MATHion: Solving Math Word Problems with Logically Consistent Problems

Anonymous ACL submission

Abstract

Solving the math word problems (MWPs) is a challenging task. Some existing MWP solvers retrieve textually similar problems and draw on their solution to solve the given problem. However textually similar questions are not guaranteed to have similar solutions. And questions could share the same solution but with different descriptions. Therefore in this work, we investigate the logical consistency among different problems and propose a novel framework named MATHion which solves **math** word problems with the log**i**cally c**on**sistent problems. Experimental results show that our method outperforms many strong baselines, including some pre-trained language model based methods. Further analysis shows that our retrieval method can learn the logical similarity between questions and plays a key role in our model's performance.

1 Introduction

Solving a Math Word Problem (MWP) is to take a mathematical descriptive **problem** as input and then generate an **expression** that can be evaluated to obtain the final answer (Zhang et al., 2020a). It needs the machine to extract relevant information from natural language text and perform mathematical reasoning to solve it (Patel et al., 2021).

Modern deep learning based MWP solvers usually adopt an encoder-decoder framework (Huang et al., 2018; Shen and Jin, 2020; Liu et al., 2019). Some methods propose sophisticated models for encoding quantities mentioned in the problem to produce a good expression (Shen and Jin, 2020; Tsai et al., 2021; Zhang et al., 2020c; Li et al., 2020; Qin et al., 2021). Some other methods retrieve similar problems from a problem bank and leverage their solutions to help solving the given problem. For instance, Robaidek et al. (2018) and Wang et al. (2019) have employed Jaccard and Cosine similarity metrics, Wang et al. (2017) have utilized word frequency based TF-IDF model, Huang et al. (2021) encode problems with word2vec embeddings and then use Maximum Inner Product (MIP) to retrieve similar problems.

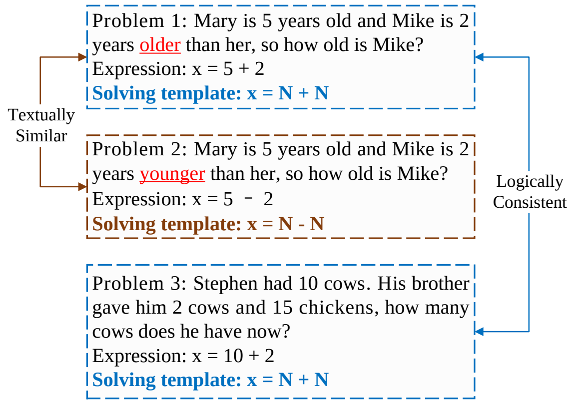

Figure 1: Example of the textually similar problems and logically consistent problems.

The previous retrievers usually tended to rely on textual similarity to find the most similar problem. However, textually similar problems are not guaranteed to have the similar solutions. Hence the similarity between problems should focus more on the logical perspective. For the rest of this paper, we say that two math word problems are *logical consistent* if they have the same solving template. Taking Fig. 1 for an example, although the problem 1 and the problem 2 only have one different word ("older" has been changed to "younger"), their solution is completely different. Hence they are textually similar but not logically consistent.

Since textual similarity based retrievers could be easily confused by this kind of problems, they tend to introduce noisy information. Therefore, the need of a retriever that can see through the narrative description and perceive the intrinsic logic of the problem arises urgently for solving MWPs.

In this paper, we propose a contrastive learning based "retrieve-then-generate" approach named MATHion, which solves **math** word problems with the logically consistent problems. Compared with the previous methods, we employ contrastive learning to train the retriever to mainly focus on the logical consistency instead of textual similarity when extracting similar problems. Besides, we propose a gated initialization and an aligned guidance mechanism thus the model could regard the solving template of the problem retrieved as hints to generate the final calculation formula.

The main contributions of this work are as follows: 1) We introduce a new method to retrieve logically consistent problems. To the best of our knowledge, this is the first attempt to consider the logical consistency among MWPs. 2) We propose two novel components, a gated initialization method and an aligned guidance mechanism, to integrate the retrieved solving template with our backbone generative model. 3) Experimental results show that our method outperform all baselines, including some pre-trained language model based methods.[1].

2 Related Works

The methods for solving Math Word Problems (MWPs) can be dated back to the 1960s (Charniak, 1969), and range from the rule-based methods (Mukherjee and Garain, 2008), semantic parsing based methods (Kwiatkowski et al., 2013; Shi et al., 2015; Huang et al., 2017) to deep learning based methods (Wang et al., 2018, 2019; Qin et al., 2020; Zhang et al., 2020b; Hong et al., 2021). Current deep learning based approaches usually regard solving MWPs as a sequence to sequence task and adopt an encoder-decoder framework to generate the solution. Extensive efforts are devoted to obtaining more accurate and substantial embedding of the input text (Wu et al., 2020; Tsai et al., 2021; Zhang et al., 2020c; Shen and Jin, 2020; Li et al., 2020; Qin et al., 2021). But the retrieval-based methods are relatively rare (Wang et al., 2019, 2017; Huang et al., 2021), it is probably because the current textual similarity based retrievers do not performs very well, which motivates us to develop a novel retriever to capture the logical consistency among problems.

Contrastive Learning (CL) has recently shown its high efficiency for learning representations in a self-supervised manner for both computer vision (Chen et al., 2020; He et al., 2020) and natural language processing domains (Jaiswal et al., 2020; Tian et al., 2020). Many previous works also used CL for training a neural network model to retrieve relevant examples, such as passages (Zhang et al., 2018; Karpukhin et al., 2020), images (El-Nouby et al., 2021; Deepak and Ameer, 2020) and even videos (Liu et al., 2021).

[1]The code and data are available at "anonymous"

3 Preliminary

In this section, we give the problem definition and briefly review our backbone generative model GTS (Xie and Sun, 2019).

3.1 Problem Definition

Formally, the MWP is formulated as follows: given a sequence of words $X = (x_1, x_2, \ldots, x_m)$ as the input problem, our goal is to generate expression $Y = (y_1, y_2, \ldots, y_n)$, which is the pre-order traversal sequence of a binary math expression tree (Liu et al., 2019). All numbers in the original problems X are replaced with the special token num and all non-constants numbers in Y are replaced with the special token num_i where i is the order of the number's appearance. Thus for each problem X, the set of numbers appearing in X is defined as $V = (num_1, num_2, \ldots, num_k)$. When generating expression Y, the entire output vocabulary contains V, the constants and the operators.

3.2 GTS Model

GTS (Xie and Sun, 2019) is a auto-regressive sequence-to-tree model to generate math expression tree in following steps.

Encoding An embedding layer and a bidirectional GRU (Cho et al., 2014) are employed to encode all tokens $(x_1, x_2, \ldots, x_m)$ in problem X as $(h_1, h_2, \ldots, h_m)$. The final hidden states in forward and backward directions are concatenated to obtain h^X for representing problem X.

Initialization To start the tree decoding process, the root embedding and the target vocabulary should be initialized. The root node embedding q_0 is initialized as h^X. The target vocabulary V_{tar} contains three parts, i.e., the operators V_o, the constants V_{con}, and the numbers appearing in problem V. The embedding in V_{tar} are initialized by the following equation:

$$e(y|X) = \begin{cases} E_{op}(y) & \text{if } y \in V_o \\ E_{con}(y) & \text{if } y \in V_{con} \\ h_{loc(y,X)} & \text{if } y \in V \end{cases} \quad (1)$$

where E_{op} and E_{con} are two embedding matrices, $loc(y, X)$ is the index position of y in X.

Tree decoding The whole tree decoding process involves following modules: **(1) Context module**: given a goal vector q and the encoder outputs, it derives a context vector c. **(2) Predict module**: given goal vector q and context vector c, it calculates the decoding score $s(y|X)$ of all the tokens in V_{tar}. The predicted token $\hat{y}$ is assigned to the token that gets the highest score. **(3) Merge module**: a recursive neural network to encode the left sub-tree as an embedding t_l. **(4) Left and right module**: Given current goal vector q and predicted token $\hat{y}$. If $\hat{y}$ is an operator, left module is used to generate the left sub-goal $q_l = LM(q, e(\hat{y}|X))$. Otherwise, the right module takes into account the embedding t_l of left sibling and generate the right sub-goal $q_r = RM(q, t_l, e(\hat{y}|X))$, where LM and RM are trainable networks defined in the original paper.

As such the entire tree decoding procedure can be summarized as follows: the numbers are leaf nodes, and the operators are non-leaf nodes and must have two children. After initialization, the left child node is repeatedly generated according to the matching score $s(y|X)$ until it is a leaf node. Afterward, merge module encodes the left subtree in a bottom-up manner and right module generates the right child nodes until the whole tree is filled up (See Algorithm 1 in Appendix A for more details).

4 Methodology

In order to imitate logically consistent problems, MATHion extends the encoder-decoder framework with a logical retriever. It contains two stages:

1) **Retrieval Stage**: As illustrated in Fig. 2, the problems with same solving template are regarded as positive problem pairs, then the contrastive loss is employed to train our retriever. Afterward, the trained retriever serves to extract problems from the problem bank to help the subsequent generation.

2) **Generation Stage**: GTS (Xie and Sun, 2019) is deployed as our backbone generative model in this stage. Given a retrieved problem (could be wrong), we propose a gated initialization to partly filter out the wrong information and use its solving template as guidance in each generation step.

4.1 Retrieval Stage

Encoder An embedding layer and a two-layer bidirectional GRU (Cho et al., 2014) are employed as the encoder to map each MWP to a dense vector.

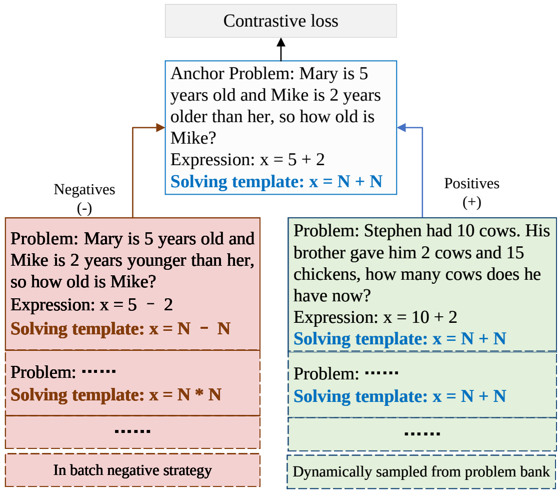

Figure 2: Our contrastive learning based retriever. The problems with the same solving template are positives. During training, each batch is dynamically composed to allow our model to see more diverse examples.

Afterwards the average of all word tokens' hidden states is calculated to represent the entire problem and denoted as H.

Positive and Negative Problem Pairs Construction To employ contrastive learning, we first need to tell the model which problems are logically consistent and which are not. Hence, given two different problems (X_i, X_j) and their corresponding expressions (Y_i, Y_j), we replace all numbers (except constants) in expression with token N to obtaining their **solving templates** $Y_i^{'}$ and $Y_j^{'}$. For example $10 + 2$ changes to $N + N$ when transferred into solving template. If $Y_i^{'} = Y_j^{'}$, we regard (X_i, X_j) as a positive pair. Since there is possibly more than one problem that can form positive pair with X_i, we call this set P_i. If $Y_i^{'} \neq Y_j^{'}$, they are regarded as a negative problem pair and the negative set for X_i are defined as N_i. It is worth noting that although $Y_i^{'} = Y_j^{'}$ is not the necessary and sufficient condition to the logical consistency of problem pair (X_i, X_j) since one expression Y may have many equal variants, we will prove the feasibility of our method through experiments.

Dynamic Batch Composition Assume that we have randomly grabbed B problems (i.e., $\{X_i | i \in [1, B]\}$) from training data, in order to ensure each problem X_i has at least n positive examples in this mini-batch and allow our model to see more diverse examples, we propose a dynamic batch composition technique. Specifically, for problem X_i, we concatenate another n positive problems that are randomly sampled from the problem bank. As such we dynamically formed a batch with batch size $(n+1)B$, and each problem could see multiple positive and negative problems in different epoch during training.

Contrastive Loss Since CL aims at reducing the distance between positive samples and pushing away the negative ones, it is coherent with our needs of clustering logically consistent problems and separating the inconsistent ones. Thereby we utilize InfoNCE loss function (van den Oord et al., 2018) and In-batch negatives strategy (Karpukhin et al., 2020) for contrastive learning. Given a batch of problems $(X_1, \cdots, X_{(n+1)B})$, they are encoded as $(H_1, \cdots, H_{(n+1)B})$. For problem X_i, its positive set P_i and negative set N_i are also given, then the InfoNCE loss L_i for X_i in this batch is defined as:

$$L_i = -\log \frac{d_i^p}{d_i^p + d_i^n}, \tag{2}$$

$$d_i^p = \sum_{X_j \in P_i} \exp\left(\frac{H_j^T H_i/\tau}{||H_j|| \cdot ||H_i||}\right), \tag{3}$$

$$d_i^n = \sum_{X_j \in N_i} \exp\left(\frac{H_j^T H_i/\tau}{||H_j|| \cdot ||H_i||}\right), \tag{4}$$

where τ is a hyper-parameter called temperature, thus the final loss for this mini-batch is:

$$L = \mathbb{E}_{\{X_i | i \in [1,(n+1)B]\}}[L_i] \tag{5}$$

Memory queue During training, we maintain and update a queue to store the representations of all problems in the problem bank. During inference, we retrieve the problem with the highest cosine similarity score in the problem bank.

4.2 Generation Stage

4.2.1 Solving Template Encoder

Given a problem retrieved and its corresponding solving template $Y^{'}$, we first use an embedding layer to map tokens in $Y^{'}$ into dense vectors. Afterward, a one-layer bidirectional GRU is exploited to encode sequential information. Finally, we calculate the sum of the final hidden states in forward $\overrightarrow{h_n^y}$ and backward $\overleftarrow{h_0^y}$ for representing the entire solving template:

$$h_i^y = [\overrightarrow{h_i^y}, \overleftarrow{h_i^y}], \quad h_{Y'} = \overrightarrow{h_n^y} + \overleftarrow{h_0^y} \tag{6}$$

where h_i^y and $h_{Y'}$ represent the hidden states of tokens and of $Y^{'}$, respectively.

4.2.2 Gated Initialization and Aligned Guidance Mechanism

Gated Initialization Since the first token decoded is very crucial in the auto-regressive model, we incorporate solving template embedding $h_{Y'}$ right from the initialization. The original GTS model only initialize the root node with the problem embedding h^X, here we propose a gated mechanism to fuse these two embeddings $h_{Y'}$ and h^X:

$$q_0 = \tanh(W_1 \cdot \hat{q}_0) \circ \text{sigmoid}(W_2 \cdot \hat{q}_0), \tag{7}$$

where $\hat{q}_0 = [h^X, h_{Y'}]$, W_1 and W_2 are trainable parameters, $[,]$ and $\circ$ represent the concatenation and the element-wise multiplication of vectors respectively. Since the retrieval model cannot achieve perfect accuracy, the retrieval problems are inevitably mixed with some wrong results. This gate mechanism is to adjust the portion of $h_{Y'}$'s contribution.

Aligned Guidance In an auto-regressive model, the token generated previously has a significant impact on the next, thereby we utilize the solving template of problem retrieved to guide the tree decoder in each generation step. As mentioned in 3.2, the original GTS model decodes each token according to the goal vector, which is generated using its left or right module, here as shown in Fig. 3, we add the embedding h_{i+1}^y to provide extra guiding information when generate the goal vector q_{i+1}:

$$q_{i+1} = \begin{cases} LM(q_i, e(\hat{y}_i|X)) + h_{i+1}^y & \hat{y}_i \in V_o \\ RM(q_i, e(\hat{y}_i|X), t_l) + h_{i+1}^y & \text{else} \end{cases} \tag{8}$$

where i denotes the i-th generation step, LM and RM represent the left and right module proposed in GTS. As such, the solving template could guide the entire tree decoding process step by step.

4.2.3 Training

Cross Entropy Loss For a problem expression pair (X, Y) in training data, our objective is to minimize cross entropy loss l, which is defined as the sum of negative log-likelihood of the probabilities for predicting each token y_i:

$$l(X, Y) = \sum -log(p(y_i|X)) \tag{9}$$

where $p(y_i|X)$ is the probability calculated with the decoding score $s(y|X)$:

$$p(y_i|X) = \frac{exp(s(y_i|X))}{\sum_{y_j \in V_{tar}} exp(s(y_j|X))} \tag{10}$$

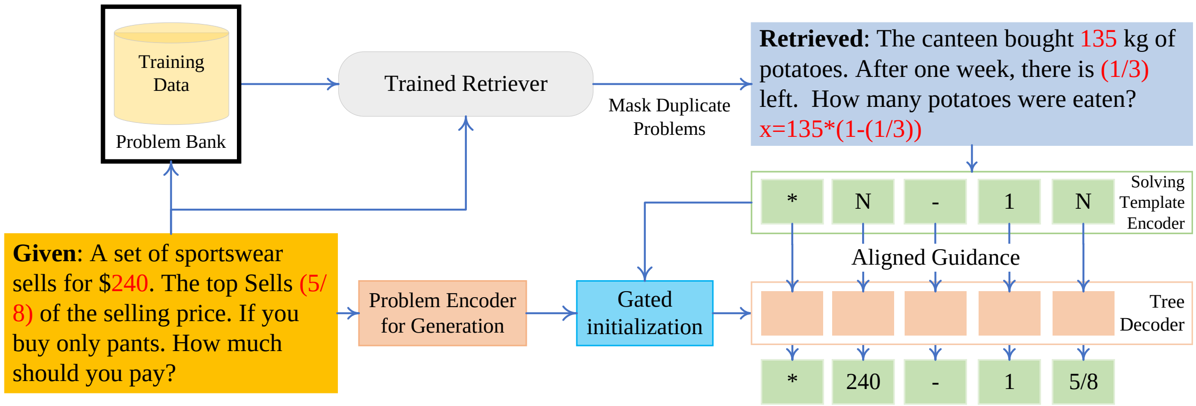

Figure 3: The inference process of our generative model. Given a problem, a trained retriever extracts a logically consistent problem. Then its solving template is leveraged into decoding procedure to generate the expression.

Artificial Noise Our retriever inevitably retrieves some wrong problems. To simulate this scenario and to improve the robustness of our generative model, we introduce a hyper-parameter teacher rate t to add extra artificial noises. During training, given a problem X_i, a random number n and teacher rate t, we provide a problem from P_i when n is smaller than t, otherwise we randomly sample one from N_i.

5 Experiment

5.1 Experimental Settings

Datasets Three datasets are used in this research. **Ape210K** (Zhao et al., 2020) consists of $210,488$ Chinese MWPs. Since different previous works (Wu et al., 2021; Liang et al., 2021; Zhao et al., 2020; Mikolov et al., 2013) applied different data filtering strategies and obtained their own version of the dataset, which makes comparison inequitable. Due to this reason, we utilize Ape210k just for problem bank construction. **Math23K** dataset (Wang et al., 2017) is labeled with $22,161$ elementary Chinese MWPs for training and $1,000$ MWPs for testing. Following most previous works, we experiment on the original split of data and conduct 5-fold cross-validation as well. We also want to evaluate MATHion on a widely-used English dataset MAWPS (Koncel-Kedziorski et al., 2016). But we have not found the appropriate English dataset for constructing problem bank. There exists large-scale English datasets but they follow very different annotation format with MAWPS (Amini et al., 2019; Cobbe et al., 2021), thus we propose MAWPS's Chinese version **CMWP**. To this end, we first use online translation tool[2] and then proofread it manually. Except for the language, all other settings remain the same. Likewise, we conduct 5-fold cross-validation on CMWP.

Configuration All words appearing less than 5 times are replaced by unknown token. All parameters are trained from scratch and initialized randomly. We utilize the training set and Ape210K to construct the entire problem bank. The duplicate problems are masked during retrieving to prevent leaking answer expression. The constant n for constructing a batch dynamically is set as 3 for Math23K and 2 for CMWP. The constant B is 64. The teacher rate t is set as 0.825 for both datasets. Following previous works, we use answer accuracy to evaluate our MWP solver, the expression is considered correct if it induced the same number as the ground truth. To evaluate our retrieval model, we employ top-k accuracy which is defined as the number of times where top k problems retrieved contain at least one positive problem belonging to positive set P.

Baselines We conduct a comprehensive comparison with the following baselines: **DNS** (Wang et al., 2017): a sequence-to-sequence model that directly maps problems to expressions. **DNS-Re** (Wang et al., 2017): a variant of DNS that combines a word frequency based retriever. **T-RNN** (Wang et al., 2019): a recursive neural network that predicts the missing operators of the predicted expression template. **T-RNN-Re** (Wang et al., 2019): T-RNN combined with a word frequency based retriever. **GTS** (Xie and Sun, 2019): our backbone model. **G2T** (Zhang et al., 2020c): a GTS based

[2] www.deepl.com

Model	Math23K	Math23K*	CMWP*
DNS	-	58.1%	-
DNS-Re†	-	64.7%	-
T-RNN	-	66.9%	-
T-RNN-Re†	-	68.7%	-
GTS	*75.6%*	74.3%	*68.9%*
G2T	*77.4%*	75.5%	*70.7%*
KA-S2T	*79.3%*	76.3%	*72.2%*
TS-G2T	79.1%	77.2%	*71.0%*
NUMS2T	*79.6%*	78.1%	*74.2%*
REAL-0	-	79.9%	-
REAL†	82.3%	80.8%	-
MATHion-RR†	76.2%	75.1%	67.71%
MATHion†	**84.0%**	**81.9%**	**77.2%**

Table 1: The answer accuracy results of MATHion and other baselines on Math23K and CMWP dataset. * denotes the 5-fold cross validation. † denotes the methods that combine a retriever. The italic numbers are not reported in original paper and obtained by running the public released codes. The other results are collected from the original paper.

solver that designs a graph network to enrich quantity representations. **KA-S2T** (Wu et al., 2020): a knowledge-aware GTS in which the entities in the problem and their categories are modeled as an entity graph. **TS-G2T** (Liang and Zhang, 2021): a G2T based model with an extra teacher module that makes the MWP encoding vector match the correct solution and disaccord from the wrong ones. **NUMS2T** (Wu et al., 2021): a KAS2T based model that explicitly incorporate numerical value information via the proposed number encoder and the auxiliary tasks. **REAL** (Huang et al., 2021): the first model that learns to solve MWP using human-like analogical learning way and it contains a memory module (REAL retriever), BERT initialized representation module, BERT initialized analogy module and a reasoning module. **REAL-0** (Huang et al., 2021): REAL without memory module. **MATHion-RR**: our generation model trained with teacher rate set as 0.55 and tested using problems retrieved by REAL retriever.

5.2 Overall results

The evaluation results of our model and the baselines are summarized in Table 1. There are the following observations: 1) MATHion has achieved a substantial gain over all baselines on all datasets. This should owe to the good performance of our proposed retrieval model since the answer accuracy drops severely if we employ the problems retrieved by the REAL retriever. 2) By comparing the performance of DNS-Re over DNS, T-RNN over T-RNN-Re, REAL over REAL-0, MATHion over GTS, we find that the retrieved problems do effectively help and our method brings the greatest improvement. DNS-Re surpasses DNS by 6.6% since the DNS can only solve a part of the problems that can be directly solved by the TF-IDF retrieval model. REAL surpass REAL-0 by only 0.9% since the BERT-initialized REAL-0 model already solves most problems. MATHion outperforms its backbone model GTS by 7.6%, illustrating that the logically consistent problems are tremendously helpful. 3) Compared with other GTS based baselines (G2T, KAS2T, NUMS2T) that integrate different kinds of knowledge, leveraging the information from the logically consistent problems is more effective. 4) The accuracy on CMWP is positively correlated with the accuracy on MATH23k, and the gaps between methods are similar too, which manifests that the results on CMWP are reasonable and the proposed CMWP is of high quality.

5.3 Ablation Study

top-k	Math23K	Math23K*	CMWP*
DNS-Re Retriever			
1	34.2%	31.20%	49.15%
3	43.9%	42.09%	60.99%
5	49.9%	48.00%	65.38%
10	57.9%	55.78%	70.49%
REAL Retriever			
1	52.3%	50.81%	55.49%
3	63.0%	61.52%	66.70%
5	67.2%	66.41%	70.54%
10	72.6%	72.39%	74.80%
MATHion Retriever			
1	83.0%	81.59%	80.76%
3	88.0%	88.03%	82.79%
5	88.1%	89.50%	83.55%
10	91.4%	91.05%	84.65%

Table 2: The top-1, top-3, top-5 and top-10 accuracy of DNS-Re retriever, REAL retriever and MATHion retriever. * denotes the results of five fold cross validation. The results of DNS-Re retriever and REAL retriever are obtained by reproduction.

Retriever Performance Since our generation model is based on our retriever, we report the performance of our retriever as well. To our best acknowledgment, few studies improve their model by retrieving associated questions. Moreover, there was not an acknowledged evaluation indicator since the ground truth of whether two problems are related is not given. Following our positive problem

Given Problem	Retrieved Problem
The farm harvested **24.2** tons of apples this autumn, which is **0.2** tons more than the **80%** of pears harvested. How many tons of pears were harvested? **x= (24.2-0.2)/80%**	In the long history of the planet, **90979** species of birds have died out, **769** more than **10** times the number of birds today. How many species of birds exist? **x= (90979-769)/10**
The red balls are **4** times the black balls. Touch one random ball at a time. After several touches, what is the ratio of the red ball touched? **x=4/ (4+1)**	The sum of a number expanded to **10** times and this original number is **84.15**. What is the original number? **x=84.15/ (10+1)**
The building has **4** floors, each with **5** classrooms. **120** lamps are installed. How many lamps are installed in each classroom on average? **x=120/4/5**	The farm has **5** barns, each barn is with **20** cows. Feed all cows **1200** kg of feed a day. How much feed to each cow on average per day? **x=1200/ (20*5)**
Mum buys cabbage for $**1.80** a kilo. The price of potatoes is **1.5** times the price of cabbage. How much more expensive are the potatoes than the cabbage per kilo? **x=1.8*(1.5-1)**	A cow weighs **156** kg. An elephant weighs **36** times as much as this cow How many kilograms does this elephant weigh more than this cow? **x=156*36-156**

Figure 4: The cases chosen form Math23K and translated. The problems given and retrieved are not necessarily the same in terms of semantic context and solving expression, but are logically consistent.

pairs construction method, we can exploit top-k accuracy as evaluation metrics to evaluate the retrieval performance. In this research, we select two retrievers of the previous state-of-the-art works as baselines. 1) **DNS Retriever** (Wang et al., 2017): a traditional statistical word frequency based TF-IDF retriever. 2) **REAL Retriever** (Huang et al., 2021): a word2vec based retriever that calculates the average of all word tokens' embedding to represent the whole problem and employs MIP search algorithm. As shown in Table 2, MATHion retriever performs much better than the other two methods. DNS-Re retriever does not perform well because it only attends to the word co-occurrence relations between questions. REAL-retriever performs better as it integrates certain semantic information. However, both of them can not succeed in deducing whether two problems are logically consistent. Because they heavily rely on textual information of the problem, as such they are likely to be misled by the problems that are textually similar but logically different.

Case Study To illustrate that our retrieval model does partly see through the narrative description and perceive the intrinsic logic of the problems, we give the following cases shown in Fig. 4. We observe that although the given problems and retrieved problems do not always share similar texts, the intrinsic logic is consistent. Such as the first line in Fig. 4, the given problem discusses the harvest in the orchard, whereas the retrieved problem takes the species of birds as semantic context, but we can use the identical template to solve both problems because the essence of them is alike. In addition, we also find that the top-k accuracy is not a completely accurate evaluation indicator since our positive pairs construction method cannot cover all logically consistent problems, such as $a \times (b + c)$ and $a \times b + a \times c$ are equivalent mathematically but are treated as negative pairs. However, our retrieval model still breaks through this limitation. Such as the third and the fourth line in Fig. 4, the given problems and retrieved problems have different semantic contexts, different solution templates, but the same intrinsic logic.

Effect of the Proposed Gate and Aligned Guidance Mechanism We conduct ablation study on Math23K to investigate the effect of our proposed gated initialization and aligned guidance mechanism. The experimental results in the Table 3 show that both two mechanisms bring improvement. We also observe that although without those two mechanisms, our model still achieves very comparable performance, further illustrating the importance of logical consistency between math word problems.

Model	Math23k	Math23K*
MATHion	**84.0%**	**81.93%**
w/o gate	83.4%	81.31%
w/o guidance	83.3%	81.35%
w/o gate & guidance	83.0%	81.18%

Table 3: Ablation study results on gated initialization and aligned guidance mechanism. w/o gate means the root node embedding is directly initialized as $\hat{q}_0$

Influence of teacher rate As aforementioned, in order to simulate the distribution of retrieval results, we introduce a parameter teacher rate t to add artificial noises during training. To investigate its influence, we first randomly re-split the entire Math23K dataset in the ratio of 8:1:1 and train a retrieval model of which top-1 accuracy attains 79.6% on the test set and 81.2% on the development set. Then we train our generative model by setting t as different values, the results are presented in Fig. 5. We find that the teacher rate significantly affects the model's performance. Because we use it to control the ratio of problems from P_i and N_i while training our generative model. Either larger or smaller teacher rate increases the distribution gap between training data and testing data. The model suffers from the overmuch wrong information when t is 0. Conversely, the model fully relies on the retrieved results when t is 1. The appropriate value is 0.8 because it is the closest to the top-1 accuracy.

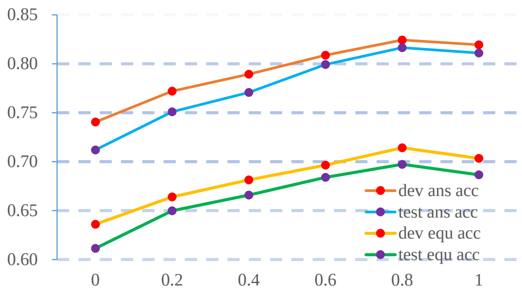

Figure 5: Answer and equation accuracy of the generative model trained with different teacher rate.

5.4 Model Analysis

Performance on problems with different number of quantities Figure 6 illustrates how our method and several baselines perform on problems with different number of quantities. For retrieval stage, when the quantity is less than 5, although there are many such problems in the problem bank, our retriever can still find the logically consistent problems, while the performance of REAL retriever is relatively poor. Conversely, when there are more quantities, the simple GRU has difficulty understanding the internal logic of the problem, but the REAL retriever achieves better results because such problems are rarely distributed in the problem bank and are textually similar. For generation stage, MATHion has achieved the best performance when the quantity is less than 4 but its performance declines when quantity gets more.

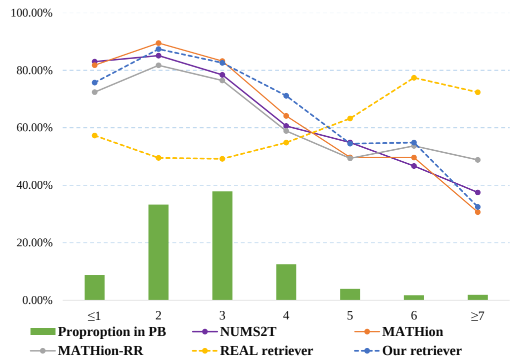

Figure 6: The answer accuracy and top-1 retrieval accuracy on problems with different number of quantities. PB is short for problem bank.

Performance on problems of different length Figure 7 illustrates how our retriever and REAL retriever perform on problems of different lengths. As the problems get longer, the encoding capability of GRU becomes relatively inadequate, thus the top 1 accuracy declines almost linearly and reduces to nearly 50% when the problem contains more than 90 characters. As for REAL retriever, once the problem length exceeds 20 the accuracy drops to below 50% severely.

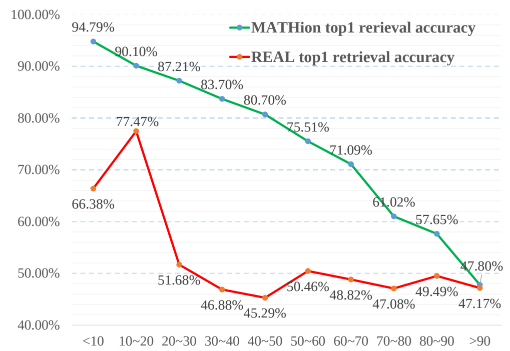

Figure 7: Performance of our retriever and REAL retriever on problems with different lengths.

6 Conclusion

In this work, we proposed MATHion which solves MWPS with logically consistent problems. Unlike previous textual similarity based retriever, we investigated the logical consistency between problems through contrastive learning and incorporate this information into the generative procedure. The experimental results have proved that it outperforms most previous methods. Except for MWPs, it can also be employed on other generative tasks such as abstract generation and machine translation.

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 *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2357–2367.

Eugene Charniak. 1969. Computer solution of calculus word problems. In *Proceedings of the 1st International Joint Conference on Artificial Intelligence*, pages 303–316.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey E. Hinton. 2020. A simple framework for contrastive learning of visual representations. In *Proceedings of the 37th International Conference on Machine Learning*, pages 1597–1607.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1724–1734.

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

S. Deepak and P. M. Ameer. 2020. Retrieval of brain MRI with tumor using contrastive loss based similarity on googlenet encodings. *Comput. Biol. Medicine*, 125:103993.

Alaaeldin El-Nouby, Natalia Neverova, Ivan Laptev, and Hervé Jégou. 2021. Training vision transformers for image retrieval. *CoRR*, abs/2102.05644.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross B. Girshick. 2020. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9726–9735.

Yining Hong, Qing Li, Daniel Ciao, Siyuan Huang, and Song-Chun Zhu. 2021. Learning by fixing: Solving math word problems with weak supervision. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence*, pages 4959–4967.

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.

Danqing Huang, Shuming Shi, Chin-Yew Lin, and Jian Yin. 2017. Learning fine-grained expressions to solve math word problems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 805–814.

Shifeng Huang, Jiawei Wang, Jiao Xu, Da Cao, and Ming Yang. 2021. Recall and learn: A memory-augmented solver for math word problems. *CoRR*, abs/2109.13112.

Ashish Jaiswal, Ashwin Ramesh Babu, Mohammad Zaki Zadeh, Debapriya Banerjee, and Fillia Makedon. 2020. A survey on contrastive self-supervised learning. *CoRR*, abs/2011.00362.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick S. H. Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 6769–6781.

Rik Koncel-Kedziorski, Subhro Roy, Aida Amini, Nate Kushman, and Hannaneh Hajishirzi. 2016. MAWPS: A math word problem repository. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics*, pages 1152–1157.

Tom Kwiatkowski, Eunsol Choi, Yoav Artzi, and Luke S. Zettlemoyer. 2013. Scaling semantic parsers with on-the-fly ontology matching. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1545–1556.

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 *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2841–2852.

Zhenwen Liang, Jipeng Zhang, Jie Shao, and Xiangliang Zhang. 2021. MWP-BERT: A strong baseline for math word problems. *CoRR*, abs/2107.13435.

Zhenwen Liang and Xiangliang Zhang. 2021. Solving math word problems with teacher supervision. In *Proceedings of the 30th International Joint Conference on Artificial Intelligence*, pages 3522–3528.

Qianying Liu, Wenyv Guan, Sujian Li, and Daisuke Kawahara. 2019. Tree-structured decoding for solving math word problems. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 2370–2379.

Song Liu, Haoqi Fan, Shengsheng Qian, Yiru Chen, Wenkui Ding, and Zhongyuan Wang. 2021. Hit: Hierarchical transformer with momentum contrast for video-text retrieval. *CoRR*, abs/2103.15049.

Tomás Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *Proceedings of 1st International Conference on Learning Representations*.

Anirban Mukherjee and Utpal Garain. 2008. A review of methods for automatic understanding of natural language mathematical problems. *Artificial Intelligence Review*, 29(2):93–122.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021. Are NLP models really able to solve simple math word problems? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2080–2094.

Jinghui Qin, Xiaodan Liang, Yining Hong, Jianheng Tang, and Liang Lin. 2021. Neural-symbolic solver for math word problems with auxiliary tasks. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*, pages 5870–5881.

Jinghui Qin, Lihui Lin, Xiaodan Liang, Rumin Zhang, and Liang Lin. 2020. Semantically-aligned universal tree-structured solver for math word problems. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 3780–3789.

Benjamin Robaidek, Rik Koncel-Kedziorski, and Hannaneh Hajishirzi. 2018. Data-driven methods for solving algebra word problems. *CoRR*, abs/1804.10718.

Yibin Shen and Cheqing Jin. 2020. Solving math word problems with multi-encoders and multi-decoders. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2924–2934.

Shuming Shi, Yuehui Wang, Chin-Yew Lin, Xiaojiang Liu, and Yong Rui. 2015. Automatically solving number word problems by semantic parsing and reasoning. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1132–1142.

Yonglong Tian, Dilip Krishnan, and Phillip Isola. 2020. Contrastive multiview coding. In *Proceedings of 16th European Conference on Computer Vision*, volume 12356, pages 776–794.

Shih-hung Tsai, Chao-Chun Liang, Hsin-Min Wang, and Keh-Yih Su. 2021. Sequence to general tree: Knowledge-guided geometry word problem solving. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*, pages 964–972.

Aäron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation learning with contrastive predictive coding. *CoRR*, abs/1807.03748.

Lei Wang, Yan Wang, Deng Cai, Dongxiang Zhang, and Xiaojiang Liu. 2018. Translating math word problem to expression tree. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1064–1069.

Lei Wang, Dongxiang Zhang, Lianli Gao, Jingkuan Song, Long Guo, and Heng Tao Shen. MathDQN: Solving arithmetic word problems via deep reinforcement learning. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, pages 5545–5552.

Lei Wang, Dongxiang Zhang, Jipeng Zhang, Xing Xu, Lianli Gao, Bing Tian Dai, and Heng Tao Shen. 2019. Template-based math word problem solvers with recursive neural networks. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence*, pages 7144–7151.

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, Copenhagen, Denmark. Association for Computational Linguistics.

Qinzhuo Wu, Qi Zhang, Jinlan Fu, and Xuanjing Huang. 2020. A knowledge-aware sequence-to-tree network for math word problem solving. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 7137–7146.

Qinzhuo Wu, Qi Zhang, Zhongyu Wei, and Xuanjing Huang. 2021. Math word problem solving with explicit numerical values. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*, pages 5859–5869.

Zhipeng Xie and Shichao Sun. 2019. A goal-driven tree-structured neural model for math word problems. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 5299–5305.

Dongxiang Zhang, Lei Wang, Luming Zhang, Bing Tian Dai, and Heng Tao Shen. 2020a. The gap of semantic parsing: A survey on automatic math word problem solvers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(9):2287–2305.

Jingyi Zhang, Masao Utiyama, Eiichiro Sumita, Graham Neubig, and Satoshi Nakamura. 2018. Guiding neural machine translation with retrieved translation pieces. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1325–1335.

Jipeng Zhang, Roy Ka-Wei Lee, Ee-Peng Lim, Wei Qin, Lei Wang, Jie Shao, and Qianru Sun. 2020b. Teacher-student networks with multiple decoders for solving math word problem. In *Proceedings of the 29th International Joint Conference on Artificial Intelligence*, pages 4011–4017.

Jipeng Zhang, Lei Wang, Roy Ka-Wei Lee, Yi Bin, Yan Wang, Jie Shao, and Ee-Peng Lim. 2020c. Graph-to-tree learning for solving math word problems. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3928–3937.

Wei Zhao, Mingyue Shang, Yang Liu, Liang Wang, and Jingming Liu. 2020. Ape210k: A large-scale and template-rich dataset of math word problems. *CoRR*, abs/2009.11506.

A The algorithm for describing the entire tree decoding process

Here we present the entire tree decoding process in Algorithm 1. All module mentioned such as Predict-module, LM are comletely defined in the paper of Xie and Sun (2019), see original paper for more concrete definition.

Algorithm 1: Tree decoding process

Input: q_0 and $\{h_i, i \in [1, m]\}$
Output: pre-order traversal expression

Step1: Calculate context vectors c

Step2: Generate q_l and predict $\hat{y}$
while $\hat{y}$ is an operator **do**
  $\hat{y} = \text{Predict-module}(q_l, c)$;
  $q_l = \text{LM}(q, e(\hat{y}|x))$;
  $c = \text{Context-module}(q, h_1, \cdots, h_m)$

Step3: Generate q_r, predict token $\hat{y}_r$ and combine the embedding of subtree
if $\hat{y}_r$ is an operator **then** go to Step2;
else go to step 4;

Step4: backtrack to find empty right node
if empty position exists **then** go to Step2;
else generation completed;

B Influence of the temperature while contrastive training

Influence of temperature The temperature τ in Eq. (4) is to control the difficulty of distinguishing positive examples from negative ones. It has a significant impact on the results of contrastive learning. To explore its influence, we randomly split the Math23K in the ratio of 8:1:1 and train our retrieval model with different value of τ and present the results in Fig. 8. The performance is

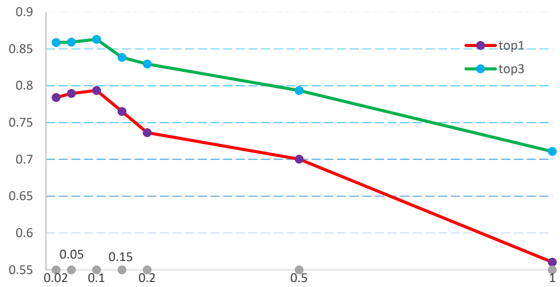

Figure 8: The top1 and top3 accuracy of different τ

the best when $\tau = 0.1$, and the performance of the retrieval model is not very sensitive to τ when τ is smaller than 0.1. Conversely, it decrease severely when τ getting larger because a large temperature may make this task to hard for simple GRU to learn. In this work, we set τ as 0.1 in our experiments.