# Seeking Patterns, Not just Memorizing Procedures:
# Contrastive Learning for Solving Math Word Problems

**Zhongli Li**[1]*, **Wenxuan Zhang**[2]*†, **Chao Yan**[2]†, **Qingyu Zhou**[1],
**Chao Li**[1], **Hongzhi Liu**[2], **Yunbo Cao**[1]

[1]Tencent Cloud Xiaowei
[2]Peking University

{neutrali,qingyuzhou,diegoli,yunbocao}@tencent.com
{zwx980624@stu,cyan@stu,liuhz@ss}.pku.edu.cn

## Abstract

Math Word Problem (MWP) solving needs to discover the quantitative relationships over natural language narratives. Recent work shows that existing models memorize procedures from context and rely on shallow heuristics to solve MWPs. In this paper, we look at this issue and argue that the cause is a lack of overall understanding of MWP patterns. We first investigate how a neural network understands patterns only from semantics, and observe that, if the prototype equations like $n_1 + n_2$ are the same, most problems get closer representations and those representations apart from them or close to other prototypes tend to produce wrong solutions. Inspired by it, we propose a contrastive learning approach, where the neural network perceives the divergence of patterns. We collect contrastive examples by converting the prototype equation into a tree and seeking similar tree structures. The solving model is trained with an auxiliary objective on the collected examples, resulting in the representations of problems with similar prototypes being pulled closer. We conduct experiments[1] on the Chinese dataset Math23k and the English dataset MathQA. Our method greatly improves the performance in monolingual and multilingual settings.

## 1 Introduction

A Math Word Problem (MWP) is described as a natural language narrative with a math question. The MWP solver is required to generate a solution equation, which can be calculated to get the numerical answer, by understanding the contextual problem description.

In teaching, students are encouraged to recognize that mathematics is really about patterns and



**Prob. A:** Norma has 88 cards. She loses 70. How many cards will Norma have ? **Eq:** 88 - 70

**Prob. B:** Joyce starts with 75 apples. She gives 52 to Larry. How many apples does Joyce end with? **Eq:** 75 - 52

**Prob. C:** A bee has 6 legs. How many legs do 2 bees have? **Eq:** 2 * 6

**Prob. D:** 2 bee have 12 legs. How many legs does a bee have? **Eq:** 12 / 2

- $n_1 + n_2$
- $n_1 - n_2$
- $n_1 * n_2$
- $n_1 / n_2$
- $(n_1 + n_2) * n_3$
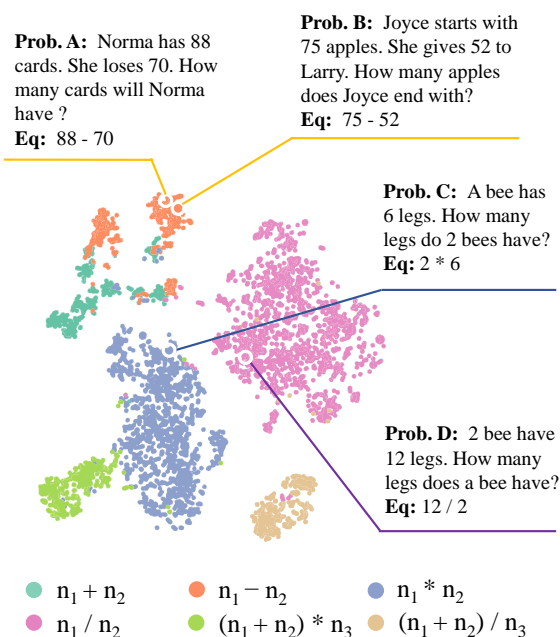- $(n_1 + n_2) / n_3$

Figure 1: The visualization of the problem representations by T-SNE. "Prob." and "Eq" are short for the math word problem and its solution equation. The problem A and B are in the same prototype equation "$n_1 - n_2$". The problem C and D are semantically similar.

not merely about numbers (Council, 1989). Mathematically excellent students explore patterns, not just memorize procedures (Schoenfeld, 1992). Recently, Patel et al. (2021) mention that existing MWP models (Xie and Sun, 2019; Zhang et al., 2020) rely on shallow heuristics to generate equations. These models can predict solutions well even if leaving only narratives without questions, which suggests that neural networks learn to solve MWPs by memorizing the lexical input like rote learning. Thus, existing models get stuck in memorize procedures. We look at this issue and hypothesize it is because they focus on text understanding or equation generation for one problem. The same quantitative relationship corresponds to many problems of different themes and scenarios, but previ-

---

ous methods overlook the outlining and distinction of MWP patterns.

In this work, we first investigate how a neural network understands MWP patterns only from semantics. We adopt the widely used encoder-decoder model structure (Cho et al., 2014). BERT (Devlin et al., 2019) is employed as the semantic encoder, and a tree decoder (Xie and Sun, 2019) is adopted to generate equations. We probe the problem representations in BERT. The visualization by T-SNE (van der Maaten and Hinton, 2008) in Figure 1 shows that, through the semantic encoder, most representations of problems with the same prototype equation are pulled closer, even if their narratives are semantically different. We also analyze the representations in different BERT layers, and the results show the lexical semantics mainly affects the problem-solving in lower layers. Besides, for each prototype equation, those problem representations far away from its center representation tend to produce incorrect solutions.

Inspired by it, we propose a contrastive learning approach that seeks similar prototypes to support model to better understand patterns and perceive the divergence of patterns. When collecting contrastive examples, we follow Xie and Sun (2019) to convert the prototype equation to a tree. Given an equation tree, the positive examples are retrieved if their trees or subtrees have the same structure, and the negative examples are collected from the rest in terms of the operator types and the size of the tree. The solving model is first jointly optimized by an equation generation loss and a contrastive learning loss on the collected examples, and then, is further trained on the original dataset. While the generation loss empowers the model to memorize procedures from the semantics, the contrastive learning loss brings similar patterns closer and disperses the different patterns apart.

We conduct experiments on the Chinese dataset Math23k (Wang et al., 2017) and the English dataset MathQA (Amini et al., 2019) in monolingual and multilingual settings. To support constructing multilingual contrastive examples, we follow Tan et al. (2021) to adapt MathQA as the counterpart of Math23k. Experimental results show that our method achieves consistent gains in monolingual and multilingual settings. In particular, our method allows the model to improve the performance in one language using data in another language, which suggests that MWP patterns are language-independent. Furthermore, we verify that, through our contrastive learning, the representations that previously generate wrong solutions get closer to their centers, and several problems are solved well.

To summarize, the contributions of this paper include: i) An analysis of the MWP model showing that the semantic encoder understands lexical semantics in lower layers and gathers the prototype equations in higher layers. ii) A contrastive learning approach helping the model to better understand MWP patterns and perceive the divergence of patterns. iii) Applications in the multilingual setting suggesting that we can further improve the model performance using data in different languages.

## 2 Related Work

### 2.1 Math Word Problem Solving

Given a natural language narrative with a mathematical question, the task is to generate a solution equation to answer the question. The methods can be divided into four categories: rule-based methods (Fletcher, 1985; Bakman, 2007), statistical machine learning methods (Kushman et al., 2014; Hosseini et al., 2014), semantic parsing methods (Shi et al., 2015; Koncel-Kedziorski et al., 2015) and deep learning methods (Wang et al., 2017; Huang et al., 2018a,b; Xie and Sun, 2019; Zhang et al., 2020).

Deep learning methods have achieved significant improvement on MWP solving. Wang et al. (2017) first attempt to use recurrent neural networks to build a seq2seq solving model. Xie and Sun (2019) propose a tree-structured decoder to generate an equation tree. Syntactically correct equations can be generated through traversing the equation tree. Zhang et al. (2020) apply graph convolutional networks to extract relationships of quantities in math problems. Recently, unsupervised pretraining of language models (Devlin et al., 2019; Yang et al., 2019a) has provided informative contextual representations for text understanding, and fine-tuning techniques (Cui et al., 2019; Li et al., 2021) have brought further performance gains. Several works (Kim et al., 2020; Tan et al., 2021; Cobbe et al., 2021) based on pretrained language models enhance the ability of problem understanding.
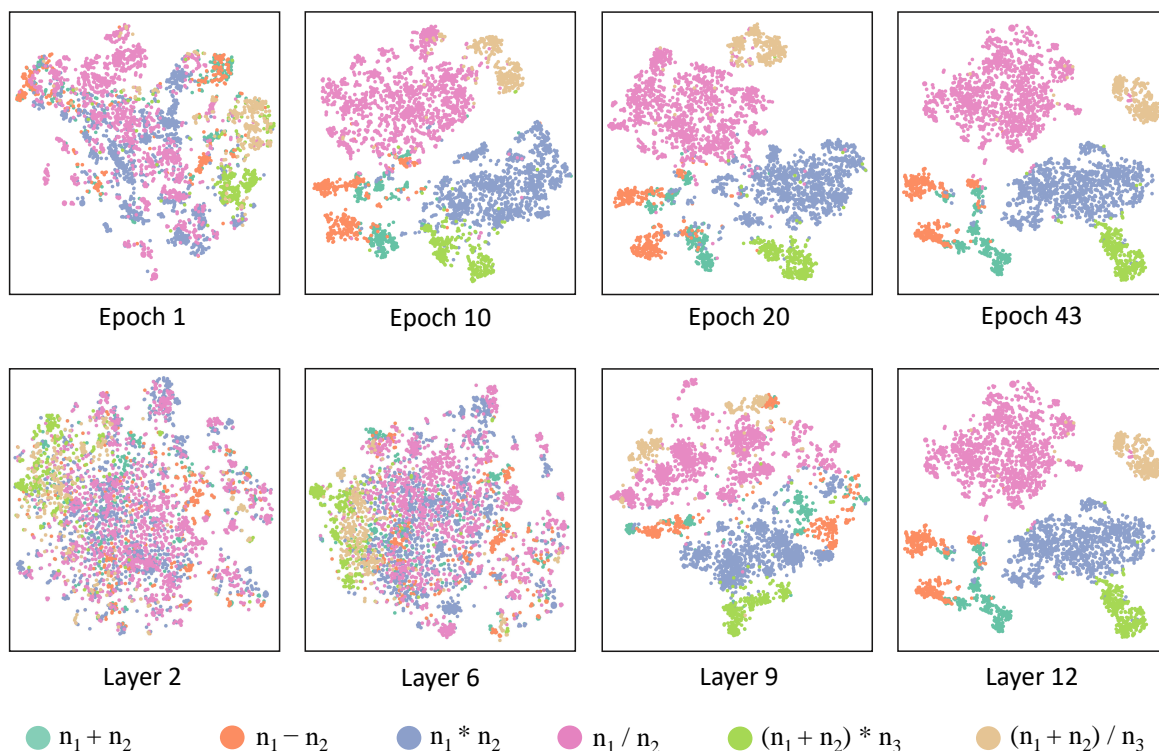
Figure 2: The T-SNE visualization of problem representations in different epochs and different layers. Different colors represent different prototype equations. The model achieves the highest accuracy at the training epoch 43.

## 2.2 Contrastive Learning

Contrastive learning is a method of representation learning, which is first designed by Hadsell et al. (2006). By pulling semantically similar embeddings together and pushing semantic different ones apart, contrastive learning can provide more effective representations. In NLP, similar approaches have been explored in many fields. Bose et al. (2018) develop a sampler to find harder negative examples, which forces the model to learn better word and graph embeddings. Yang et al. (2019b) use contrastive learning to reduce word omission errors in neural machine translation. Clark et al. (2020) train a discriminative model on contrastive examples to obtain more informative language representations. Gao et al. (2021) advance the performance of sentence embeddings by using contrastive learning in supervised and unsupervised settings. Yu et al. (2021) develop a contrastive self-training to help language model fine-tuning and label denoising in weak supervision.

To the best of our knowledge, this is the first work to adopt contrastive learning to MWP solving. With the supervision of contrastive learning, we seek similar MWP patterns to pull them closer, and collect confusing patterns to push them apart.

## 3 Semantic Encoder Gathers Prototypes

In this section, we explore how a neural network understands patterns from semantics. We adopt the encoder-decoder model structure to solve problems, and perform analyses on the problem representations. The observation is that the semantic encoder understands lexical semantics at lower layers and gathers the prototype equations at higher layers.

### 3.1 Experimental Setup

#### 3.1.1 Datasets

We perform analyses on two widely used datasets Math23k (Wang et al., 2017) and MathQA (Amini et al., 2019). The Math23k dataset is composed of 23k MWPs in elementary education, and the MathQA has 37k MWPs with multiple choices and equations.

#### 3.1.2 Model Architecture

**Semantic Encoder** The pre-trained language model BERT (Devlin et al., 2019) is employed as the semantic encoder. The unsupervised pre-training on large corpora renders the model to learn linguistic knowledge, which provides rich textual representations.
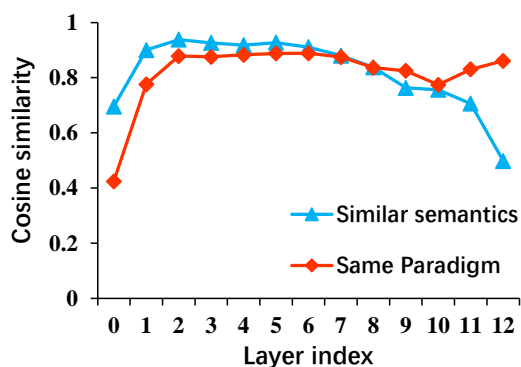
Figure 3: Similarities of problem representations in different BERT layers. The blue polyline corresponds to the semantically similar problems. The red polyline corresponds to problems with same prototype equation.



Figure 4: Model performance in each distance interval. The interval index $x$ indicates the cosine distances are in the interval $[0.1 \times (x-1), 0.1 \times x]$. The dotted line is computed by polynomial least squares fitting.

**Equation Decoder**   A tree decoder (Xie and Sun, 2019) is adopted to generate solution equations. We use the BERT-encoded representation of `[CLS]` token to initialize the root node when decoding. Recursively, the decoder generates the embedding of each node, and predicts the probabilities of number and operator candidates.

For brevity, we denote our model as **BERT-TD**. The model takes the textual problem description as the input and is optimized by minimizing the negative log-likelihoods of node probabilities for predicting the ground-truth equation tree.

### 3.2   Shifts of Problem Representation

To explore how the neural model learns MWP patterns during training, we first extract BERT-encoded representations of `[CLS]` token in different epochs and different layers. Then we perform the T-SNE visualization (van der Maaten and Hinton, 2008) shown in Figure 2. The representations of different epochs are picked from the top layer of BERT, and the representations of different layers are picked from the best trained model. It can be seen that, as the training goes on, the representations with the same prototype equation are gathering. Besides, with the increase of the depth of encoder layers, the gathering tendency becomes more and more obvious.

Intuitively, the prototype equation exhibits the essential relationship between the quantities in MWP. These results also verify that the patterns learned by the neural model are directly associated with the prototype equations.
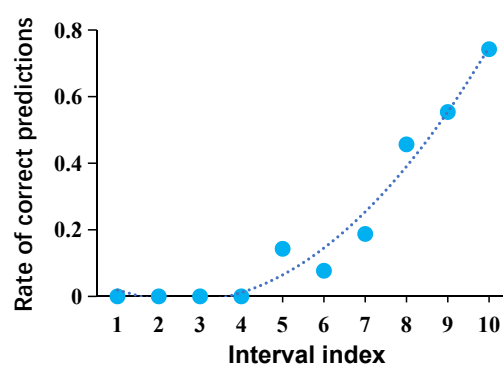
### 3.3   Semantics and Prototype Equation

From the visualizations, we can not see how semantics affects problem-solving. To this end, we collect 20 problem pairs with similar lexical semantics but exactly different prototypes, and 20 problem pairs with the same prototype but in different themes or scenarios. Not like taking the `[CLS]` representation in Section 3.2, we average the representations over all words in one problem. The cosine similarities of the averaged representations are calculated for these problem pairs in different BERT layers.

The averaged similarities are shown in Figure 3. The semantically similar problems obtain higher values in lower layers but the similarity gradually decreases as the model deepens. Meanwhile, with the increase of the model depth, although in different semantics, the problems with the same prototype equation achieve higher similarity. This demonstrates that lexical semantics affects problem-solving at lower layers, and the model further extracts prototypes from the semantics at higher layers.

### 3.4   Clustering and Solving Ability

With the above observation, we attempt to discover the relationship between prototype clustering and model performance. For each prototype equation, we first average the representations of the corresponding problems to obtain its center point, and then calculate the cosine distances between representations and its centers. A higher cosine distance means the representation is closer to its center. We split the cosine distance into several intervals and compute the proportion of correct predictions for each interval. The results are shown in Figure 4,

| Problem | Prototype Equation |
|---|---|
| Larry starts with $n_1$ cards. $n_2$ are eaten by a hippopotamus. How many cards does Larry end with? | $n_1 - n_2$ |
| Frank made $n_1$ dollars mowing lawns over the summer. If he spent $n_2$ dollars buying new mower blades, how many $n_3$ dollar games could he buy with the money he had left? | $(n_1 - n_2)/n_3$ |

Table 1: Math word problems with the same quantitative relationship, i.e. the subtraction of numerics $n_1$ and $n_2$. The same prototype equations are in red color.

which suggests that the representations apart from centers tend to produce wrong solutions.

## 4 Contrastive Learning

In this section, we propose a contrastive learning approach to help the model to perceive the divergence of MWP patterns. One drawback of existing deep learning methods is that they overlook the outlining and distinction of MWP patterns. In contrast, we seek similar prototype equations from various problems to support model to understand patterns, and collect easily confused patterns for model to distinguish.

### 4.1 Data Collection

We construct contrastive MWP triples $(p, p^+, p^-)$ containing a basic problem $p$ and its positive and negative examples $\{p^+, p^-\}$.

**Positive Example** One direct way is to collect problems whose prototype equation is completely the same as the given problem $p$. However, the same quantitative relationship in $p$ also exists in other problems. As shown in Table 1, for the second problem, before answering "How many games could he buy?", another hidden question is "How much money does he have?" whose solving equation is in the same prototype as the first problem. Thus, we parse the prototype equation to tree structure by following Xie and Sun (2019) and consider its sub-equations and subtrees. The problem $p^+$ is taken as a positive example if its tree or subtree has the same structure as $p$, such as "$tree$" and the subtree of "$tree^+$" in Figure 5.

**Negative Example** Bose et al. (2018) and Kalantidis et al. (2020) stress the importance of hard negative examples in contrastive learning. If we choose $p^-$ whose prototype is totally different from
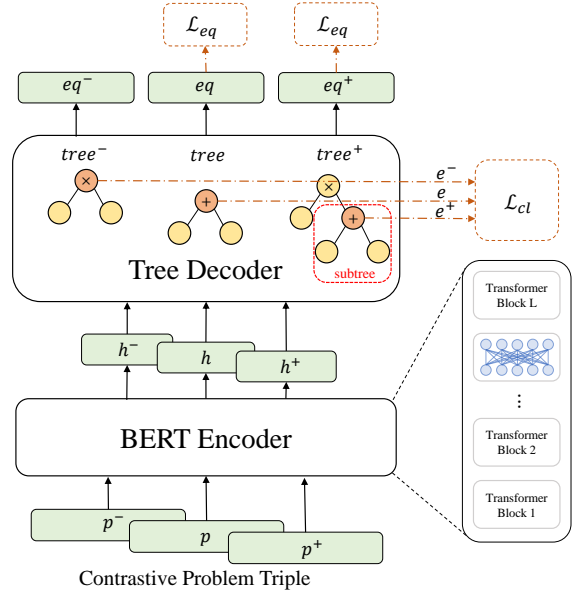


Figure 5: An overview of our model.

$p$, the original MWP model can easily distinguish them apart. Thus, in this work, the problem $p^-$ is chosen as a hard negative example if its tree has the same number of nodes but different operator node types, such as "$tree$" and "$tree^-$" in Figure 5. With the training on hard negative examples, our model can distinguish more subtle differences from various prototypes, and further grasp the inner pattern of MWP.

### 4.2 Training Procedure

We train the model on our contrastive problem triples. As shown in Figure 5, the problems are first encoded by BERT, and then the tree decoder predicts the nodes of the equation tree.

During contrastive learning, the triple $z = (p, p^+, p^-)$ are input to the model together to predict equation trees. Owing to the decoding manner of Xie and Sun (2019), each node embedding represents the whole subtree information rooted in it. The root node embeddings of the problem $p$ and its negative problem $p^-$ are picked for model to distinguish. For its positive problem $p^+$, we find the root node of the tree or subtree containing the same structure as $p$, and pull its embedding closer to that of $p$. For brevity, we denote these node embeddings as $(e, e^+, e^-)$ and the contrastive learning loss becomes:

$$\mathcal{L}_{cl} = \sum_z \max(0, \eta + sim(e, e^-) \\ - sim(e, e^+)), \quad (1)$$

| Dataset | #Train | #Dev | #Test |
|---------|--------|------|-------|
| Math23k | 21,162 | 1,000 | 1,000 |
| MathQA | 29,837 | 4,475 | 2,985 |
| MathQA$^{\dagger}$ | 23,703 | 3,540 | 2,410 |

Table 2: Statistics of the used datasets. The "MathQA$^{\dagger}$" is the adapted MathQA dataset by following Tan et al. (2021).

where $sim(\cdot)$ is the cosine similarity, and the $\eta$ is a margin hyper-parameter.

The basics of a MWP solving model is to generate a solution equation to answer the math question. We transform the target equation $y$ into Polish notation as $[y_1, y_2, ..., y_m]$, where $m$ is the equation length. The tree decoder generates $k$-node token $y_k$ recursively, and the loss of generating equation is computed as:

$$\mathcal{P}(y|p) = \prod_{k=1}^{m} \mathcal{P}(y_k|p) \tag{2}$$

$$\mathcal{L}_{eq} = \sum_{p} -\log \mathcal{P}(y|p) \tag{3}$$

The final training objective is to minimize the equation loss and contrastive loss as follows:

$$\mathcal{L} = \mathcal{L}_{eq} + \alpha \cdot \mathcal{L}_{cl} \tag{4}$$

where $\alpha$ is a hyper-parameter that represents the importance of the contrastive learning.

However, not all problems have positive examples, such as those problems whose solution is one value without any operator. With this in mind, we develop the two-stage training strategy. The MWP solver is first trained on our contrastive triples at stage I, and then further trained on the original dataset at stage II.

## 5 Experiments

We evaluate our method on two widely used datasets (Wang et al., 2017; Amini et al., 2019), and demonstrate its effectiveness in monolingual and multilingual settings.

### 5.1 Configuration

**Data and Metrics** We collect problems from the Chinese dataset Math23k (Wang et al., 2017) and the English dataset MathQA (Amini et al., 2019). As the formula formats of the two datasets are different, we follow Tan et al. (2021) to adapt MathQA

as a counterpart of Math23k. Table 2 shows data statistics. We report the accuracy of equation generation, namely as "Acc (eq)", that the problem is solved well if the generated equation is equal to the annotated formula. Considering several equations satisfy the problem solution, we report the accuracy of answer value, namely as "Acc (ans)", to see whether the value calculated by the generated equation is equal to the target value.

**Implementation** We conduct our contrastive learning in the monolingual and multilingual perspectives. In the monolingual setting, we construct contrastive triples inside each dataset. In the multilingual setting, for each problem, the positive and negative examples are from different sources. Specifically, given a Chinese MWP in Math23k, we collect positive examples from MathQA and negative examples from Math23k. We adopt BERT-base (Devlin et al., 2019) as the problem encoder, and follow Xie and Sun (2019) to build the tree-decoder for solution generation. The hidden size of the decoder is set to 768. Multilingual BERT is used in the multilingual setting. The max input length is set to 120 and the max output length is set to 45. The loss margin $\eta$ is set to 0.2. The weight $\alpha$ of contrastive learning loss is set to 5. We use AdamW (Loshchilov and Hutter, 2017) as our optimizer, and perform grid search over the sets of the learning rate as {5e-5, 1e-4} and the number of epochs as {30, 50} for each training stage. The batch size is fixed to 16 to reduce the search space, and we evaluate models for every epoch. We use the dropout of 0.5 to prevent over-fitting and perform a 3-beam search for better generations.

### 5.2 Baselines

To verify the effectiveness of the proposed method, we directly train our model on original datasets without contrastive learning. In particular, the multilingual baseline model is trained by mixing Math23k and the adapted MathQA. In addition to comparing with BERT, we also investigate the following approaches:

**GroupAttention**[2] (Li et al., 2019) develop an attention mechanism to capture the quantity-related and question-related information.

**GTS**[3] (Xie and Sun, 2019) generate equation

---

[2] https://github.com/lijierui/group-attention
[3] https://github.com/ShichaoSun/math_seq2tree

|  | Math23k | | MathQA[†] | |
| Models | Acc (eq) | Acc (ans) | Acc (eq) | Acc (ans) |
|---|---|---|---|---|
| *Monolingual Setting* | | | | |
| GroupAttention (Li et al., 2019) | - | 69.5 | 63.3* | 70.4* |
| GTS (Xie and Sun, 2019) | - | 75.6 | 68.9* | 71.3* |
| Graph2Tree (Zhang et al., 2020) | - | 77.4 | 70.0* | 72.0* |
| BERT-TD w/o CL | 71.2 | 82.4 | 73.5 | 75.1 |
| BERT-TD w CL | **71.8** | **83.2** | **74.4** | **76.3** |
| *Multilingual Setting* | | | | |
| mBERT-TD w/o CL | 67.8 | 80.5 | 72.0 | 73.5 |
| mBERT-TD w CL | **70.9** | **83.9** | **74.2** | **76.3** |

Table 3: Main results on Math23k and the adapted MathQA test sets. "Acc(eq)" is the equation accuracy and "Acc(ans)" is the answer accuracy. "∗" means our reimplementation based on released codes. "CL" is short for the contrastive learning. "mBERT" is short for the multilingual BERT.

|  | Pos. | Neg. | Math23k | MathQA[†] |
|---|---|---|---|---|
| Baseline | - | - | 80.5 | 73.5 |
| CL | Same | Ours | 82.3 | 75.5 |
|  | Ours | Rand | 82.3 | 75.8 |
|  | Ours | Ours | **83.9** | **76.3** |

Table 4: Results (answer accuracy) of different strategies collecting examples. "Pos." and "Neg." are corresponding to positive and negative examples. "Same" indicates the positive examples have exactly the same prototype equations. "Rand" indicates the negative examples are randomly selected from the rest.

| Margin $\eta$ | 0.05 | 0.1 | 0.15 | 0.2 | 0.3 |
|---|---|---|---|---|---|
| Math23k | 82.6 | 83.7 | 83.4 | **83.9** | 81.8 |
| MathQA[†] | 76.1 | 76.2 | 76.1 | **76.3** | 76.0 |

Table 5: Results (answer accuracy) of using different loss margin $\eta$ in the multilingual setting.

|  |  | Acc (eq) | Acc (ans) |
|---|---|---|---|
| Baseline | | 71.2 | 82.4 |
| CL ($\alpha = 1$) | Stage I | 70.1 | 81.5 |
|  | Stage II | 70.5 | 83.0 |
| CL ($\alpha = 5$) | Stage I | 70.6 | 82.5 |
|  | Stage II | **71.8** | **83.2** |

Table 6: Results of using different loss weight $\alpha$ on Math23k in the monolingual setting. Two-stage results are reported.

trees through a tree structure decoder in a goal-driven mannner.

**Graph2Tree**[4] (Zhang et al., 2020) design a graph-based encoder for representing the relationships and order information among the quantities.

### 5.3 Main Results

Experimental results are shown in Table 3. Training the MWP solver with our proposed contrastive learning outperforms the baseline models on all datasets.

**Monolingual Results** Compared to previous methods, the pretrained linguistic knowledge in BERT can help the MWP solver improve performance greatly. With our proposed contrastive learning method, our model achieves consistent gains on Math23k and the adapted MathQA. This suggests that seeking patterns with supervision benefits the model to solve MWPs.

**Multilingual Results** We adapt our model to the multilingual setting by using multilingual BERT and mixing two train sets. The contrastive learning improves Math23k answer accuracy to 83.9 (3.4 absolute improvements) and MathQA answer accuracy to 76.3 (2.8 absolute improvements), which are competitive with the monolingual results. This demonstrates that the model can learn similar patterns in different languages.

### 5.4 Analysis

We conduct ablations to better understand the contributions of different components in our contrastive learning method.
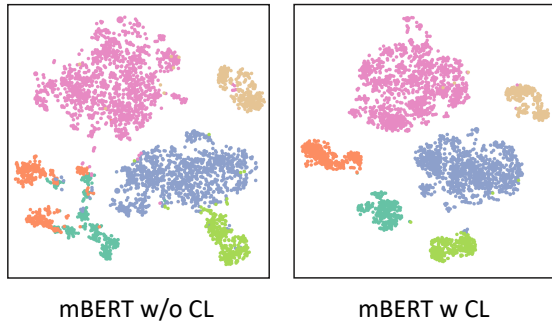
---

[4]https://github.com/2003pro/Graph2Tree

Figure 6: T-SNE visualization of the problem representation with and without our contrastive learning.



Figure 8: Equation accuracy in each distance interval with and without our contrastive learning.



Figure 7: Calinski-Harabasz index on the train/test set with and without our contrastive learning.

| **Input:** A boatman selling a boat along river flow. If he sell boat in steal water at 3 m/sec and flow of river is 2 m/sec, how much time he will take to sell 100 m. **Output (w/o CL):** 100 / (3 / 2) **Output (w CL):** 100 / (3 + 2) |
| :--- |
| **Input:** A pipe can fill the tank in 30 minutes and pipe b can empty the tank in 90 minutes. How long it will take to fill the tank if both pipes are operating together? **Output (w/o CL):** 1 / ((1 / 30) + (1 / 90)) **Output (w CL):** 1 / ((1 / 30) - (1 / 90)) |
| **Input:** If 20 liters of chemical x are added to 80 liters of a mixture that is 25% chemical x and 75% chemical y, then what percentage of the resulting mixture is chemical x? **Output (w/o CL):** 1 + ((25 / 100) * 5) **Output (w CL):** 20 + ((25 / 100) * 80) |

Table 7: Examples of the problem input and equation output of MWP solvers.

### 5.4.1 Effects of Data Collection

The contrastive examples consist of positive examples with similar patterns and negative examples with exactly different patterns. In this work, we investigate different strategies of collecting positive and negative examples. As well as our strategy, we attempt to collect MWPs containing the same prototype equation to be the positive examples, and randomly select negative examples from the rest.

Table 4 shows that our strategy achieves better performance on all datasets. In addition to the problems with the same prototype equations, our collected examples include more problems having the same equation subtree structures. It can be seen that the model can benefit from these examples. For the negative examples, we take the problems with the same number of operators but different operator types. If performing random selection, the model performance drops, which suggests that our collected examples can support the model to disperse the different patterns. No matter which strategy we use, compared to the baseline without contrastive learning, our method advances MWP solving and gives one way to improve the performance by using data in different languages.
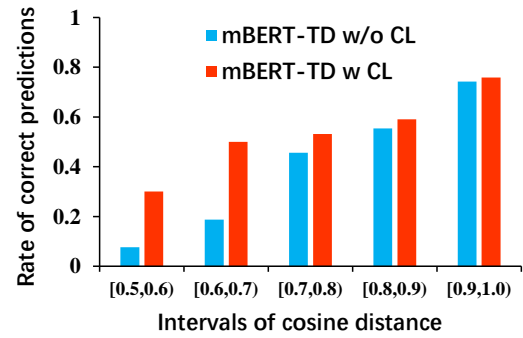
### 5.4.2 Effects of Hyperparameters

We train the "mBERT-TD" model with several loss margins (0.05, 0.1, 0.15, 0.2 and 0.3) to disperse the different patterns. As shown in Table 5, the margin 0.2 can help the model achieve the best performance but lower margins 0.1 and 0.15 also perform well.

As introduced in Section 4.2, we train our model in two stages and the loss weight $\alpha$ represents the importance of the contrastive learning. Table 6 shows the results of using different weights in each stage. It can be seen that the higher weight achieves better performance, and at stage II, training on all examples further improves the performance.

### 5.4.3 Visualization and Statistics

We perform the T-SNE visualization shown in Figure 6. The problem representations with the same prototype equation are more gathered through our contrastive learning. To measure this variation, we calculate the Calinski-Harabasz index (Caliński and Harabasz, 1974). Figure 7 shows that our

method supports the model to gain higher clustering scores.

The above results illustrate that, for each prototype equation, the representations are pulled closer to its centers. We re-compute the proportion of correct predictions as described in Section 3.4. The results are shown in Figure 8. We observe the accuracy increases in most intervals, which also verifies the effectiveness of contrastive learning. In particular, our model also performs well in lower intervals such as [0.6,0.7) and [0.7,0.8), which indicates those problems a little far away from their centers are not easily confused with other problems of different patterns, and our model disperses different patterns apart indeed.

Besides, we show few examples in Table 7. It can be seen that the contrastive learning method helps the model capture the quantitative relationships exactly.

# 6 Conclusion

In this paper, we find the neural network generates incorrect solutions due to the non-distinction of MWP patterns. To this end, we propose a contrastive learning approach to support the model to perceive divergence of patterns. We seek similar patterns in terms of the equation tree structure and collect easily confused patterns for our model to distinguish. Our method outperforms previous baselines on Math23k and MathQA in monolingual and multilingual settings.

# 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, Volume 1 (Long and Short Papers)*, pages 2357–2367, Minneapolis, Minnesota. Association for Computational Linguistics.

Yefim Bakman. 2007. Robust understanding of word problems with extraneous information. *arXiv preprint math/0701393*.

Avishek Joey Bose, Huan Ling, and Yanshuai Cao. 2018. Adversarial contrastive estimation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1021–1032, Melbourne, Australia. Association for Computational Linguistics.

T. Caliński and J Harabasz. 1974. A dendrite method for cluster analysis. *Communications in Statistics*, 3(1):1–27.

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, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1724–1734. ACL.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. ELECTRA: pre-training text encoders as discriminators rather than generators. *CoRR*, abs/2003.10555.

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.

National Research Council. 1989. *Everybody Counts: A Report to the Nation on the Future of Mathematics Education*. The National Academies Press, Washington, DC.

Baiyun Cui, Yingming Li, Ming Chen, and Zhongfei Zhang. 2019. Fine-tune BERT with sparse self-attention mechanism. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3548–3553, Hong Kong, China. Association for Computational Linguistics.

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 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.

Charles R Fletcher. 1985. Understanding and solving arithmetic word problems: A computer simulation. *Behavior Research Methods, Instruments, & Computers*, 17(5):565–571.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. Simcse: Simple contrastive learning of sentence embeddings. *arXiv preprint arXiv:2104.08821*.

Raia Hadsell, Sumit Chopra, and Yann LeCun. 2006. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 1735–1742. IEEE.

Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. 2014. Learning to solve arithmetic word problems with verb categorization. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 523–533.

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

Danqing Huang, Jin-Ge Yao, Chin-Yew Lin, Qingyu Zhou, and Jian Yin. 2018b. Using intermediate representations to solve math word problems. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 419–428, Melbourne, Australia. Association for Computational Linguistics.

Yannis Kalantidis, Mert Bulent Sariyildiz, Noe Pion, Philippe Weinzaepfel, and Diane Larlus. 2020. Hard negative mixing for contrastive learning. *arXiv preprint arXiv:2010.01028*.

Bugeun Kim, Kyung Seo Ki, Donggeon Lee, and Gahgene Gweon. 2020. Point to the expression: Solving algebraic word problems using the expression-pointer transformer model. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3768–3779.

Rik Koncel-Kedziorski, Hannaneh Hajishirzi, Ashish Sabharwal, Oren Etzioni, and Siena Dumas Ang. 2015. Parsing algebraic word problems into equations. *Transactions of the Association for Computational Linguistics*, 3:585–597.

Nate Kushman, Yoav Artzi, Luke Zettlemoyer, and Regina Barzilay. 2014. Learning to automatically solve algebra word problems. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 271–281.

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 57th Annual Meeting of the Association for Computational Linguistics*, pages 6162–6167, Florence, Italy. Association for Computational Linguistics.

Zhongli Li, Qingyu Zhou, Chao Li, Ke Xu, and Yunbo Cao. 2021. Improving BERT with syntax-aware local attention. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 645–653, Online. Association for Computational Linguistics.

Ilya Loshchilov and Frank Hutter. 2017. Fixing weight decay regularization in adam. *CoRR*, abs/1711.05101.

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, Online. Association for Computational Linguistics.

A. Schoenfeld. 1992. Learning to think mathematically: Problem solving, metacognition, and sense making in mathematics (reprint). *Journal of Education*, 196:1 – 38.

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.

Minghuan Tan, Lei Wang, Lingxiao Jiang, and Jing Jiang. 2021. Investigating math word problems using pretrained multilingual language models.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605.

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.

Zhipeng Xie and Shichao Sun. 2019. A goal-driven tree-structured neural model for math word problems. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 5299–5305. International Joint Conferences on Artificial Intelligence Organization.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. 2019a. XLNet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237*.

Zonghan Yang, Yong Cheng, Yang Liu, and Maosong Sun. 2019b. Reducing word omission errors in neural machine translation: A contrastive learning approach. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6191–6196.

Yue Yu, Simiao Zuo, Haoming Jiang, Wendi Ren, Tuo Zhao, and Chao Zhang. 2021. Fine-tuning pretrained language model with weak supervision: A contrastive-regularized self-training approach. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1063–1077, Online. Association for Computational Linguistics.

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 58th Annual Meeting of the Association for Computational Linguistics*, pages 3928–3937, Online. Association for Computational Linguistics.