPs. In this paper, we develop a novel MWP generation approach that leverages i) pre-trained language models and a context keyword selection model to improve the language quality of the generated MWPs and ii) an equation consistency constraint for math equations to improve the mathematical validity of the generated MWPs. Extensive quantitative and qualitative experiments on three real-world MWP datasets demonstrate the superior performance of our approach compared to various baselines.

1 Introduction

Math word problems (MWPs) are an important type of educational resource that help assess and improve students' proficiency in various mathematical concepts and skills (Walkington, 2013; Verschaffel et al., 2020). An MWP usually has a corresponding underlying math equation that students will need to identify by parsing the problem and then solve the problem using this equation. An MWP is usually also associated with a "context", i.e., the (often real-world) scenario that the math equation is grounded in, expressed in the question's text. The equation associated with an MWP is often exact and explicit, while the context of the MWP is more subtle and implicit. It is not immediately clear how the context information can be extracted or represented. Table 1 shows an example of an MWP and its associated equation.

Table 1: An examples of MWP and its underlying equation. See Table 2 for more information on the datasets.

MWP: Joan found 70 seashells on the beach . She gave Sam some of her seashells . She has 27 seashells . How many seashells did she give to Sam ?
Equation: `x = (70 - 27)`

In this work, we study the problem of *automatically generating MWPs* from equations and context, which is important for three reasons. First, an automatic MWP generation method can aid instructors and content designers in authoring MWP questions, accelerating the (often costly and labor-intensive) MWP production process. Second, an automated MWP generation method can generate MWPs tailored to each student's background and interests, providing students with a personalized learning experience (Walkington, 2013) that often leads to better engagement and improved learning outcomes (Connor-Greene, 2000; Karpicke, 2012; Karpicke and Roediger, 2008; Koedinger et al., 2015; Kovacs, 2016; Rohrer and Pashler, 2010). Third, an automated MWP generation method can potentially help instructors promote academic honesty among students. While new technologies create new learning opportunities, instructors have growing concerns of technologies that enable students to easily search for answers online without actually solving problems on their own (McCabe et al., 2012; Lancaster and Cotarlan, 2021). Automatically generated MWPs that are unique and previously unseen yet preserve the underlying math components can potentially reduce plagiarism.

In addition to its educational utility, MWP generation is also technically challenging and interesting. An important consideration for MWP generation is *controllability*: in practice, human instructors or content designers often have clear preferences in the type of MWPs they want to use. Therefore, an MWP generation method should be able to generate MWPs that are of high language quality and are textually and mathematically consistent

Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pages 5986–5999
November 7–11, 2021. ©2021 Association for Computational Linguistics

with the given equations and contexts. To date, there exist limited literature on MWP generation. Most prior works focus on automatically answering MWPs, e.g., (Li et al., 2019, 2020; Qin et al., 2020; Shi et al., 2015; Wang et al., 2018a; Roy and Roth, 2015; Wu et al., 2020a) instead of generating them (Nandhini and Balasundaram, 2011; Williams, 2011; Polozov et al., 2015; Deane and Sheehan, 2003). Existing MWP generation methods also often generate MWPs that either are of unsatisfactory language quality or fail to preserve information on math equations and contexts that need to be embedded in them. See Section 4 for a detailed discussion.

1.1 Contributions

In this work, we take a step towards controllable generation of mathematically consistent MWPs with high language quality. Our approach leverages a pre-trained language model (LM) as the base model for improved language quality. The input to the LM is an equation and a context, from which the LM generates an MWP. On top of that, we introduce 2 components that impose constraints on the mathematical and contextual content of the generated MWP. First, to improve mathematical consistency and control over equations, we introduce an equation consistency constraint, which encourages the generated MWP to contain the exact same equation as the one used to generate it. Second, to improve control over contexts, we introduce a context selection model that automatically extracts context from an MWP. Quantitative and qualitative experiments on real-world MWP datasets show that our approach (often significantly) outperforms various baselines on various language quality and math equation accuracy metrics.

2 Methodology

We formulate the task of controllable MWP generation as a conditional generation problem. In this paper, we work with datasets $\mathcal{D} = \{(M_i, E_i)\}_{i=1}^N$ in the form of N (MWP, equation) pairs where M_i and E_i represent MWP and its associated equation, respectively. In the remainder of the paper, we will remove the data point index to simplify notation. This setup assumes each MWP in our dataset is labeled with an underlying equation but its context is unknown. Then, the MWP generation process can be described as

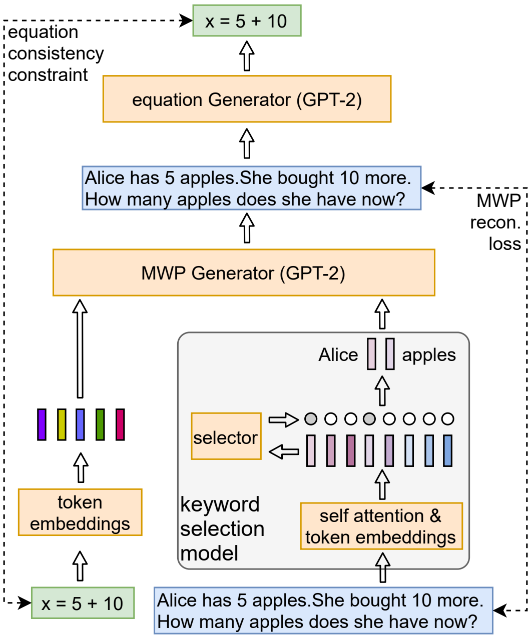
x = 5 + 10
equation consistency constraint
equation Generator (GPT-2)
Alice has 5 apples.She bought 10 more. How many apples does she have now?
MWP recon. loss
MWP Generator (GPT-2)
Alice
apples
selector
token embeddings
keyword selection model
self attention & token embeddings
x = 5 + 10
Alice has 5 apples.She bought 10 more. How many apples does she have now?

Figure 1: An illustration of our MWP generation approach and its key components.

$$M \sim \mathbb{E}_{E\sim\mathcal{D}}[p_\Theta(M|E)] \\ = \mathbb{E}_{E\sim\mathcal{D},\, \mathbf{c}\sim p(\mathbf{c}|E,M)}[p_\Theta(M|E,\mathbf{c})]\,, \qquad (1)$$

where $M = \{m_1, \ldots, m_T\}$ represents the MWP as a sequence of T tokens (e.g., words or wordpieces (Vaswani et al., 2017; Radford et al., 2019)). E and $\mathbf{c}$ are the controllable elements, where $\mathbf{c}$ represents a problem context. p_Θ is the MWP generative model parametrized by a set of parameters Θ.

In this work, we use a pre-trained language model (LM) as the generative model p_Θ, similar to the setup in (Keskar et al., 2019). We choose LMs over other approaches such as sequence-to-sequence (seq2seq) models because they can be pre-trained on web-scale text corpora. Pre-trained LMs thus often generate high-quality text and generalizes well to out-of-domain words not present in the training data. Under an LM, we can further decompose Eq. 1 into

$$p_\Theta(M|E,\mathbf{c}) = \prod_{t=1}^{T} p_\Theta(m|E,\mathbf{c},\{m_s\}_{s=1}^{t-1})\,. \quad (2)$$

To train p_Θ via fine-tuning the LM, we use the usual negative log-likelihood objective:

$$\mathcal{L}_{LM} = \sum_{t=1}^{T} -\log p_\Theta(m_t|E,\mathbf{c},\{m_s\}_{s=1}^{t-1})\,. \quad (3)$$

The above training objective serves as a proxy that optimizes for language quality. However, it alone is unsatisfactory in 2 ways. First, there is no guarantee that the generated MWP is mathematically valid; even if it is, its solution may correspond to an equation that is different from the input equation (Zhou and Huang, 2019). Second, while the context $\mathbf{c}$ can be manually specified, i.e., as a set of keywords, it is unobserved during training and needs to be inferred from data through the costly-to-compute posterior distribution. In the remainder of this section, we introduce our novel approach to tackle these challenges. We first describe our equation consistency constraint that improves the generated MWP's mathematical consistency and then detail our context selection method that learns to extract the context in the form of a set of keywords from an MWP. Figure 1 provides a high-level overview of our overall approach.

2.1 Equation Consistency

We propose an equation consistency constraint to promote the generated MWP to correspond to an equation that is the same as the input equation used to generate the MWP.

To formulate this constraint, we need a model to parse an equation given an MWP, i.e., a mwp2eq model, and a loss function, which we call $\mathcal{L}_{eq}$. The mwp2eq generative process can be written as

$$E' \sim \mathbb{E}_{M' \sim p_\Phi(M|E)}[p_\Phi(E|M')] \,,$$

where p_Φ is the mwp2eq model, E represents an equation, and M' represents the generated MWP. Here, we treat the equation as a sequence of math symbols e_t, making it appropriate for sequential processing. Specifically, we treat each variable (e.g., x, y), math operator (e.g., $=$, $\times$, $+$), and numeric value (e.g., integers, fractions, and decimal numbers) as a single math symbol. Therefore, we can decompose $p_\Phi(E, M')$ similar to Eq. 2. There are ways to represent math equations other than a sequence of symbols, such as symbolic trees (Zanibbi and Blostein, 2012; Davila and Zanibbi, 2017; Mansouri et al., 2019); finding ways to make them compatible to LMs is left for future work. Similar to $\mathcal{L}_{\text{LM}}$, we minimize a negative log-likelihood loss that uses the input equation E as the ground truth for the equation E' parsed from M':

$$\mathcal{L}_{\text{eq}} = \sum_{t=1}^{T} -\log p_\Phi(e_t|M', e_1, \ldots, e_{t-1}) \,. \quad (4)$$

This constraint is reminiscent of the idea of "cycle consistency" that have found success in image and text style transfer (Zhu et al., 2017; Shen et al., 2017), question answering (Yang et al., 2017; Wang et al., 2017a), and disentangled representation learning (Jha et al., 2018).

Gumbel-Softmax Relaxation. To back-propagate loss to p_Θ and compute gradient for Θ, we need the loss $\mathcal{L}_{\text{eq}}$ be differentiable with respect to Θ. The challenge here is that M' is sampled from p_Θ and that this discrete sampling process is non-differentiable, preventing gradient propagation (Nie et al., 2019). To tackle this challenge, we resort to the Gumbel-softmax relaxation (Jang et al., 2017; Maddison et al., 2017) of the discrete sampling process $m_t \sim p_\Theta$. Details are deferred to the Supplementary Materials.

We remark that the gradient derived under the Gumbel-softmax relaxation is a biased but low-variance estimate of the true gradient (Jang et al., 2017; Maddison et al., 2017). The low-variance property makes it more attractive for real applications than other unbiased but high-variance estimators such as REINFORCE (Williams, 1992). We refer to (Jang et al., 2017; Maddison et al., 2017) for more details on the Gumbel-softmax method. In addition, while one can also use deterministic relaxation such as softmax, Gumbel-softmax injects stochastic noise during the training process, which regularizes the model and potentially improves performance; See an empirical comparison in Table 6.

2.2 Context Selection

In practice, we do not have access to the contexts $\mathbf{c}$ during training since they are not specified for real-world MWPs. Therefore, we need ways to specify the context for the MWP generative process. Existing methods characterize context as a "bag-of-keywords", using heuristic methods such as TF-IDF weights to select a subset of tokens as "keywords" from an MWP as its context. These methods are simple but lack flexibility: they either require one to specify the number of tokens to use for each MWP or heuristically select only certain types of tokens (e.g., nouns and pronouns) (Zhou and Huang, 2019; Liu et al., 2020).

In this work, we adopt this "bag-of-tokens" characterization of context, which fits well into LMs, but instead *learn* a context (token) selection method from data. To do so, we interpret $\mathbf{c}$ as a "context keyword selection" variable, i.e., a binary random vector whose dimension is the number of tokens in the vocabulary. Each entry $c^{(i)}$ in $\mathbf{c}$ is an i.i.d. Bernoulli random variable with prior probability ρ, i.e., $p_c(c^{(i)} = 1) = \rho$. Thus, $\mathbf{c}$ acts as a selector that chooses appropriate context tokens from the entire vocabulary. To circumvent the intractable posterior $p(\mathbf{c}|E, M)$, we resort to the auto-encoding variational Bayes (VAE) paradigm (Kingma and Welling, 2013), similar to (Shen et al., 2019). Under the VAE setup, we select a set of tokens conditioned on the MWP as $\mathbf{c} \sim q_\Psi(M)$ where $q_\Psi(M)$ is a proposal distribution, i.e., the keyword selection model.

Context Keyword Selection Model. Given an MWP, we first compute the contextualized embeddings of each token using a simple linear self-attention method as

$$\widetilde{\boldsymbol{m}}_t = \boldsymbol{M}\boldsymbol{a}_t \,, \quad \boldsymbol{a}_t = \mathrm{softmax}\left(\frac{\boldsymbol{M}^\top \boldsymbol{m}_t}{\sqrt{D}}\right),$$

where $\boldsymbol{M} = [\boldsymbol{m}_1, \ldots, \boldsymbol{m}_T] \in \mathbb{R}^{D \times T}$ is the matrix with all token embeddings and D is the embedding dimension. The $\sqrt{D}$ term is added for numerical stability (Vaswani et al., 2017). Then, we compute $q_\Psi^{(i)}(M)$, the probability that each word in the vocabulary is selected as a context keyword, with a single projection layer with Sigmoid activation

$$q_\Psi^{(i)}(M) = \sigma(\mathbf{w}^\top \widetilde{\boldsymbol{m}}_t + b)\, \mathbf{1}_{\{V^{(i)} \in M\}} \,, \tag{5}$$

where $\mathbf{w}$ and b are part of the model parameters Ψ. The indicator function at the end ensures that only tokens that appear in M can be selected as context keywords. In practice, we also mask out stopwords and punctuation; these steps ensure that the context selector selects keywords that are relevant to MWPs and are not too generic.

Optimization Objective. Under the VAE paradigm, we optimize the keyword selection model using the so-called evidence lower bound (ELBO):

$$\mathcal{L}_{\text{VAE}} = \mathcal{L}_{\text{LM}} + \beta \mathcal{L}_{\text{c}} \,, \tag{6}$$

where $\mathcal{L}_{\text{c}} = \mathrm{KL}(q_\Psi \| p_c)\, \mathbf{1}_{\{V^{(i)} \in M\}}$ and the Kullback-Leibler divergence term can be computed analytically thanks to our Bernoulli parametrization. $\mathcal{L}_{\text{c}}$ can be interpreted as a context constraint that prevents the keyword selection model from choosing too many keywords. The hyperparameter β and prior ρ controls the strength of this constraint.

Table 2: Summary statistics of datasets.

Dataset	#MWPs	avg #words per MWP	avg #symbols per eq
arithmetic	1,492	29.89	8.05
MAWPS	2,373	31.25	8.16
Math23K	23,162	35.23	8.78

Because $\mathbf{c}$ is discrete and its sampling process is also non-differentiable, we use the straight-through estimator of the gradient (Bengio et al., 2013) for Θ involved in $\mathcal{L}_{\text{LM}}$ in Eq. 6.

2.3 Training

We train (fine-tune) the LM, the mwp2eq model, and the keyword selection model jointly. The mwp2eq model and keyword selection model are optimized using their respective objectives defined in Eqs. 4 and 6. The overall objective for the MWP generative model p_Θ is

$$\mathcal{L} = \mathcal{L}_{\text{LM}} + \alpha \mathcal{L}_{\text{eq}} + \beta \mathcal{L}_{\text{c}} \,,$$

where $\alpha > 0$ and $\beta > 0$ are hyperparameters that balance these constraint terms.

3 Experiments

We now perform a series of experiments to validate the effectiveness of our proposed MWP generation approach. Quantitatively, we compare our approach to several baselines on various automated language quality and mathematical consistency metrics. Qualitatively, we showcase the capability of our approach in generating controllable, high-quality MWPs.

Datasets. We focus on MWP datasets in which each MWP is associated with a single equation and each equation contains a single unknown variable. Therefore, we consider three such MWP datasets including **Arithmetic** (Hosseini et al., 2014), **MAWPS** (Koncel-Kedziorski et al., 2016), and **Math23K** (Wang et al., 2017b). Table 2 shows summary statistics for each dataset. We follow the preprocessing steps in (Zhou and Huang, 2019) by first replacing all numbers in both MWPs and equations to special tokens `num1`, `num2` etc. and then tokenizing both MWPs and equations into tokens and math symbols, respectively. In addition, we translate Math23K to English because this dataset is originally in Mandarin Chinese. Extension to languages other than English is left for future work.

Other popular MWP datasets such as Algebra (Kushman et al., 2014; Upadhyay and Chang, 2015), Dolphin18K (Huang et al., 2016) and

Table 3: A comparison of language quality and mathematical validity for MWPs generated by our method to various baselines. Numbers in brackets indicate the accuracy of the mwp2eq model trained on each dataset, which is an upper bound on the performance under the ACC-eq metric.

	Arithmetic				**MAWPS**				**Math23K**			
	BLEU-4	METEOR	ROUGE-L	ACC-eq (0.769)	BLEU-4	METEOR	ROUGE-L	ACC-eq (0.755)	BLEU-4	METEOR	ROUGE-L	ACC-eq (0.672)
seq2seq-rnn	0.075	0.152	0.311	0.413	0.153	0.175	0.362	0.472	0.196	0.234	0.444	0.390
+ GloVe	**0.351**	0.310	0.555	0.399	0.592	0.412	0.705	0.585	0.275	0.277	0.507	0.438
seq2seq-tf	0.339	0.298	0.524	0.405	0.554	0.387	0.663	**0.588**	0.301	0.294	0.524	**0.509**
GPT	0.237	0.248	0.455	0.401	0.368	0.294	0.538	0.532	0.282	0.297	0.512	0.477
GPT-pre	0.316	**0.322**	0.554	0.403	0.504	0.391	0.664	0.512	0.325	**0.333**	**0.548**	0.498
ours	0.338	**0.322**	**0.567**	**0.453**	**0.596**	**0.427**	**0.715**	0.557	**0.329**	0.328	0.544	0.505

Table 4: % of generated MWPs that are *not* present in the training data. Our approach generates novel MWPs not seen in the training data most of the time while seq2seq-tf may simply memorize the training data.

	Arithmetic	**MAWPS**	**Math23K**
seq2seq-tf	6.24%	2.49%	38.88%
ours	**94.90%**	**63.77%**	**95.72%**

MathQA[1] (Amini et al., 2019) contain MWPs with multiple equations and many variables, which are challenging to generate even for humans. We leave the more challenging case of generating multi-variable, multi-equation MWPs to future work.

Setup and Baselines. We implement the LM and the mwp2eq models in our approach using pre-trained GPT-2 (Radford et al., 2019); one can also use other models since our approach is agnostic to the specific model architecture. We consider three baselines: **seq2seq-rnn**, a sequence-to-sequence (seq2seq) model using LSTMs with attention that serves as the base architecture in (Zhou and Huang, 2019; Liu et al., 2020); **seq2seq-rnn-glove**, a modification to the previous baseline with GloVe (Pennington et al., 2014) instead of random embeddings at initialization; and **seq2seq-tf**, a seq2seq model with transformers (Vaswani et al., 2017). We also compare our approach to vanilla GPT-2, either randomly initialized or pre-trained; we denote these baselines as **GPT** and **GPT-pre**, respectively. For fair comparison, each baseline takes both equation and a set of keywords chosen by heuristics (see Section C.2) as input to be consistent with the setup in our approach. For each dataset, we perform five-fold cross-validation and report the averaged evaluation results. See the Supplementary Material for more details on the experimental setup and baselines.

Metrics. For language quality, we use the following three evaluation metrics: **BLEU-4** (Papineni et al., 2002), **METEOR** (Lavie and Agarwal, 2007), and **ROUGE-L** (Lin, 2004), following recent literature on question generation (Wang et al., 2018c). We implement these metrics using the package provided by (Chen et al., 2015). For mathematical consistency, We use the equation accuracy (**ACC-eq**) metric that measures whether the generated MWP is mathematically consistent with the controlled input equation. The idea of this metric originates from other applications such as program translation and synthesis (Chen et al., 2018b, 2020). In our case, because the equation associated with a generated MWP is not readily available, we resort to a mwp2eq model fine-tuned on each MWP dataset to predict the equation from an MWP. During the evaluation, we feed the generated MWP to the mwp2eq model as input and check whether the output of the mwp2eq model exactly matches the equation used as input to the MWP generator.

3.1 Quantitative Results

Table 3 shows the quantitative results of our experiments. The number in parenthesis below ACC-eq is the equation accuracy when we feed the mwp2eq model the ground-truth MWPs in the respective datasets. We see that our approach outperforms the best baseline on most occasions, especially on language quality metrics. However, there are a few exceptions, especially for the ACC-eq metric on the Math23K dataset. Specifically, we note that the seq2seq-tf baseline seems to yield an ACC-eq value even higher than the oracle accuracy at the first attempt. Upon closer investigation, we find that the baseline seq2seq models, especially the seq2seq-tf baseline, simply memorize the training data. Table 4 illustrates this finding and shows the percentage of generated MWPs that are *not* present in the training data. We see that the seq2seq-tf base-

[1] MathQA is the most difficult MWP dataset we have encountered, which containing GRE and GMAT level questions.

Table 5: Generated MWP examples with fixed context and varying equations.

Context: `candies`	
Equation #1: `x = num1 + num2`	**Equation #2: `x = num1 - num2`**
seq2seq-tf: ethan has num1 presents . alissa has num2 more than ethan . how many presents does alissa have ? (in training data) **GPT-pre**: There are num1 scissors in the drawer. Keith placed num2 scissors in the drawer. How many scissors are now there in total? (irrelevant to context) **ours**: Mildred collects num1 candies. Mildred's father gives Mildred num2 more. How many candies does Mildred have? (✓)	**seq2seq-tf**: mildred weighs num1 pounds . carol weighs num2 pounds . how much heavier is mildred than carol ? (in training data) **GPT-pre**: Joan has num1 blue balloons but lost num2 of them. How many blue balloons does Joan have now? (irrelevant to context) **ours**: There are num1 candies in the jar. num2 are eaten by a hippopotamus. How many candies are in the jar? (✓)
Equation #3: `x = num1 * num2`	**Equation #4: `x = num1 / num2`**
seq2seq-tf: each banana costs $ num1 . how much do num2 bananas cost ? (in training data) **GPT-pre**: Joan has saved num1 quarters from washing cars. How many cents does Joan have? (inconsistent with equation) **ours**: Each child has num1 candies. If there are num2 children, how many candies are there in all? (✓)	**seq2seq-tf**: there are num1 bananas in diane ' s banana collection . if the bananas are organized into num2 groups , how big is each group ? (in training data) **GPT-pre**: Joan has num1 blue marbles. Sandy has num2 times more blue marbles than Melanie. How many blue marbles does Joan have? (inconsistent with equation) **ours**: There are num1 candies in the candy collection. If the candies are organized into num2 groups, how big is each group? (✓)

Table 6: Results of the ablation study, which validate the effectiveness of each component in our approach.

	Arithmetic		**MAWPS**		**Math23K**	
	BLEU-4	ACC-eq	BLEU-4	ACC-eq	BLEU-4	ACC-eq
$\mathcal{L}_{\text{eq}}$ (softmax)	0.110	0.417	0.308	**0.555**	0.284	0.466
$\mathcal{L}_{\text{eq}}$ **(Gumbel-softmax)**	**0.303**	**0.455**	**0.522**	0.527	**0.306**	**0.495**
keyword, TF-IDF	0.313	**0.424**	0.518	0.536	0.310	0.498
keyword, noun+pronoun	0.316	0.413	0.504	0.512	**0.325**	0.498
context selection	**0.320**	0.412	**0.533**	**0.542**	0.324	**0.501**
full model w/o $\mathcal{L}_{\text{c}}$	0.303	**0.455**	0.522	0.527	0.306	0.495
full model w/o $\mathcal{L}_{\text{eq}}$	0.320	0.412	0.491	0.500	0.324	0.501
full model w/o both	0.316	0.403	0.504	0.512	0.325	0.498
full model	**0.338**	0.453	**0.596**	**0.557**	**0.332**	**0.513**

line tends to directly copy MWPs from the training data as its "generated" MWPs, especially on the 2 smaller datasets. In contrast, our approach generates novel MWPs most of the time. We thus report ACC-eq only on the novel MWPs generated by the seq2seq-tf baseline on the Math23K dataset. Our approach outperforms seq2seq-tf on this modified ACC-eq metric.

Ablation Study. To validate that each component in our approach contributes to its success, we conduct an ablation study and compare our approach with several variants and several baselines after removing some of these components. For the use of Gumbel-softmax in the equation consistency constraint computation, we compare to softmax (Goodfellow et al., 2016), which removes sampling from the Gumbel variable. For the context keyword selection model, we compare to several context keyword selection heuristics including TF-IDF (Jones, 1972; Zhou and Huang, 2019) and nouns+pronouns; see the Supplementary Material for more details on these baselines. Table 6 shows the ablation study results, reporting on BLEU-4 and ACC-eq as the representative metric for language quality and mathematical consistency, respectively. These comparisons validate the necessity of each component in our approach: Gumbel-softmax outperforms softmax and our context keyword selection method outperforms other heuristic methods. We also see that our approach outperforms variants with either component removed and that the equation consistency constraint and the context keyword selection method tend to improve the mathematical consistency and language quality of the generated MWPs, respectively.

3.2 Qualitative Results

Since seq2seq baselines outperform our approach on a few occasions under the automated metrics, we now conduct a few case studies to investigate each approach. We investigate i) how controllable is each approach by giving it different input equations and contexts and ii) how generalizable each approach is by giving it unseen contexts in the dataset. Specifically, we conduct two qualitative experiments: First, we hold an input context fixed and change the input equation; Second, we hold the input equation fixed and change the context. We compare the MWPs generated by our approach to those generated by the seq2seq-tf and GPT-pre baselines trained on the MAWPS dataset, where these baselines perform well under automated met-

Table 7: Generated MWP examples with novel context not present in the training data.

Equation: `x = num1 + num2 + num3`	
Context #1: `violin piano acoustic guitar`	**Context #2: `beets eggplant`**
seq2seq-tf: sara grew num1 onions , sally grew num2 onions , and fred grew num3 onions . how many onions did they grow in all ? (in training data)	**seq2seq-tf**: sara grew num1 onions , sally grew num2 onions , and fred grew num3 onions . how many onions did they grow in all ? (in training data)
GPT-pre: There are num1 dogwood trees currently in the park. Park workers will plant num2 dogwood trees today and num3 dogwood trees tomorrow. How many dogwood trees will the park have when the workers are finished? (irrelevant to context)	**GPT-pre**: There are num1 orchid bushes currently in the park. Park workers will plant num2 orchid bushes today and num3 orchid bushes tomorrow. How many orchid bushes will the park have when the workers are finished? (irrelevant to context)
ours: Mike joined his school's band. He bought a clarinet for \$ num1, a music stand for \$ num2, and a song book for \$ num3. How much did Mike spend at the music store? (✓)	**ours**: Sara grew num1 beets, Sally grew num2 beets, and Fred grew num3 beets. How many beets did they grow in total? (✓)

rics (see Table 3). The Supplementary Material contains additional qualitative examples.

Fixed Context, Changing Equation. Table 5 shows the MWPs generated by each approach using the same input context and different input equations. We see that every approach can generate MWPs with high language quality and are mathematically valid most of the time. However, upon closer inspection, we find that MWPs generated by the seq2seq-tf baseline are often exact copies of those it has seen in the training data. In other words, the model does nothing more than memorizing the training data and retrieving the most relevant one given the input equation and context; see Table 4 for a numeric comparison and the discussion in Section 3.1. This observation is not surprising because training only on small MWP datasets leads to overfitting. It also explains why the seq2seq baselines perform well on the automated metrics since these MWP datasets contain problems that lack language diversity, which results in many overlapping words and phrases that often appear in both the training and validation sets. The GPT-pre baseline, on the other hand, is sometimes capable of generating novel MWPs, but they are either irrelevant to the input context or are inconsistent with the input equation. Only our approach consistently generates MWPs that are both novel and mathematically consistent with the input equation.

Fixed Equation, Changing Context. Table 7 shows the MWPs generated by each approach using the same input equation and different input contexts. The keywords in these contexts are not part of the vocabulary of the training set and are thus unseen by the model during training/fine-tuning. Similar to the results of the previous experiment, here we also see that the seq2seq baseline simply retrieves an MWP from the training dataset as its "generated" MWP. This observation is unsurprising for the seq2seq baseline because it simply converts an out-of-vocabulary word in the input context into a special `unknown` token, which is uninformative. Interestingly, the GPT-pre baseline also generates MWPs that have minimal difference from MWPs in the training set or seems to ignore the input context. We again attribute this phenomenon to the small dataset size, on which the model also overfits if no additional constraints are introduced. Once again, in this setting, only our approach consistently generates novel and high-quality MWPs that are relevant to the input context.

Table 8: Examples of the keywords that are selected from a (possibly long) input context.

MWP: Emily collects num1 cards . Emily ' s father gives Emily num2 more . Bruce has apples . How many cards does Emily have ? **Context keywords:** Emily cards collects father
MWP: The school cafeteria had num1 apples . If they used num3 to make lunch for the students and then bought num2 more , how many apples would they have ? **Context keywords:** apples cafeteria

Selected Context Keywords. To investigate our context keyword selection model, we show in Table 8 a few examples of the input context (which is the original MWP in our training setting) and the selected context keywords, i.e., those with $c^{(i)} > 0.5$ (recall Eq. 5). We see that our context keyword selection model can identify components that are key to the relevant underlying mathematical components in the MWP; for example, it identifies only "Emily collects cards father" as the key to this MWP and ignores the part with "Bruce apples", which is unrelated to the math equation. Such a context keyword selection method is useful in practice to summarize (possibly long) input contexts provided by human instructors/content designers.

4 Related Work

MWP Generation and Answering. Earlier works on MWP generation do so in a highly structured way, explicitly relying on domain knowledge and even pre-defined equation and text templates (Nandhini and Balasundaram, 2011; Williams, 2011; Polozov et al., 2015; Deane and Sheehan, 2003). More recently, neural network-based approaches have shown significant advantages in generating high-quality questions compared to template-based approaches. Recent approaches on MWP generation also take this approach, usually using recurrent neural networks in a seq2seq pipeline (Zhou and Huang, 2019; Liu et al., 2020). Instead of focusing on building new datasets or specific model architectures, we tackle the MWP generation problem from a controllable generation perspective, where we focus on the generated MWPs' language quality and mathematical consistency. This focus leads to our proposed approach that specifically aims at tackling these two challenges; our framework is model-agnostic and can be combined with almost any existing MWP generation approach.

Our approach also involves a model (mwp2eq) that parses an MWP into its underlying equation, which has been a very active research area with a plethora of related work, e.g., (Huang et al., 2018; Chiang and Chen, 2019; Xie and Sun, 2019; Zou and Lu, 2019; Li et al., 2019, 2020; Qin et al., 2020; Shi et al., 2015; Wang et al., 2018b,a; Roy and Roth, 2015; Wang et al., 2019a; Amini et al., 2019; Wu et al., 2020a). In this work, we simply use a pre-trained LM as the mwp2eq model; investigation of leveraging the above recent advances to improve mathematical consistency of the generated MWPs is left for future work.

Controllable Text Generation. Our work is also related to a growing body of literature on controllable text generation (Prabhumoye et al., 2020; Wang et al., 2019b; Hu et al., 2017; Keskar et al., 2019; Shen et al., 2019). In particular, our equation consistency constraint takes inspiration from the above works that impose similar constraints to improve control over the generation process. A major difference between our work and most of these prior works is that, in most of these approaches, the control elements, such as emotion, sentiment, and speaker identity, are usually represented as scalar numerical values. In contrast, our control elements (equation and context) consist of a sequence of math symbols or tokens rather than numeric values, which requires additional technical solutions to propagate gradient. The Gumbel-softmax trick (Jang et al., 2017; Maddison et al., 2017) that we employ has found success in text generation using generative adversarial networks (GANs) (Kusner and Hernández-Lobato, 2016; Chen et al., 2018a; Nie et al., 2019; Wu et al., 2020b; Jiao and Ren, 2021), a setting similar to ours where discrete sampling becomes an issue.

5 Conclusions and Future Work

In this paper, we developed a controllable MWP generation approach that (i) leverages pre-trained language models to improve language quality of the generated MWP, (ii) imposes an equation consistency constraint to improve mathematical consistency of the generated MWP, and (iii) includes a context selector that sets the context (in the form of a set of keywords) to use in the generation process. Experimental results on several real-world MWP datasets show that, while there is plenty of room for improvement, our approach outperforms existing approaches at generating mathematically consistent MWPs with high language quality.

Automatically generating MWPs remains a challenging problem and our work opens up many avenues for future work. First, our study is limited to the case of simple MWPs, each with a single equation and variable. While results are encouraging, our approach does not generalize well to the more challenging case when the input consists of multiple, complex equations. In these cases, we need more informative representations of the input equations (Wang et al., 2021). Second, there is no clear metric that can be used to evaluate the generated MWPs, especially their mathematical validity. It is not uncommon when a generated MWP with high scores under our metrics is either unanswerable or inconsistent with the input equation. Therefore, future work should also focus on developing metrics for better evaluation of generated MWPs' mathematical validity. Last but not least, while we focus on 2 control elements (equation, context), an interesting future direction is to add more control elements to the generation process such as question difficulty and linguistic complexity.

Acknowledgements

This work is supported by NSF grants 1842378, 1937134, and 1917713, ONR grant N0014-20-1-2534, AFOSR grant FA9550-18-1-0478, and a Vannevar Bush Faculty Fellowship, ONR grant N00014-18-1-2047.

References

Aida Amini, Saadia Gabriel, Shanchuan Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. 2019. MathQA: Towards interpretable math word problem solving with operation-based formalisms. In *Proc. NAACL*, pages 2357–2367.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. 2013. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv:1308.3432*.

Liqun Chen, Shuyang Dai, Chenyang Tao, Dinghan Shen, Zhe Gan, Haichao Zhang, Yizhe Zhang, Ruiyi Zhang, Guoyin Wang, and Lawrence Carin. 2018a. Adversarial text generation via feature-mover's distance. In *Proc. NeurIPS*, page 4671–4682.

Xinlei Chen, Hao Fang, Tsung-Yi Lin, Ramakrishna Vedantam, Saurabh Gupta, Piotr Dollar, and C. Lawrence Zitnick. 2015. Microsoft COCO Captions: Data Collection and Evaluation Server. *arXiv:1504.00325*.

Xinyun Chen, Chen Liang, Adams Wei Yu, Denny Zhou, Dawn Song, and Quoc V. Le. 2020. Neural symbolic reader: Scalable integration of distributed and symbolic representations for reading comprehension. In *Proc. ICLR*.

Xinyun Chen, Chang Liu, and Dawn Song. 2018b. Tree-to-tree neural networks for program translation. In *Proc. NeurIPS*, page 2552–2562.

Ting-Rui Chiang and Yun-Nung Chen. 2019. Semantically-aligned equation generation for solving and reasoning math word problems. In *Proc. NAACL*, pages 2656–2668.

Patricia A. Connor-Greene. 2000. Assessing and promoting student learning: Blurring the line between teaching and testing. *Teaching Psychol.*, 27(2):84–88.

Kenny Davila and Richard Zanibbi. 2017. Layout and semantics: Combining representations for mathematical formula search. In *Prof. Intl. ACM SIGIR Conf. Res. Develop. Info. Retrieval*, page 1165–1168.

Paul Deane and Kathleen Sheehan. 2003. Automatic item generation via frame semantics: Natural language generation of math word problems. Technical report, ERIC.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press.

Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. 2014. Learning to solve arithmetic word problems with verb categorization. In *Proc. EMNLP*, pages 523–533.

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P. Xing. 2017. Toward controlled generation of text. In *Proc. ICML*, page 1587–1596.

Danqing Huang, Jing Liu, Chin-Yew Lin, and Jian Yin. 2018. Neural math word problem solver with reinforcement learning. In *Proc. ACL*, pages 213–223.

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 *Proc. ACL*, pages 887–896.

Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical reparameterization with gumbel-softmax. In *Proc. ICLR*.

Ananya Harsh Jha, Saket Anand, Maneesh Singh, and VSR Veeravasarapu. 2018. Disentangling factors of variation with cycle-consistent variational autoencoders. In *Proc. ECCV*.

Ziyun Jiao and Fuji Ren. 2021. WRGAN: Improvement of RelGAN with wasserstein loss for text generation. *Electronics*, 10(3):275.

Karen S. Jones. 1972. A statistical interpretation of term specificity and its application in retrieval. *J. Documentation*, 28(1):11–21.

Jeffery D. Karpicke. 2012. Retrieval-based learning: Active retrieval promotes meaningful learning. *Current Directions Psychol. Sci.*, 21(3):157–163.

Jeffery D. Karpicke and Henry L. Roediger. 2008. The critical importance of retrieval for learning. *Science*, 319(5865):966–968.

Nitish Shirish Keskar, Bryan McCann, Lav Varshney, Caiming Xiong, and Richard Socher. 2019. CTRL - A Conditional Transformer Language Model for Controllable Generation. *arXiv:1909.05858*.

Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv:1312.6114*.

Kenneth R. Koedinger, Jihee Kim, Julianna Zhuxin Jia, Elizabeth A. McLaughlin, and Norman L. Bier. 2015. Learning is not a spectator sport: Doing is better than watching for learning from a mooc. In *Proc. Conf. Learn. Scale*, pages 111–120.

Rik Koncel-Kedziorski, Subhro Roy, Aida Amini, Nate Kushman, and Hannaneh Hajishirzi. 2016. MAWPS: A math word problem repository. In *Proc. NAACL*, pages 1152–1157.

Geza Kovacs. 2016. Effects of in-video quizzes on mooc lecture viewing. In *Proc. Conf. Learn. Scale*, pages 31–40.

Nate Kushman, Yoav Artzi, Luke Zettlemoyer, and Regina Barzilay. 2014. Learning to automatically solve algebra word problems. In *Proc. ACL*, pages 271–281.

Matt J. Kusner and José Miguel Hernández-Lobato. 2016. Gans for sequences of discrete elements with the gumbel-softmax distribution. *arXiv:1611.04051*.

Thomas Lancaster and Codrin Cotarlan. 2021. Contract cheating by stem students through a file sharing website: a covid-19 pandemic perspective. *Int. J Educ. Integrity*, 17(1):3.

A. Lavie and A. Agarwal. 2007. Meteor: An automatic metric for MT evaluation with high levels of correlation with human judgments. In *Proc. Workshop Statistical Mach. Transl.*, pages 228–231.

Jierui Li, Lei Wang, Jipeng Zhang, Yan Wang, Bing Tian Dai, and Dongxiang Zhang. 2019. Modeling intra-relation in math word problems with different functional multi-head attentions. In *Proc. ACL*, pages 6162–6167.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016. A diversity-promoting objective function for neural conversation models. In *Proc. NAACL*, pages 110–119.

Shucheng Li, Lingfei Wu, Shiwei Feng, Fangli Xu, Fengyuan Xu, and Sheng Zhong. 2020. Graph-to-tree neural networks for learning structured input-output translation with applications to semantic parsing and math word problem. In *Proc. EMNLP*, pages 2841–2852.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Proc. Workshop Text Summarization Branches Out*, pages 74–81.

Tianqiao Liu, Qian Fang, Wenbiao Ding, and Zitao Liu. 2020. Mathematical word problem generation from commonsense knowledge graph and equations. *arXiv:2010.06196*.

Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. 2017. The concrete distribution: A continuous relaxation of discrete random variables. In *Proc. ICLR*.

Behrooz Mansouri, Shaurya Rohatgi, Douglas W. Oard, Jian Wu, C. Lee Giles, and Richard Zanibbi. 2019. Tangent-cft: An embedding model for mathematical formulas. In *Proc. Intl. ACM SIGIR Conf. Res. Develop. Info. Retrieval*, page 11–18.

Donald L. McCabe, Kenneth D. Butterfield, and Linda K. Treviño. 2012. *Cheating in college: Why students do it and what educators can do about it*. The Johns Hopkins University Press.

K. Nandhini and S. R. Balasundaram. 2011. Math word question generation for training the students with learning difficulties. In *Proc. Int. Conf. Workshop Emerg. Trends Technol.*

Weili Nie, Nina Narodytska, and Ankit Patel. 2019. RelGAN: Relational generative adversarial networks for text generation. In *Proc. ICLR*.

K. Papineni, S. Roukos, T. Ward, and W. Zhu. 2002. BLEU: A method for automatic evaluation of machine translation. In *Proc. ACL*, pages 311–318.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proc. EMNLP*, pages 1532–1543.

Oleksandr Polozov, Eleanor O'Rourke, Adam M. Smith, Luke Zettlemoyer, Sumit Gulwani, and Zoran Popovic. 2015. Personalized mathematical word problem generation. In *Proc. AAAI*, page 381–388.

Shrimai Prabhumoye, Alan W Black, and Ruslan Salakhutdinov. 2020. Exploring controllable text generation techniques. In *Proc. ACL*, pages 1–14.

Jinghui Qin, Lihui Lin, Xiaodan Liang, Rumin Zhang, and Liang Lin. 2020. Semantically-aligned universal tree-structured solver for math word problems. In *Proc. EMNLP*, pages 3780–3789.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Doug Rohrer and Harold Pashler. 2010. Recent research on human learning challenges conventional instructional strategies. *Educational Researcher*, 39(5):406–412.

Subhro Roy and Dan Roth. 2015. Solving general arithmetic word problems. In *Proc. EMNLP*, pages 1743–1752.

Tianxiao Shen, Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2017. Style transfer from non-parallel text by cross-alignment. In *Proc. NeurIPS*, volume 30.

Xiaoyu Shen, Jun Suzuki, Kentaro Inui, Hui Su, Dietrich Klakow, and Satoshi Sekine. 2019. Select and attend: Towards controllable content selection in text generation. In *Proc. EMNLP-IJCNLP*, pages 579–590.

Shuming Shi, Yuehui Wang, Chin-Yew Lin, Xiaojiang Liu, and Yong Rui. 2015. Automatically solving number word problems by semantic parsing and reasoning. In *Proc. EMNLP*, pages 1132–1142.

Shyam Upadhyay and Ming-Wei Chang. 2015. Draw: A challenging and diverse algebra word problem set. Technical report, Microsoft.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proc. NeurIPS*, volume 30.

Lieven Verschaffel, Stanislaw Schukajlow, Jon Star, and Wim Van Dooren. 2020. Word problems in mathematics education: a survey. *ZDM*, 52(1):1–16.

Candace A. Walkington. 2013. Using adaptive learning technologies to personalize instruction to student interests: The impact of relevant contexts on performance and learning outcomes. *J. Educ. Psychol.*, 105(4):932–945.

Lei Wang, Yan Wang, Deng Cai, Dongxiang Zhang, and Xiaojiang Liu. 2018a. Translating a math word problem to a expression tree. In *Proc. EMNLP*, pages 1064–1069.

Lei Wang, Dongxiang Zhang, Lianli Gao, Jingkuan Song, Long Guo, and Heng Tao Shen. 2018b. Mathdqn: Solving arithmetic word problems via deep reinforcement learning. In *Proc. AAAI*, pages 5545–5552.

Lei Wang, Dongxiang Zhang, Jipeng Zhang, Xing Xu, Lianli Gao, Bing Tian Dai, and Heng Tao Shen. 2019a. Template-based math word problem solvers with recursive neural networks. In *Proc. AAAI*, volume 33, pages 7144–7151.

Tong Wang, Xingdi Yuan, and Adam Trischler. 2017a. A joint model for question answering and question generation. *arXiv:1706.01450*.

Wenlin Wang, Zhe Gan, Hongteng Xu, Ruiyi Zhang, Guoyin Wang, Dinghan Shen, Changyou Chen, and Lawrence Carin. 2019b. Topic-guided variational auto-encoder for text generation. In *Proc. NAACL*, pages 166–177.

Yan Wang, Xiaojiang Liu, and Shuming Shi. 2017b. Deep neural solver for math word problems. In *Proc. EMNLP*, pages 845–854.

Zichao Wang, Andrew S. Lan, and Richard G. Baraniuk. 2021. Mathematical formula representationvia tree embeddings.

Zichao Wang, Andrew S. Lan, Weili Nie, Andrew E. Waters, Phillip J. Grimaldi, and Richard G. Baraniuk. 2018c. Qg-net: A data-driven question generation model for educational content. In *Proc. Fifth Annu. ACM Conf. Learn. at Scale*.

Ronald J. Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.*, 8(3–4):229–256.

Sandra Williams. 2011. Generating mathematical word problems. In *Proc. AAAI*, volume FS-11-04.

Qinzhuo Wu, Qi Zhang, Jinlan Fu, and Xuanjing Huang. 2020a. A knowledge-aware sequence-to-tree network for math word problem solving. In *Proc. EMNLP*, pages 7137–7146.

Yue Wu, Pan Zhou, Andrew G Wilson, Eric Xing, and Zhiting Hu. 2020b. Improving gan training with probability ratio clipping and sample reweighting. In *Proc. NeurIPS*, volume 33, pages 5729–5740.

Zhipeng Xie and Shichao Sun. 2019. A goal-driven tree-structured neural model for math word problems. In *Proc. IJCAI*.

Zhilin Yang, Junjie Hu, Ruslan Salakhutdinov, and William Cohen. 2017. Semi-supervised QA with generative domain-adaptive nets. In *Proc. ACL*, pages 1040–1050.

Richard Zanibbi and Dorothea Blostein. 2012. Recognition and retrieval of mathematical expressions. *Intl. J. Document Anal. Recognit.*, 15(4):331–357.

Qingyu Zhou and Danqing Huang. 2019. Towards generating math word problems from equations and topics. In *Proc. Int. Conf. Natural Lang. Gener.*, pages 494–503.

Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. 2017. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proc. ICCV*, pages 2242–2251.

Yanyan Zou and Wei Lu. 2019. Text2Math: End-to-end parsing text into math expressions. In *Proc. EMNLP-IJCNLP*, pages 5327–5337.

Supplementary Material for

Math Word Problem Generation with Mathematical Consistency and Problem Context Constraints

A Gumbel-Softmax in Section 2.1

We describe in detail the procedure to approximate sampling M' from p_Φ, i.e., sampling discrete tokens $m'_t \sim p_\Phi(m_t|E, \mathbf{c}, m_1, \ldots, m_{t-1})$, using the Gumbel-softmax relaxation. In the first step, we reparametrize sampling from a categorical distribution p_Θ using the Gumbel-max trick (Maddison et al., 2017) as follows:

$$
\begin{aligned}
&u^{(i)} \sim \text{uniform}(0, 1)\,, \\
&g_t^{(i)} = -\log(-\log(u^{(i)}))\,, \\
&m_t = \text{one\_hot}\Big(\operatorname*{argmax}_{i \in |V|}\big(f_{\Theta,t}^{(i)} + g_t^{(i)}\big)\Big)\,,
\end{aligned}
$$

where $|V|$ is the size of the vocabulary, $f_{\Theta,t}^{(i)}$ is the pre-softmax activation of p_Θ at the t-th generation step for the i-th word, and $g_t^{(i)}$ are i.i.d. samples from the standard Gumbel distribution.

In the second step, we approximate the discrete argmax operator with the continuous, differentiable softmax operator, which enables us to obtain the final approximation

$$
m'_t = \text{softmax}((f_{\Theta,t} + g_t)/\tau)\,,
$$

where τ is a temperature hyperparameter, resulting in the Gumbel-softmax distribution. When τ approaches 0, this approximation approaches the categorical distribution parametrized by $\text{one\_hot}\big(\text{argmax}_{i \in |V|}(f_{\Theta,t})\big)$.

B Quality of the Math23K Dataset

Some reviewers brought up a concern on the quality of the Math23K dataset because it is originally in Chinese and we use the English-translated version (via Google Translate API) of this dataset. Despite using an automated translation service, we find that most data points in the translated Math23K dataset is good enough to use for training and evaluation. Figure A reports the perplexity score under a small GPT-2 model for each dataset, averaged over all data points. We see that the translated Math23K dataset has comparable perplexity compared to that of the other two datasets. This observation suggests that the translated Math23K dataset has similar language quality compared to the other two datasets that are originally in English.

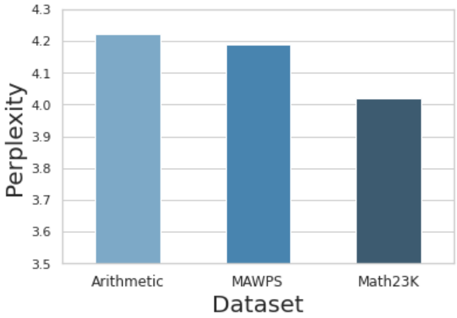

Figure A: Averaged perplexity of each dataset under a small GPT-2. The translated Math23K dataset has similar perplexity compared to the other two datasets, suggesting similar language quality of the three datasets.

C Experiment Details

C.1 Training Details

We train the three components in our method jointly. Specifically, we first jointly train the context keyword selection model and the MWP generator. After that, we freeze the context selection model and continue to jointly train the MWP generator and the mwp2eq model. Notably, the input token embeddings to the context selector are the same as the pre-trained GPT-2 token embeddings. These embeddings are kept fixed throughout training for training stability.

Table A provides the configurations for all models under consideration. The numbers marked with an asteroid (*) are the setting for the Math23K dataset. The number marked with a dagger (†) is the configuration for our approach. For all baselines, we use the noun+pronoun words (see Section C.2) extracted from the MWPs as the input context. Each model are trained on a single NVIDIA RTX 8000 GPU. For the GPT-based models, including our approach, the training time ranging from 1.3 minute per epoch (for the Arithmetic and MAWPS datasets) to 20 min per epoch (for the Math23K dataset). The much slower runtime for the Math23K dataset is due to both its smaller batch size in order to fit into memory and its large size, which is almost 20 times as big as the Arithmetic dataset. For the seq2seq baselines, the training speed is about 7 seconds per 10 steps (equivalent of 640 data points). The seq2seq baselines are implemented using the OpenNMT framework.[2] The

[2] https://opennmt.net/

Table A: Model configurations.

architecture	#layers	input size	layer size	#params	optimizer	learning rate	batch size	training epochs/steps
seq2seq-rnn	2	300	512	11M	adagrad	0.15	64	{5000, 15000*}
seq2seq-attn-rnn	1	300	512	11M	adagrad	0.15	64	{5000, 15000*}
seq2seq-transformer	6	512	512	52M	Adam	2	4096	{5000, 15000*}
GPT	36	1280	1280	774M	Adam	5e-5	8	20
ours	36	1280	1280	774M	Adam	5e-5	{8*, 16}	{5, $15^\dagger$}

Table B: Additional examples of MWPs generated by our approach.

Equation	`x = num1 / num2`
Context	`Sue bag bags cookies fill mother`
Gen. MWP	Sue ' s mom baked num1 cookies. If she wants to distribute the cookies among num2 children, how many cookies will each child get?
Equation	`x = num1 + num2 + num3`
Context	`Charlie Cortland bag bags fruit pick picked visit`
Gen. MWP	Cortland picked num1 pears, and Mike picked num2 pears, and Alyssa picked num3 pears from the pear tree. How many pears were picked in total?
Equation	`x = num1 - num2`
Context	`cousin game playing points scored video`
Gen. MWP	Zach scored num1 points in the football game. Ben scored num2 points. How many more points did Zach score than Ben?
Equation	`x = num1 + num2`
Context	`teacher worksheet`
Gen. MWP	The secretary prints a copy of the worksheet, num1 copies in the morning, and num2 copies in the afternoon. How many copies were printed throughout the day?
Equation	`x = ( num1 + num2 )`
Context	`mike baseball football marble total toy`
Gen. MWP	The total cost of a toy factory to produce a football is num1 yuan, which is num2 yuan less than the total cost. How much is the total cost?
Equation	`x = ( num1 * num2 )`
Context	`anne hour mile`
Gen. MWP	It takes num2 hours for a car to travel num1 kilometers per hour from A to B. How many kilometers are the distance between A and B?

pre-trained GPT-2 model is loaded from the HuggingFace repository.[3]

When training all models, we also randomly drop words in the context with a probability of 0.3 for each word and permute the order of these words. We do so to improve the models' robustness with respect to the number and the order of the keywords in the context. In addition, for the seq2seq baselines, we additionally lowercase all string to reduce the total number of vocabulary.

C.2 Baselines for the Ablation Study

Here, we explain the different method to select keywords. We first tokenize[4] the MWPs. For the noun+pronoun method, we extract words that have "noun" or "pronoun" as their part-of-speech tags. Stopwords and punctuation are not included. For the TF-IDF method, we compute the TF-IDF weights for all tokens, again excluding stopwords and punctuation, and then choose 5 words with the highest weights for each MWP as its context.

C.3 Mathematical Consistency Metric

In principle, A more accurate mwp2eq model leads to more accurate mathematical consistency evaluation and many other state-of-the-art mwp2eq methods, including those targeting automatic MWP answering that we reviewed in Section 4, can be employed. We have observed that fine-tuning pre-trained GPT-2 achieves competitive performance comparing to a number of existing approaches and thus use it for this present work. Using more ad-

[3] https://huggingface.co/

[4] https://spacy.io/

Table C: Examples of failed cases.

Equation	`x = ( num1 / num2)`
Context	`Tom cars dollars money week weekend`
Gen. MWP	On Sunday, num1 yuan can buy num2 cars. So how much money is needed to buy a car?
Error	Incomplete information: implicitly assumes each car costs the same. Can be remedied by adding “on average”.)
Equation	`x = ( num1 * num2 )`
Context	`Tom box boxes brother candy chocolate pieces`
Gen. MWP	There are num2 boxes of chocolates in the candy store, and the price is num1 yuan per piece. How much does it cost to buy a piece of chocolate?
Error	Wrong question asked. Can be remedied by changing the “a piece of chocolate” to “those chocolates”.
Equation	`x = ( num1 * num2 )`
Context	`David box dog dogs dollars toy`
Gen. MWP	A toy dog is num1 yuan, and the price of a puppy is num2 times that of a puppy. How much is a puppy?
Error	Incoherent question statement. Can be remedied by changing the second “puppy” to “dog”.

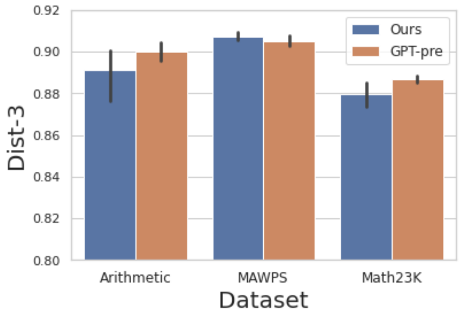

Figure B: Diversity of generation comparing our approach with a fine-tuned pre-trained GPT-2. Our approach achieves similar generation diversity according to the Dist-3 metric.

vanced methods to improve the mathematical consistency evaluation is left for future work.

D Additional Results

D.1 Generation Diversity

Per the reviewers request, we compare the generation diversity using the Dist-3 metric (Li et al., 2016) in Figure B, where higher numbers indicating more diversity. We can observe that our approach achieves similar generation diversity across all datasets compared to GPT-2, with differences smaller than 0.1, suggesting our regularizations do not compromise the generation diversity.

D.2 Additional Qualitative Examples

Table B presents additional examples of MWPs generated by our approach. The contexts and equations in the first three rows and the last three rows are taken from the Arithmetic and the Math23K datasets, respectively. These examples are consistent with the qualitative results in Section 3.2.

E Limitations

Despite promising results, our approach can still generates problematic MWPs. Because some of our baselines simply copy a sample from the training data as the “generated” sample during evaluation, which would make a unfair comparison, here we instead conduct a small case study for our approach on the most challenging generation scenario where we randomly sample 25 contexts and combine it with each of the four equations that involve two variables.[5] This procedure produces 100 generated examples. We then qualitatively evaluate their generation quality. In total, we find that 17 out of the 100 generated samples are completely satisfactory and another 17 can become satisfactory with minor changes. Some common errors in our generated samples include: 1) incomplete information; 2) wrong question asked; and 3) incoherent question statement. We illustrate these types of errors in Table C. The errors suggest that, for better generation quality, we should further improve the model’s understanding of the semantics of the math operations and the relationship between various mathematical entities in the equation and the important words in the MWPs.

[5] In general, evaluating generated MWPs is a challenging task and we defer the investigation of human evaluation criteria and more comprehensive human evaluations to a future work.