Compositional Mathematical Encoding for Math Word Problems

Anonymous ACL submission

Abstract

Solving math word problem (MWP) remains a challenging task, as it requires to understand both the semantic meanings of the text and the mathematical logic among quantities, i.e., for both semantics modal and quantity modal learning. Current MWP encoders work in a uni-modal setting and map the given problem description to a latent representation, then for decoding. The generalizability of these MWP encoders is thus limited because some problems are semantics-demanding and others are quantity-demanding. To address this problem, we propose a *Compositional Math Word Problem Solver (C-MWP)* which works in a bi-modal setting encoding in an interactive way. Extensive experiments validate the effectiveness of *C-MWP* and show its superiority over state-of-the-art models on public benchmarks.

1 Introduction

The task of math word problem (MWP) solving aims to map natural language problem descriptions into executable solution equations to get the correct answer, which is a sub-area of neuro-symbolic reasoning. It requires perceptual abilities such as comprehending the question, identifying the quantities and corresponding attributes, as well as complex semantics understanding skills like performing logical inference, making comparisons and leveraging external mathematical knowledge.

While MWP encoders have been sophisticatedly designed to understand the natural language problem description, the difference on understanding diverse types of problems has not been aware of. Our investigation finds that MWP can generally be grouped into three categories, "Story Problem", "Algebra Problem" and "Knowledge Problem". "Story Problem" often includes significant amount of background information like characters, objectives and behaviors. "Algebra Problems" involves math notations or is composed of elementary concepts. "Knowledge Problem" asks for external knowledge like geometry and number sequence. Examples of different types of problems are given in the appendix.

These types of problems can be compositionally understood at the different level attention to the *semantics* modal and *quantity* modal. However, the encoders in existing MWP solvers either model only the semantics modality or utilize quantity modal priors to refine the MWP encoding (Zhang et al., 2020; Shen and Jin, 2020). This limitation, ***one joint modal cannot do it all***, decreases the generalization of MWP solvers and is what compositional learning aims to address. In this work, ***we propose to disentangle semantics modal and quantity modal by compositional learning at the encoding stage, aiming to improve the generalization across different types of problems***.

Contributions. Our main contributions are threefold. (i) A novel and effective bi-modal approach is proposed for the first time to enable MWP compositional understanding. (ii) A joint reasoning module with multi-step is designed for our bi-modal architectures to flexibly incorporate different modalities. (iii) Extensive experiments and ablative studies on two large-scale MWP benchmarks – Math23k (Wang et al., 2017) and MAWPS (Koncel-Kedziorski et al., 2016) show the superiority of the proposed approach over related works.

2 Our approach

The overview of our proposed model is shown in Figure 1.

2.1 Compositional Mathematical Encoder

The CMEncoder block consists of an semantic encoder, a quantity encoder and a dynamic fusion block. The semantic encoder aims to extract semantics information from the problem description, understanding the background and objectives. The

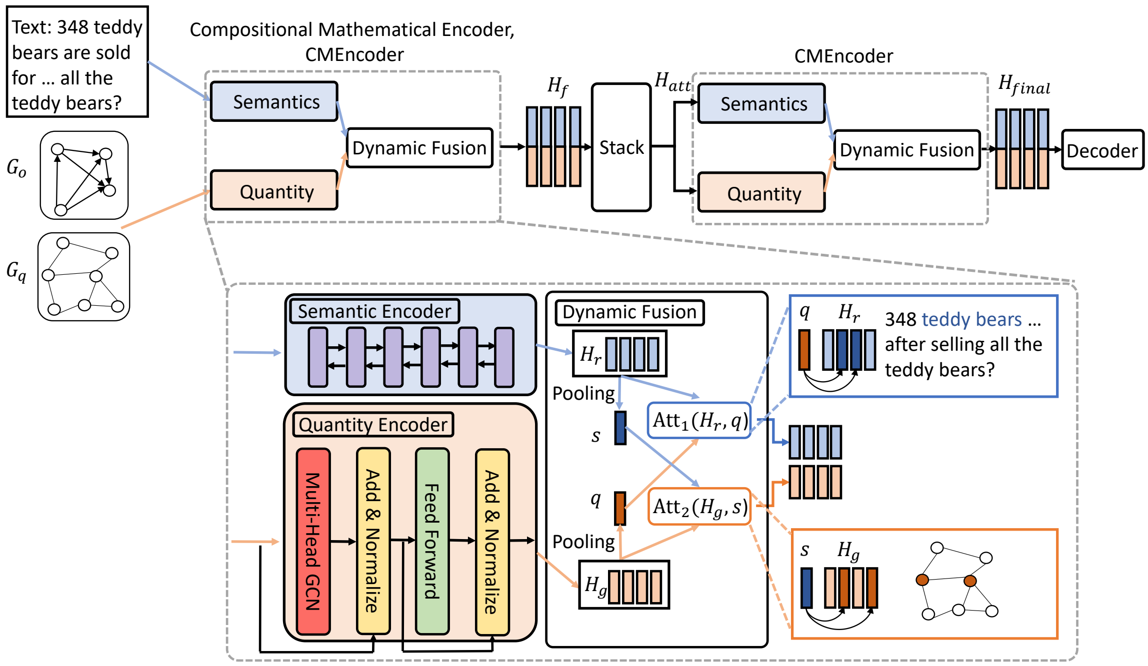

Figure 1: The overall architecture of the proposed network *C-MWP*, which is composed of two stacked CMEncoders and a decoder (the top part). The CMEncoder block (shown at the bottom) takes the given problem description, and runs in parallel to obtain H_r from the semantic encoder, and H_g from the quantity encoder. A dynamic fusion module incorporates H_r and H_g by cross-modal attention. The obtained H_f is attentively stacked with H_r and H_g. The resulted H_{att} is sent to the next CMEncoder block. The final problem representation H_{final} goes to the decoder for generating the final solution equation.

latter part encodes problems only with quantity-related graphs, helping the encoder to know the properties about quantities and relationship between quantities.

Sematic Encoder. To demonstrate the robustness of our approach, we implemented two different semantic encoders as our backbone.

Firstly, similar to the classic Seq2Seq model, we encode the problem description W by a bidirectional gated recurrent unit (BiGRU) (Cho et al., 2014). The outputs of GRU are hidden state vectors of all tokens, $H_r = \{h_1, h_2, ..., h_n\}$, where n is the length of problem W.

$$H_r = BiGRU(Embed_s(W)) \quad (1)$$

where $Embed_s(W)$ is the embedding result of textual description W in semantics modal. Empirically, we find that two stacked CMEncoders as shown in Figure 1 achieve the best performance.

Secondly, pre-trained language models (PLMs) have been ubiquitous in NLP tasks. We use the latest push of MWP-BERT (Liang et al., 2022) as our semantic encoder to obtain H_r. In this case, we use only one CMEncoder without stacking another one.

Quantity Encoder. To encode the quantity modal in the problem W, we feed a graph transformer G_{trans} with $Embed_q(W)$,

$$H_g = G_{trans}(Embed_q(W)) \quad (2)$$

where $Embed_q(W)$ is the embedding that representation the information in quantity modal from constructed quantity graphs and order graphs following (Zhang et al., 2020), aiming to capture the relationship among quantities and contexts. In the training process, the two encoders with $Embed_s(W)$ and $Embed_q(W)$ are updated to extract the semantics and quantity features, respectively. In this way, semantics and quantity modals are disentangled, which allivates the issue of "one joint modal cannot do it all". In other words, the compositionality of the CMEncoder enables the C-MWP solver to pay different levels of attention when solving different problems.

Dynamic Fusion. To achieve joint reasoning over the semantics information and quantity information, we design a dynamic fusion module to flexibly incorporate the features from these two modals. First, we get s and q from the mean pooling of H_r and H_g, respectively. Then, cross-modal attention is applied between H_r and q, H_g and s:

$$
\begin{aligned}
Att_1(H_r, q) &= \Sigma_{i=1}^{n} a_i H_{ri} \\
Att_2(H_g, s) &= \Sigma_{i=1}^{n} b_i H_{gi}
\end{aligned} \tag{3}
$$

where the attention scores a_i, b_i come from:

$$
\begin{aligned}
a_i &= W_a^1 \, tanh(W_a^2 \, (H_{ri} \,||\, q)) \\
b_i &= W_b^1 \, tanh(W_b^2 \, (H_{gi} \,||\, s))
\end{aligned} \tag{4}
$$

where W_a^1, W_a^2, W_b^1 and W_b^2 are parameter matrices. The cross-modal attention here grounds the quantity information in the semantics modal, and vice versa. By applying different weights on different modals, our model is flexible to pay more or less attention on a certain modal. Finally, the output of dynamic fusion is:

$$
H_f = Att_1(H_r, q) \,||\, Att_2(H_g, s). \tag{5}
$$

2.2 Stack Multiple CMEncoders

Human often need to make multiple glimpses to refine an MWP solution. Similarly, a CMEncoder can be stacked in multiple steps to refine the understanding of an MWP, as shown in Figure 1. Given the output from the semantic encoder, quantity encoder and dynamic fusion module at layer $k-1$, the features are stacked as:

$$
H_{att}^{(k-1)} = c_r H_r^{(k-1)} + c_g H_g^{(k-1)} \tag{6}
$$

where the attention weights c_r and c_g are:

$$
\begin{aligned}
c_r &= W_r^1 \, tanh(W_r^2 \, (H_r^{(k-1)} \,||\, H_f^{(k-1)})) \\
c_g &= W_g^1 \, tanh(W_g^2 \, (H_g^{(k-1)} \,||\, H_f^{(k-1)}))
\end{aligned} \tag{7}
$$

where W_r^1, W_r^2, W_g^1 and W_g^2 are parameter matrices. The following CMEncoder block takes $H_{attn}^{(k-1)}$ as input, and outputs $H_r^{(k)}$, $H_g^{(k)}$ and $H_f^{(k)}$, which can be sent for the update at layer $k+1$.

After finishing the K-th step reasoning, we concatenate the final $H_r^{(K)}$ and $H_g^{(K)}$ as the final output representation of the input problem:

$$
H_{final} = H_r^{(K)} \,||\, H_g^{(K)}. \tag{8}
$$

Then H_{final} is fed to the decoder to generate the solution tree.

2.3 Decoder

We follow the same implementation as proposed in (Xie and Sun, 2019). Eventually, the decoder will output the pre-order traversal sequence of the solution tree.

	Math23k	Math23k*	MAWPS	#E
RNN Based				
DNS	-	58.1	59.5	$3.0M$
GTS	75.6	74.3	82.6	$7.2M$
Graph2Tree	77.4	75.5	83.7	$9.0M$
NUMS2T	78.1	-	-	$7.9M$
Multi-E/D	78.4	76.9	-	$14.2M$
HMS	78.4	-	80.3	$9.5M$
EEH-G2T	78.5	–	84.8	$9.9M$
C-MWP (RNN)	**80.3**	**77.9**	**84.9**	$7.6M$
PLM Based				
REAL	82.3	80.0	–	$110M$
BERT-CL	83.2	–	–	$102M$
RPKHS	83.9	82.2	–	$102M$
MWP-BERT	84.7	82.4	–	$110M$
Gen&Rank	85.4	84.3	–	$610M$
MWPtoolkit	–	76.9	88.4	$110M$
C-MWP (PLM)	**86.1**	**84.5**	**89.1**	$130M$

Table 1: Math23k column shows the results when evaluating on the public test set of Math23k, while the Math23k* column shows the result of 5-fold cross validation on Math23k dataset. The last column #E denotes the number of parameters in encoders.

2.4 Training Method

Given the training samples with problem description W and the corresponding solution S, the main training objective is to minimize the negative log probability for predicting S from W, empowered by the compositionality of the CMEncoders. Therefore, the overall loss is:

$$
L = L_{MWP} + ||Embed_s||_2 + ||Embed_q||_2 \tag{9}
$$

where L_{MWP} is the negative log prediction probability $-\log p(S \mid W)$. The L_2 norm of the encoder embedding matrices is added to the loss function as regularization terms.

3 Experiments

3.1 Datasets

Math23k (Wang et al., 2017) containing 23,162 Chinese MWPs is collected from several educational websites. Some previous works choose the public training/test set in evaluation, while others use 5-fold cross validation. In this work, we report the evaluation results on both settings.
MAWPS (Koncel-Kedziorski et al., 2015) is an MWP dataset owning 2,373 English MWPs.

3.2 Baselines

GTS (Xie and Sun, 2019) proposes a powerful tree-based decoder. **Graph2Tree** (Zhang et al.,

	Graph Encoder	Compositional Structure	Dynamic Fusion	Acc(%)
GTS	✗	✗	✗	75.6
Graph2Tree	✓	✗	✗	77.4
	✓	✓	✗	78.1
	✓	✗	✓	78.9
C-MWP	✓	✓	✓	**80.3**

Table 2: Accuracy among different ablated models.

Model	Overall	Story	Algebra	KNWL
GTS	75.4	75.1	82.8	64.3
Graph2Tree	77.4	76.3	89.7	57.1
Multi-E/D	78.4	77.8	88.8	61.9
C-MWP (RNN)	**80.3**	**80.0**	**90.0**	**66.7**
MWP-BERT	84.7	85.6	88.8	72.0
C-MWP (PLM)	**86.1**	**87.5**	**90.7**	**72.0**

Table 3: The answer accuracy (%) of problems in different types. KNWL stands for the *external knowledge* required problems.

2020) constructs graphs to extract useful relationships in an MWP. **NumS2T** (Wu et al., 2021b) encode quantities with explicit numerical values. **Multi-E/D** (Shen and Jin, 2020) proposes to use multiple decoders in MWP solving. **HMS** (Lin et al., 2021) develops a hierarchical word-clause-problem encoder. **EEH-G2T** (Wu et al., 2021a) aims to capture the long-range word relationship by graph network. **REAL** (Huang et al., 2021) proposes a analogical auxiliary learning strategy by extracting similar MWPs. **BERT-CL** (Li et al., 2021) uses contrastive learning with PLMs. **RP-KHS** (Yu et al., 2021) performs hierarchical reasoning with PLMs. **MWP-BERT** released a BERT-based encoder that is continually pre-trained on MWP corpus. **Gen&Rank** (Shen et al., 2021) designs a multi-task learning framework with encoder-decoder pre-training. **MWPtoolkit** finds a RoBERTa-to-RoBERTa model has the best performance in MWP solving.

3.3 Experimental Results

As Table 1 shows, our approach outperforms all other RNN-based baselines in terms of answer accuracy and achieves new state-of-the-art. On Math23k, we outperform the latest RNN-based push from Wu et al. (2021a) by 1.8%. For the first time, an RNN-based MWP solver reaches over 80% answer accuracy on the Math23k dataset. What is more, the even fewer parameters with the best performance suggest that our model is also memory-efficient by separating the encoder into two modals.

PLM-based solvers benefit from the pre-training on a huge amount of corpus and thus achieve great semantic understanding ability. From a different point of view, our work aims to effectively and efficiently integrate semantic and quantity understanding. Therefore, by incorporating the MWP-BERT model as our semantic extractor, the answer accuracy of C-MWP achieves state-of-the-art performance. It proves the feasibility of combining PLM-based semantic modal encoder and graph-based quantity modal encoder, which will be an interesting inspiration to the community.

Ablative Study of Different Components. In order to evaluate the effectiveness of each component in *C-MWP*, we report the model performance after removing several components. Compared with **Graph2Tree**, our compositional structure and dynamic fusion module allow the full usage of both modals and excel in improving performance.

Performance on Different Types of MWP. In order to investigate how our model performs across various types of MWP, we introduce a new split of Math23k with regard to three types of problems: story problems, algebra problems and knowledge problems. Split details are shown in the appendix. The evaluation results are presented in Table 3. Without a compositional manner, Graph2Tree and Multi-E/D perform better than GTS on story and algebra testing problems, whereas they perform worse on knowledge problems. As stated before, *one joint modal cannot do it all*. These baselines work well on some types of problems while having weak performance on other types of problems. Our C-MWP offers a general improvement over all types of MWPs, which firmly supports our motivation for alleviating the generalization issue.

4 Conclusion and Future Work

The semantic meaning and quantity information are important intrinsic properties of a math word problem. Aiming at dealing with uni-modal bias and achieve better generalization, we make the first attempt to propose a compositional MWP solver, *C-MWP*. Multi-step reasoning and specified training methods are leveraged to enhance the learn-from-components ability of the model. As the method proposed in this paper could be applied in a broader range of neuro-symbolic learning problems, we will keep exploring the adaptiveness of this compositional encoding method.

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 *NAACL*, pages 2357–2367.

Xinyun Chen, Chen Liang, Adams Wei Yu, Dawn Song, and Denny Zhou. 2020. Compositional generalization via neural-symbolic stack machines. *NeurIPS*, 33:1690–1701.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *EMNLP*, pages 1724–1734.

Shifeng Huang, Jiawei Wang, Jiao Xu, Da Cao, and Ming Yang. 2021. Recall and learn: A memory-augmented solver for math word problems. In *Findings of EMNLP*, pages 786–796.

Zhanming Jie, Jierui Li, and Wei Lu. 2022. Learning to reason deductively: Math word problem solving as complex relation extraction. In *ACL*, pages 5944–5955.

Rik Koncel-Kedziorski, Hannaneh Hajishirzi, Ashish Sabharwal, Oren Etzioni, and Siena Dumas Ang. 2015. Parsing algebraic word problems into equations. *ACL*, 3:585–597.

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

Yihuai Lan, Lei Wang, Qiyuan Zhang, Yunshi Lan, Bing Tian Dai, Yan Wang, Dongxiang Zhang, and Ee-Peng Lim. 2022. Mwptoolkit: An open-source framework for deep learning-based math word problem solvers. In *AAAI*, volume 36, pages 13188–13190.

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 *ACL*, pages 6162–6167.

Zhongli Li, Wenxuan Zhang, Chao Yan, Qingyu Zhou, Chao Li, Hongzhi Liu, and Yunbo Cao. 2021. Seeking patterns, not just memorizing procedures: Contrastive learning for solving math word problems. *arXiv preprint arXiv:2110.08464*.

Zhenwen Liang, Jipeng Zhang, Lei Wang, Wei Qin, Yunshi Lan, Jie Shao, and Xiangliang Zhang. 2022. MWP-BERT: Numeracy-augmented pre-training for math word problem solving. In *Findings of NAACL*, pages 997–1009.

Zhenwen Liang and Xiangliang Zhang. 2021. Solving math word problems with teacher supervision. In *IJCAI*, pages 3522–3528.

Xin Lin, Zhenya Huang, Hongke Zhao, Enhong Chen, Qi Liu, Hao Wang, and Shijin Wang. 2021. Hms: A hierarchical solver with dependency-enhanced understanding for math word problem. In *AAAI*, pages 4232–4240.

Shen-Yun Miao, Chao-Chun Liang, and Keh-Yih Su. 2020. A diverse corpus for evaluating and developing english math word problem solvers. In *ACL*, pages 975–984.

Jianhao Shen, Yichun Yin, Lin Li, Lifeng Shang, Xin Jiang, Ming Zhang, and Qun Liu. 2021. Generate & rank: A multi-task framework for math word problems. In *EMNLP*, pages 2269–2279.

Yibin Shen and Cheqing Jin. 2020. Solving math word problems with multi-encoders and multi-decoders. In *COLING*, pages 2924–2934.

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

Charles Welch, Jonathan K Kummerfeld, Verónica Pérez-Rosas, and Rada Mihalcea. 2020. Compositional demographic word embeddings. In *EMNLP*, pages 4076–4089.

Francis CK Wong and William SY Wang. 2007. Generalisation towards combinatorial productivity in language acquisition by simple recurrent networks. In *2007 International Conference on Integration of Knowledge Intensive Multi-Agent Systems*, pages 139–144. IEEE.

Qinzhuo Wu, Qi Zhang, and Zhongyu Wei. 2021a. An edge-enhanced hierarchical graph-to-tree network for math word problem solving. In *Findings of EMNLP*, pages 1473–1482.

Qinzhuo Wu, Qi Zhang, Zhongyu Wei, and Xuanjing Huang. 2021b. Math word problem solving with explicit numerical values. In *ACL*, pages 5859–5869.

Zhipeng Xie and Shichao Sun. 2019. A goal-driven tree-structured neural model for math word problems. In *IJCAI*, pages 5299–5305.

Jingfeng Yang, Le Zhang, and Diyi Yang. 2022. SUBS: Subtree substitution for compositional semantic parsing. In *NAACN*, pages 169–174, Seattle, United States. Association for Computational Linguistics.

Weijiang Yu, Yingpeng Wen, Fudan Zheng, and Nong Xiao. 2021. Improving math word problems with pre-trained knowledge and hierarchical reasoning. In *EMNLP*, pages 3384–3394.

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 *ACL*, pages 3928–3937.

Appendix

Related Work

Compositional Learning in NLP. Modeling compositionality in language has been a long-standing issue (Wong and Wang, 2007) in NLP community. One common practice is to perform disentanglement over language representations at different levels (Welch et al., 2020).. They usually focus on atomic semantics units like character, word and phrase. As logic form annotations naturally own compositional features, compositionality is incorporated in generating correct logic contents. Therefore, the compositionality is often injected into traditional semantic parsing tasks(Chen et al., 2020; Yang et al., 2022) where the goals during training can be decomposed and then reorganized as a novel goal.

Our work firstly tries to inject compositional prior into MWP encoding. It is worth noting that MWP solving owns the same well-organized logic form annotations as machine reasoning, which naturally requires compositionality.

Math Word Problem Solving. Earlier MWP solvers parse problem descriptions semantically, and learn templates for generating answers (Koncel-Kedziorski et al., 2015). Recent works (Wang et al., 2017; Xie and Sun, 2019; Li et al., 2019; Zhang et al., 2020; Shen and Jin, 2020; Wu et al., 2021b,a; Lin et al., 2021; Liang and Zhang, 2021; Jie et al., 2022) focus on employing the encoder-decoder framework (e.g., sequence-to-sequence, sequence-to-tree, graph-to-tree) to translate MWP texts into equations based on traditional RNN structure. There are also new settings (Amini et al., 2019; Miao et al., 2020) introduced to extend MWP solving in equation group generation and diagnosing awareness of external knowledge. Nowadays, many researchers build strong MWP solvers upon pre-trained language models (PLMs) (Huang et al., 2021; Li et al., 2021; Yu et al., 2021; Shen et al., 2021; Lan et al., 2022) and have achieved great performance. Differently, our work lays the groundwork of feature extraction of quantity modal, which is orthogonal to those works.

In this work, we not only propose an explicit compositional encoding module with a multi-step design, but also incorporate detailed analysis to verify its compositional learning ability, to jointly leverage semantic and quantity information to achieve effective MWP understanding.

Implementation Details

We train our model on an NVIDIA RTX 2080Ti GPU, all implementation [1] of training and testing is coded in Python with Pytorch framework. For our RNN-based model, 2 CMEncoders are stacked and only 1 CMEncoder is used in the PLM-based model. The size of hidden dimensions in encoders and decoders are set to 512 and 768 for RNN-solver and PLM-solver, respectively. Each GCN block has 2 GraphConv layers and each GNN encoder has 4 heads of GCN blocks. During training, Adam optimizer is applied with the initial learning rates of 0.001/0.00003 for RNN/PLM, which would be halved every 30 epochs. During testing, we use a 5-beam search to get reasonable solutions. We also apply Gaussian noise with mean 0 and variance 1 on the embedding result during training. This simple operation can help models to learn more robust parameters. Through grid search at 0.1 level, the noise is multiplied by 0.2 to achieve the best performance.

Hyper-Parameter Tuning

In general, we apply grid-search with manually designed search space and use answer accuracy as the evaluation metric to select the hyper-parameters. For the number of stacked encoders, the search space is $\{1, 2, 3, 4\}$ and we finally use 2. For the weight of L_2 normalization loss, we choose weight 1 from $\{0.01, 0.1, 1, 5, 10\}$. The weight of random noise 0.2 is selected from 0.1 level by grid search with range 0 to 1. We also tune the beam size of beam search from $\{3, 4, 5, 6, 7\}$ and choose 5. The dropout probability 0.5 is selected from $\{0.1, 0.3, 0.5, 0.7\}$. Initial learning rate 0.001 is selected from $\{0.01, 0.001, 0.0001\}$. For the hidden size and embedding size in encoder, we select 256 from $\{64, 128, 256, 512\}$.

Variance and Significance Evaluation

We evaluated our solver with 5-fold cross-validation and found that the accuracy of our RNN-based C-MWP (0.779%±0.028) is significantly higher than Graph2Tree (0.755%±0.016) (p<0.01), and the accuracy of our PLM-based C-MWP (0.845%±0.21) is significantly higher than vanilla MWP-BERT (0.824%±0.016) (p<0.01).

[1] We will release all the materials including code and data after this paper is published.

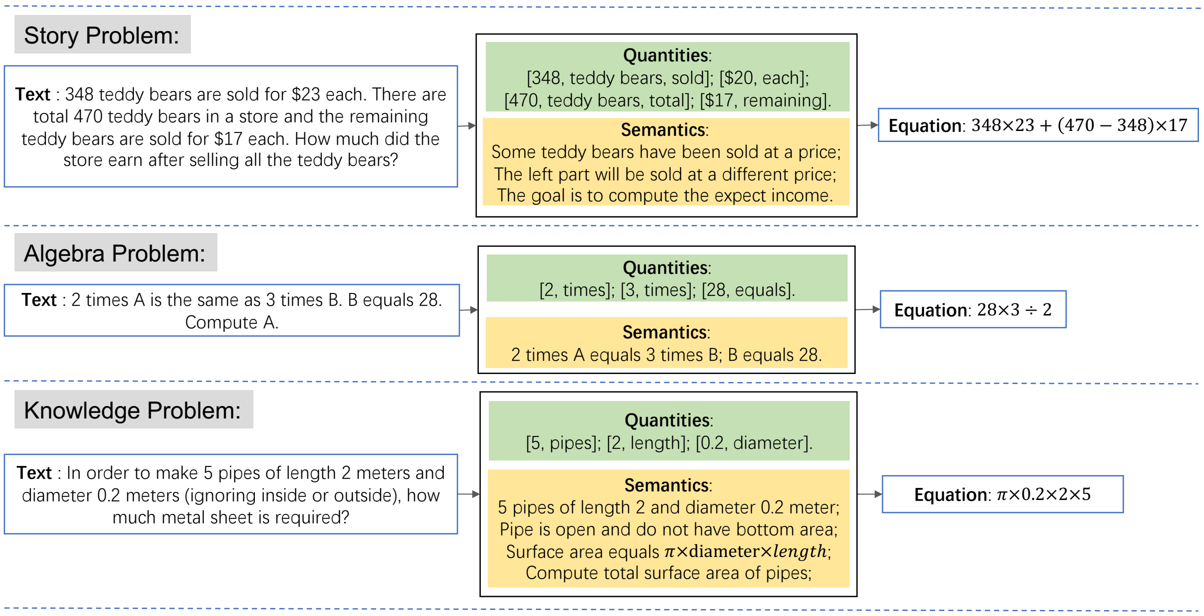

Figure 2: Examples of different types of problems in MWP solving.

	Keywords
Knowledge	半径, 圆, 直径, 周长, 三角形, 正方形, 长方形, 平行四边形, 梯形, 边长, 体积, 侧面积, 横截面, 表面积, 圆柱, 长方体, 正方体, 数列, 内项, 外项,
Story	速度, 千米, 路程, 相距, 全程, 相遇, 相对, 相反, 相背, 相向, 修, 利润, 间隔, 隔, 放假, 利息, 利率, 超产, 减产, 买, 钱, 定价, 出售, 书, 树, 广场, 大桥, 服装, 衣, 蛋, 身高, 山, 电线, 绳子, 化肥, 同学, 机关, 油, 花, 钢, 车, 厂, 山庄, 煤, 学生, 植物, 蜂蜜, 水泥, 皂, 沙, 港, 班, 师, 学, 园, 月, 海, 票, 巧克力, 米, 兔, 养, 价, 超市, 运动, 体重, 产量, 县, 工, 货, 桃子, 岁, 校, 箱, 座, 字, 瓜, 生, 父, 母, 爸爸, 妈妈, 鸡, 鸭, 水, 球, 木, 桌, 麦, 动画, 玩具, 音乐, 美术, 羊, 邮件, 村, 拖拉机, 页, 绳, 猪, 象, 金, 朋友, 电影, 茶, 芽, 碟, 年, 菜, 人, 电, 布, 光盘, 粮, 食, 苹果, 糖果, 果树, 如果, 农场, 纸, 猴, 牛, 糖, 猫, 河, 果汁, 罐子, 珠子, 市场, 齿轮, 饮料, 口算, 比赛, 鸽, 鹅, 面, 奶, 英语, 粉, 青蛙, 环境, 俑, 带, 马虎, 测试, 奥运, 楼, 梯, 时间, 飞机, 氧气, 梨, 橘, 桔, 河泥, 竞赛, 虫, 毽, 旅游, 参赛, 年级, 营业, 纳税, 葡萄, 图, 草, 旗, 饼, 果酱, 衬衫, 船, 铁, 挣, 商场, 鱼, 田, 农, 石, 弟, 哥, 公司, 叔, 伯, 鞋, 林, 标本, 作业, 耕, 动物, 画, 豆, 兵, 蝶, 品, 味精, 棒, 袜, 盘, 酒, 店, 出口, 棋, 成绩, 文具, 盒, 温, 药, 竞赛, 计划, 浓度, 表, 区, 矿, 邮, 剪, 塔, 芒果, 房, 棵, 奖, 客, 网, 户, 元, 币, 筐, 中国, 账, 走, 成本, 蜜蜂, 信封, 砖, 醋, 山, 土, 爷, 卡片,

Figure 3: Keywords for "Story" and "Knowledge" problems. Problems that do not fall into "Story" and "Knowledge" are labeled as "Algebra" problems.

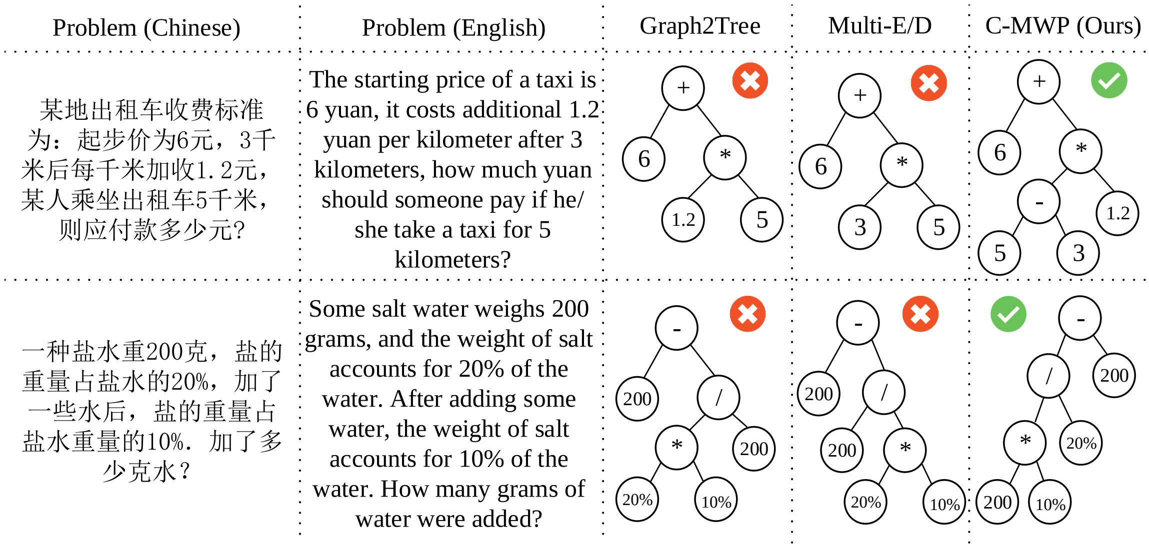

Figure 4: Case study from Math23k

	Overall	Story	Algebra	Knowledge
train	21,162	17,546	2,595	957
val	1,000	817	133	50
test	1,000	842	116	42

Table 4: Statistics of different types of problems in Math23k.

Case Study

Figure 4 shows generated solutions of two selected problems by Graph2Tree (Zhang et al., 2020), Multi-E/D (Shen and Jin, 2020) and our proposed *C-MWP* (RNN-Based). The first problem has 4 quantities and they are all useful, which means that it requires sufficient problem understanding and mathematical reasoning to generate the right answer. Both Graph2Tree and Multi-E/D which directly connect semantics modal and quantity modal fail to extract clear representations of the problem, finally resulting in unreasonable solutions which only contain 3 quantities. For the second problem, although Graph2Tree and Multi-E/D utilize all 3 quantities in the problem description, they still fail to generate a plausible solution. These two cases show that our proposed encoder is able to extract more comprehensive representations from problem descriptions, eventually guiding the decoder to generate the correct solutions.

MWPs in Different Categories

Figure 2 shows the MWP examples of "Story Problem", "Algebra Problem" and "Knowledge Problem". "Story Problem" often includes a significant amount of background information like characters, objectives and behaviors. "Algebra Problems" involves math notations or is composed of elementary concepts. "Knowledge Problem" asks for external knowledge like geometry and number sequence. The category of each problem is determined based on keywords. The keywords of "Story" and "Knowledge" problems are listed in Figure 3. Other problems that do not fall into "Story" and "Knowledge" are labeled as "Algebra" problems. The statistics of these problems are shown in Table 4.

Limitations

Explainability Most current MWP solvers are only able to generate solutions. In our work, although we achieved better generalization ability, it is still hard to explain how the model solves MWPs both correctly or incorrectly. These automated solvers would be much more helpful for tutoring students if they could explain their equation solutions by generating reasoning steps.