REAL2: An End-to-end Memory-augmented Solver for Math Word Problems

Shifeng Huang [*†]
CVTE Research
huangshifeng@cvte.com

Jiawei Wang [*]
Hunan University
wangjiawei0531@gmail.com

Jiao Xu
CVTE Research
xujiao@cvte.com

Da Cao
Hunan University
caoda0721@gmail.com

Ming Yang
CVTE Research
yangming@cvte.com

Abstract

The task of math word problems (MWPs) has recently shown encouraging progress, e.g. in Recall and Learn (REAL), that solving problem by retrieving most similar questions based on a pre-trained memory module. In this article, we verify the effectiveness of different neural memory modules that can be trained end-to-end. Specifically, we first propose a Top-N pre-ranking process to retrieve candidate questions based on a Word2Vec model, and then we utilize a trainable memory module to re-rank the candidates to obtain the most similar Top-K questions. With this simple modification, we establish a stronger framework REAL2 that achieves state-of-the-art results, and opens a new designed space for memory-augmented solver in MWP task.

1 Introduction

During the post-epidemic era, with the support of policies and the development of internet technologies, online education behaviors and habits are popularized. Due to the learning difficulty of mathematics subject, the mathematical question search service plays a pivotal role in the K12 education stage. This service requires a search engine that can retrieve a solution of a specified question from a question bank, which is an extremely difficult project for there are tremendous mathematical questions that need to be collected. To this end, building AI systems that are capable of solving mathematical questions will effectively assist students in math learning, which is related to the research field of Math Word Problems (MWPs). A typical math word problem consists of problem description, expression and answer. The task of MWPs aims at automatically solving a math word problem given its problem description by a machine learning model, which requires the model to understand the problem description and using mathematics to infer the corresponding expression.

Despite its value and significance, learning mathematical knowledge from labeled examples and generalizing the knowledge to adapt to unseen examples is still a long-standing unsolved problem in the MWPs. One approach to this problem involves the inductive learning method [13, 11, 14, 12, 7, 15], which aims to summarize general rules through a single training example, and applies the general rules to unseen data. This approach leverages a deep neural model to learn mathematical knowledge in its parameters during training, which is difficult for nowadays neural architecture that lack human-like reasoning ability. An alternative approach, the analogical learning method [3], can

[*]Both authors contributed equally to this research.
[†]Corresponding author.

35th Conference on Neural Information Processing Systems (NeurIPS 2021) Workshop on Math AI for Education (MATHAI4ED)

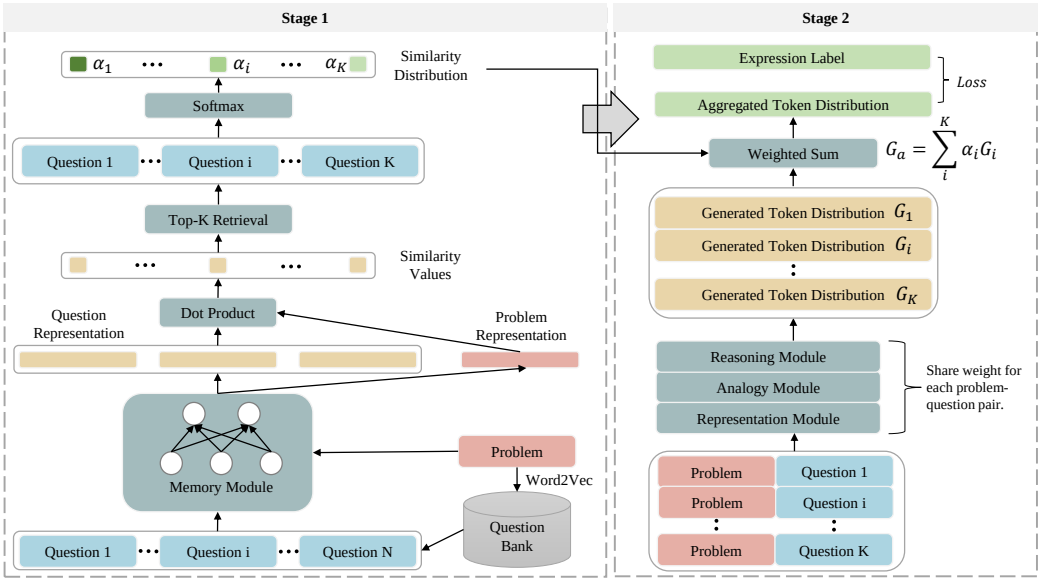

Figure 1: The illustration of our proposed REAL2 framework, which is composed of two stages. In the first stage, for an unsolved problem X, we first use a Word2Vec-based retriever to pre-ranking the most similar N candidate questions from the question bank. And the K similar questions are re-ranking based on the candidate questions and a trainable memory module. In the second stage, the token distributions of solution are generated based on the unsolved problem with its retrieved questions, and the aggregated token distribution is the final solution of the unsolved problem.

solve MWPs by referring to the most similar questions. Huang and Wang et al. [3] propose a novel memory-augmented model named REAL, which utilizes a pre-trained Word2Vec module to retrieve similar questions and use a UniLM-based [2] model to solve the unsolved problem by considering the relation of the problem and the retrieved questions. Extensive experiments had shown that REAL is an effective framework to solve MWPs using the analogical learning method.

Motivated by the previous work, this paper first explores the retriever (Memory Module) of the REAL framework, we deem the similarity of the retrieved questions may greatly affect the model for learning and reasoning solution while using analogical manner. Therefore, considering reducing the model noise introduced from the retriever, we aim to design a trainable memory module to replace the pre-trained Word2Vec retriever. Our proposed algorithm, named REAL2*, relies on a trainable neural retriever to model the similarity between unsolved problem and candidate question. This setting may promote the performance of the retriever when the whole framework is optimized by the end-to-end training scheme. In the following, we provide a detailed description of the proposed algorithm.

2 Problem Setting

The framework of REAL2 is presented in Figure 1. In general, our proposed framework is composed of two stages. In the first stage, we aim to retrieve Top-K similar questions given an unsolved problem. In particular, we first retrieve Top-N candidate questions from a question bank using Word2Vec model [9]. Then we utilize a trainable memory module to embed the representation of the unsolved problem and the candidate questions. Finally, we apply dot product and softmax operation to acquire the similarity distribution between each question and the problem, in which the similarity distribution can be used to retrieve the most similar K questions of the unsolved problem. In the second stage, we simply follow the implementation of REAL [3] that construct a neural framework by three key components: 1) Representation Module is a 6 layer Transformer-based encoder to represent the problem and the questions respectively; 2) Analogy Module is also a 6 layer Transformer-based encoder that can aggregate the information between the problem and the question; 3) Reasoning Module is a decoder used to generate token distribution based on the input sequence. With the three

*Our code has been released at `https://github.com/sfeng-m/REAL2`

components, K token distributions are generated. And the weighted sum of the similarity distribution and the generated token distributions are utilized to ensemble different predictions as the final output.

2.1 Problem Formulation

In this work, we consider the problem of solving a math word problem given its retrieved questions. Formally, we denote a problem that contains text description as $X \in \mathcal{X}$ which is formed by a sequence of word tokens $X = \{x_1, x_2, \cdots, x_L\}$, where $\mathcal{X}$ is named as question bank. Given a problem X, the N candidate questions $\mathcal{C}$ are retrieved by a Word2Vec-based ranking model $r(X, \mathcal{X})$, where $\mathcal{C} = \{C_1, C_2, \cdots, C_N\}$. And the K retrieved similar questions are retrieved by a DNN-based model $\mathcal{Z} = g(X, \mathcal{C})$, which are denoted as $\mathcal{Z} = \{Z_1, Z_2, \cdots, Z_K\}$. For each unsolved problem, the goal of the math word problem is to learn a model $Y = f(X, \mathcal{Z})$ that can predict each token $y_t \in \mathcal{V}$ of the expression $Y = \{y_1, y_2, \cdots, y_{|Y|}\}$, where V is a vocabulary of word.

3 Method

The framework of REAL2 consists of two important stages: 1) Similar Questions Retrieval; 2) Math Word Problems. In the first stage, REAL2 relies on a Word2Vec model and a Memory Module to represent questions and thus the questions can be retrieved effectively. In the second stage, given an unsolved problem and its retrieved similar questions, the relationships and similarities among the problem and the questions are considered by a UniLM-based model for estimating the token distribution of the expression.

3.1 Similar Questions Retrieval

Top-N Candidate Questions Retrieval. To reduce computational complexity, we first design a pre-ranking module $r(X, \mathcal{X})$ to rank N similar questions from a training set by a Word2Vec model, in which the module is pre-trained in the training set to learn word associations. And then we use the Word2Vec model to represent each word embedding of each question from the training set. In addition, the question embedding is acquired by applying average pooling on the word embeddings, and the similarity between two questions can be measured by the dot product of the corresponding two question embeddings. Therefore, given an unsolved problem, we calculate all of its similarities to other questions, and the Top-N candidate questions can be selected by filtering the highest similarities.

Top-K Similar Questions Retrieval. To retrieve the most similar questions of the unsolved problem by a trainable framework $\mathcal{Z} = g(X, \mathcal{C})$, we apply a memory module to learn the representations of the problem and the candidate questions, in which the memory module is instantiated as DNN-based model. As shown in Figure 1, to retrieve Top-K similar questions, we apply dot product on representations to calculate the similarities between the problem and the candidate questions, and the similarities are utilized to select K most similar questions. In addition, we apply the softmax operation to translate the similarities of K questions into similarity distribution, which will be used to aggregate the generated token distribution while solving the math word problem in the stage 2.

3.2 Math Word Problems

To solve a problem given its retrieved questions, we follow the settings of REAL [3] that leverages the text information of both problem and question to form the final target label by a UniLM-based model, which is implemented as three different modules. In particular, each problem-question pair is processed individually to form the generated token distribution. Therefore, given K retrieved questions, K generated distribution will be obtained. To further enhance the model's ability to learn to focus on the most similar question, we perform a weighted sum operation to connect stage 1 and stage 2, in which the memory module can be trained end-to-end with the whole framework. Specifically, given K generated token distributions $\mathcal{G} = \{G_1, G_i, \cdots, G_K\}$, where $G_i \in [0, 1]^{|V| \times L}$ and L is the maximum expression length. To fuse K different predictions into one aggregated token distributions G_a, the weighted sum operation is defined as follows:

$$G_a = \sum_{i}^{K} \alpha_i G_i,$$

where $[\alpha_1, \alpha_i, \cdots, \alpha_K]$ is the similarity distribution that measure the similarity between unsolved problem and each question.

Memory Module	Math23K (Accuracy)
Baseline	81.40
TextCNN	82.70
TextRCNN	82.94
Transformer	83.04
BERT	**83.18**

Table 1: Performance comparisons of various models of memory module on Math23K dataset. Note that the baseline result is published by the REAL paper [3] with the setting of $K = 3$.

Top-N \ Top-K	3	5
3	83.4	-
5	84.9	85.2
10	84.2	84.9

Table 2: Performance comparisons of a various number of candidate questions and retrieved questions. "-" means we do not perform the experiment because N should be greater than K in our proposed framework.

4 Experiments

Different Backbone of Memory Module To investigate the effectiveness of the trainable memory module, we implemented our framework follows the settings of REAL and only modified the framework of stage 1 part (Figure 1). In particular, we build different memory module that involves TextCNN [16], TextRCNN [5], Two-layer Transformer [10] and BERT [1] with the settings of $N = 10$, $K = 3$. And we report the experimental results on the Math23K dataset via 5-fold cross-validation follows the previous work [3]. More details can be found in Appendix A.1 and A.2. Table 1 shows the comparison results among different memory modules, we have the following observations: 1) By comparing the experimental results, we can see that the performance of all trainable memory modules is significantly better than the baseline's performance, which sets a new state-of-the-art baseline for the MWP task in learning by analogy. This indicates the quality of retrieved questions can be further improved through the end-to-end training procedure, which shows the effectiveness of our proposed framework; 2) BERT backbone achieves the best performance as compared to the others backbone, which is reasonable because BERT is a more effective model for representing text in most NLP tasks [1, 6, 8], and thus may re-rank more similar questions from the candidates.

Hyperparameter Search To better understand how the hyperparameters of Top-N and Top-K may affect the performance of the model, we perform experiments by setting different combinations of N and K based on the BERT backbone model. In order to reduce the error of the experimental results, we repeat the experiment 3 times on the public testing dataset [13], and take the average accuracy as the experimental result. As shown in Table 2, we have the following observations: 1) By comparing the performance of $N = 3, K = 3$ and $N = 5, K = 5$ experiments, we can see that the latter has higher accuracy than the former by 1.8 points. This is a significant improvement, indicating that the number of retrieved questions is a key factor in our proposed framework; 2) When the number of candidate questions is fixed, with the increase of the number of retrieved questions, the performance of the model increases consistently, which demonstrates that our model can comprehensively use more knowledge from different sources to solve the problem; 3) When the number of retrieved questions is fixed, retrieving more candidate questions do not improve the performance of the model consistently. A possible answer is, when the number of candidates reaches a certain number, providing more candidate questions will introduce more noise data to the memory module, and thus may retrieve some dissimilar questions to the main framework and reduce the performance of the model.

5 Conclusion and Future Work

We have presented an end-to-end memory-augmented solver for the MWP task, which is able to retrieve similar questions based on a trainable neural retriever. We showed the importance of the end-to-end training scheme of the entire framework, which achieves state-of-the-art performance than the previous work.

The proposed framework offers a novel stronger baseline for solving math word problems and opens a newly designed space in learning by analogy. For future work, we hope to speed up the retrieval process in a more neat way and establish a more effective model in the MWP task.

References

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics*, pages 4171–4186.

[2] Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. 2019. Unified language model pre-training for natural language understanding and generation. In *Advances in Neural Information Processing Systems*, pages 13063–13075.

[3] Shifeng Huang, Jiawei Wang, Jiao Xu, Da Cao, and Ming Yang. 2021. Recall and learn: A memory-augmented solver for math word problems. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

[4] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[5] Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Recurrent convolutional neural networks for text classification. In *Twenty-ninth AAAI conference on artificial intelligence*.

[6] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*.

[7] 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 *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 6162–6167.

[8] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

[9] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

[10] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

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

[12] 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 AAAI Conference on Artificial Intelligence*, 01, pages 7144–7151.

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

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

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

[16] Ye Zhang and Byron Wallace. 2015. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. *arXiv preprint arXiv:1510.03820*.

A Appendix

A.1 Math23K Dataset

The dataset Math23K contains 23,161 math word problems annotated with expressions and answers, which are split into 22,162 questions as the training set and 1,000 questions as the testing set. [†] [13]

A.2 Implementation Details

Our model is implemented using PyTorch[‡] based on a Ubuntu system equipped with GTX1080Ti GPU. The main framework that translates the problem-question pair into prediction is implemented by reusing the code of REAL [3]. REAL2 is trained for 50 epochs where the mini-batch size is set to 2. For optimizer, we use the same settings as REAL that using ADAM optimization algorithm [4] with the learning rate of 5e-4, $\beta_1 = 0.9$ and $\beta_2 = 0.99$. The learning rate is halved per 5 epochs when the number of the epoch is greater than 25. In addition, we set beam size to 1 in the beam search algorithm during decoding. And we use the answer accuracy as the evaluation metric which follows previous works [11, 3].

A.3 Case Study

Unsolved Problem	Binhe Park originally had 20 boats, earning 360 yuan a day. According to this calculation, now the park has added 15 boats. How much more yuan can be earned every day? *Expression:* (360 ÷ 20) × 15
Candidate Questions (CQ) Top-5	*CQ1:* People's Park used to have 30 boats, earning 540 yuan a day. According to this calculation, now there are 45 boats. How much more yuan can you earn a day? *Expression:* ((540 ÷ 30) × 45) - 450
	CQ2: The Golden Dragon Shipping Company has 26 ships and earns 780 yuan per day. Assuming that there are 30 boats, how much is the total income per day? *Expression:* (780 ÷ 26) × 30
	CQ3: There were 26 boats in a park, earning 910 per day. According to this calculation, now the park has added 6 more boats. How much yuan can be earned per day? *Expression:* (910 ÷ 26) × (26 + 6)
	CQ4: There used to be 3 cruise ships in Dipu Port, earning 225 yuan a day. If there are 14 cruise ships, how much yuan can be earned per day? *Expression:* (225 ÷ 3) × 14
	CQ5: There were originally 20 cruise ships in a park, earning 1,440 yuan a day. According to this calculation, adding 6 of the same cruise ships, how much more yuan can you earn every day? *Expression:* (1440 ÷ 20) × 6
Similar Questions (SQ) Top-3	*SQ1:* CQ2 *SQ2:* CQ4 *SQ3:* CQ3

Table 3: Typical case of the retrieving process. Note that the model is trained with the settings of $N = 5$ and $K = 3$.

Further, we conduct case analyses in Table 3. Our analyses are summarized as follows: 1) Given an unsolved problem, we can see that REAL2 can retrieve some similar candidates from the question bank. However, when we take a closer look at the expression of the unsolved problem and the candidate questions, the expressions of the top-ranked candidates does not mean that it is similar to the one of the unsolved problem; 2) By observing the experimental results of the similar questions, it can be found that the results of the candidate questions have been re-ranked, and the top-ranked results become more similar to the unsolved problem, which demonstrates the effectiveness of the trainable memory module.

[†] https://github.com/SumbeeLei/Math_EN/tree/master/data

[‡] http://www.pytorch.org