Improving Math Word Problems with Pre-trained Knowledge and Hierarchical Reasoning

Weijiang Yu, Yingpeng Wen, Fudan Zheng, Nong Xiao*
School of Computer Science and Engineering, Sun Yat-sen University
weijiangyu8@gmail.com, {wenyp6,zhengfd3}@mail2.sysu.edu.cn,
xiaon6@mail.sysu.edu.cn

Abstract

The recent algorithms for math word problems (MWP) neglect to use outside knowledge not present in the problems. Most of them only capture the word-level relationship and ignore to build hierarchical reasoning like the human being for mining the contextual structure between words and sentences. In this paper, we propose a **R**easoning with **P**re-trained **K**nowledge and **H**ierarchical **S**tructure (**RPKHS**) network, which contains a pre-trained knowledge encoder and a hierarchical reasoning encoder. Firstly, our pre-trained knowledge encoder aims at reasoning the MWP by using outside knowledge from the pre-trained transformer-based models. Secondly, the hierarchical reasoning encoder is presented for seamlessly integrating the word-level and sentence-level reasoning to bridge the entity and context domain on MWP. Extensive experiments show that our RPKHS significantly outperforms state-of-the-art approaches on two large-scale commonly-used datasets, and boosts performance from 77.4% to 83.9% on Math23K, from 75.5 to 82.2% on Math23K with 5-fold cross-validation and from 83.7% to 89.8% on MAWPS. More extensive ablations are shown to demonstrate the effectiveness and interpretability of our proposed method.

1 Introduction

Math Word Problem (MWP) is a reasoning task for answering a mathematical query based on the problem description, which is an interdisciplinary research topic to bridge the mathematics and natural language processing. As shown in Table 1, a short narrative is presented to describe a problem and poses a question about the unknown quantity. In recent years, research on MWP by using deep learning methods has been gaining increasing attention. Early research mainly focuses on Seq2Seq-based models (Sutskever et al., 2014; Ling et al., 2017b; Wang et al., 2017; Huang et al., 2018; Wang et al., 2017). These Seq2Seq-based methods aim to train an end-to-end model from scratch by using the training dataset. Some research focuses on developing structure-based approaches (Xie and Sun, 2019a; Wang et al., 2018a, 2019b; Liu et al., 2019a; Zhang et al., 2020b; Li et al., 2020b; Hong et al., 2021; Li et al., 2020a) by incorporating parsing tree into the neural models to produce promising results in generating solution expression for the MWP.

Problem: Conner has 25000 dollars in his bank account. Every month he spends 1500 dollars. He does not add money to the account. How much money will Conner have in his account after 8 months?
Equation: $x = 25000.0 - (1500.0 * 8.0)$; **Solution:** 13000.0

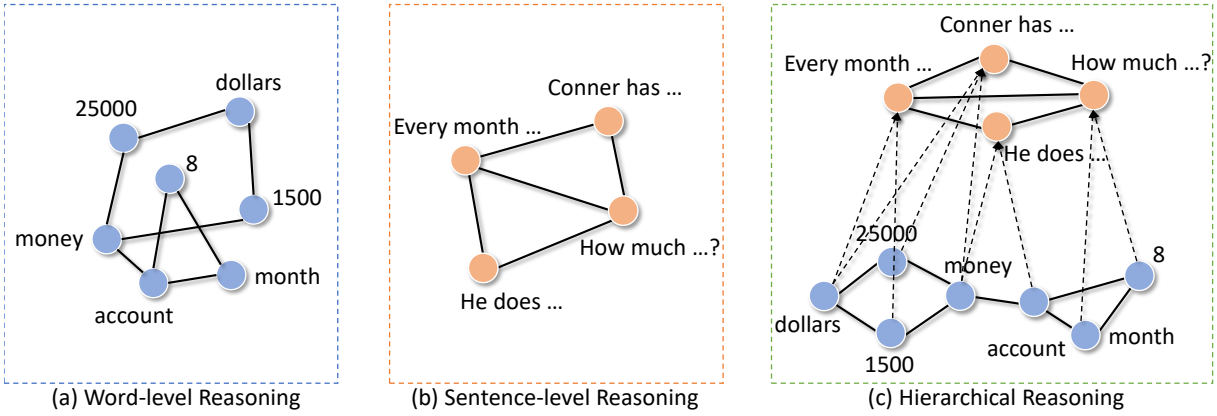
Table 1: The example of the math word problem task. Given a natural language description for a mathematical problem, it requires the model to infer a formal math equation and final quantity solution.

To answer this question, human beings not only need to parse the question and understand the context but also use external knowledge. However, the previous methods learn the textual description purely from the short and limited narrative without using any background knowledge that not present in the description, which restrain the ability of the models for inferring the MWP from a global perspective. Moreover, current methods mainly focus on designing diverse entity-level structures for word-level reasoning rather than bridging the hierarchical reasoning between the entity (word-level) and context (sentence-level). Obviously, it is not enough to use single-level reasoning for solving the MWP. In this paper, we propose reasoning with pre-trained knowledge and hierarchical structure (RPKHS) to jointly solve the two limitations.

Our RPKHS as shown in Figure 2 consists of

*Corresponding Author

Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pages 3384–3394
November 7–11, 2021.



(a) Word-level Reasoning (b) Sentence-level Reasoning (c) Hierarchical Reasoning

Figure 1: (a) Word-level reasoning is to build the relationship of each word in all textual descriptions, which can also be considered as entity-level reasoning; (b) Sentence-level reasoning aims at mining the intra-relationship of each sentence from the paragraph. (c) Hierarchical reasoning is to jointly excavate intra-relationship and inter-relationship between word and sentence from the same paragraph.

two encoders, namely pre-trained knowledge encoder and hierarchical reasoning encoder, and a tree-structured decoder. It effectively incorporates the *implicit* linguistic knowledge into the model via pre-trained knowledge encoder and generates *structural* representation by our hierarchical reasoning encoder. The outputs of the two encoders are fed into a tree-structured decoder (Xie and Sun, 2019b) for final prediction.

To the best of our knowledge, we are the first one to study the application of pre-trained knowledge to the MWP task. We have implicit knowledge which is embedded into some non-symbolic form such as the weights of a neural network derived from annotated data or large-scale unsupervised language training. Recently, Transformer-based (Vaswani et al., 2017) and specifically BERT-based (Devlin et al., 2019b; Liu et al., 2019c) models have been proposed, which incorporate large-scale linguistic pre-training, implicitly capturing language-based knowledge. This type of knowledge can be quite useful for parsing the textual description.

For example, there are two sentences: 'He has 25000 dollars in his bank account.'; 'Paul appeared before the faculty to account for his various misdemeanors'. The word 'account' has totally different meanings between the two sentences due to different scene-aware descriptions. Hence, we think such diverse semantics of each word containing rich representation in the implicit pre-trained knowledge. Such knowledge can be also regarded as a huge implicitly vocabulary to endow each word with rich representation. It can help the model to parse the correct semantics of words from complex text. In this paper, we take advantage of the implicit knowledge in pre-trained Roberta (Liu et al., 2019c) and analyze the effect of various pre-trained knowledge on the MWP task.

Current methods mainly learn the MWP by building word-level reasoning (as shown in Figure 1 (a)) by GNN (Zhang et al., 2020b; Li et al., 2020b) and Seq2Seq model (Wang et al., 2017). They seldom consider modeling hierarchical structure. Since the descriptions of MWP have a hierarchical structure (words from sentences, sentences from a narrative), we likewise construct hierarchical reasoning (as shown in Figure 1 (c)) by first building representations of sentences from words, and then aggregating those into a whole narrative representation.

It is observed that different words and sentences in a mathematical narrative are differentially informative. The importance of words and sentences are highly context-dependent, i.e. the same word or sentence may be differentially important in different contexts (e.g., 5 dollars and 5 pencils, the word of 5 has different meanings.). To include sensitivity to this fact, our model includes two levels of reasoning mechanisms. One at the word-level and one at the sentence-level. They lead the model to pay more or less attention to individual words and sentences when constructing the representation of the narrative. Taking an example as shown in Table 1, intuitively, the first, second and fourth sentences have stronger information in assisting the prediction of the solution. Within these sentences, the words *25000 dollars* and *every month* contribute more in inferring the math-aware results. In this

paper, we propose a hierarchical reasoning encoder to achieve this functionality.

Contributions. (1) As far as we know, we are the first one to explore pre-trained knowledge on the MWP task via our pre-trained knowledge encoder. (2) We propose a hierarchical reasoning encoder to seamlessly integrate the word-level and sentence-level reasoning for bridging the entity and context domain on MWP. It can provide insight into which words and sentences contribute to the prediction which can be of value in applications and analysis. (3) Our RPKHS outperforms previous approaches by a significant margin.

2 Related Work

The MWP is the task of translating a short paragraph consisting with multiple short sentences into target mathematical equations. Previous approaches usually solve the MWP by using rule-based methods (Yuhui et al., 2010; Bakman, 2007), statistical machine learning methods (Kushman et al., 2014; Mitra and Baral, 2016; Roy and Roth, 2018; Zou and Lu, 2019), semantic parsing methods (Shi et al., 2015; Roy and Roth, 2015; Huang et al., 2017) and deep learning methods (Ling et al., 2017a; Wang et al., 2018b; Liu et al., 2019b; Wang et al., 2017; Zhang et al., 2020a). Recently, the deep learning based methods have been paid more attention for their significant improvement. (Wang et al., 2017) proposed a Seq2Seq-based model to directly map the linguistic text to a solution. (Wang et al., 2018b) and (Chiang and Chen, 2019) implicitly modeled tree-based structure for decoding the MWP expressions, while (Wang et al., 2019a; Liu et al., 2019b; Xie and Sun, 2019b) optimized the decoder via explicit tree structure. Some research focused on graph structure on word-level reasoning. For example, (Zhang et al., 2020a) built two customized graphs for enriching the quantity representations in the problem. (Li et al., 2020b) presented a graph-to-tree encoder-decoder framework for grammar parsing.

However, they ignore the sentence-level relationship and the correlation between word and sentence. Different from the previous methods, we propose to use hierarchical reasoning containing word-level and sentence-level reasoning. Besides, we are the first ones to explore the effect of implicit knowledge from the pre-trained neural network weights on the task of math word problems.

3 Methodology

3.1 Overview

In this section, we explain the architecture and design of our proposed RPKHS network (i.e. Reasoning with Pre-trained Knowledge and Hierarchical Structure) composed of pre-trained knowledge encoder, hierarchical reasoning encoder and tree-structured decoder, which can appropriately incorporate the outside knowledge into the model and bridge the hierarchical reasoning between the entity (word-level) and context (sentence-level). The overview of our RPKHS is illustrated in Figure 2. Our contributions mainly focus on the design of a joint-learning framework and two innovative encoders on the MWP task, which are unveiled and discussed in detail in the following sections.

3.2 Problem Formulation

The math word problems (MWP) can be formulated as (P, E), where P is the problem text and E is a solution expression. Assuming a description of MWP has L sentences s_i, and each sentence contains T_i words. w_{it} with $t \in [1, T]$ represents the words in the i-th sentence. Our proposed encoders project the raw problem descriptions into a vector representation, on which we build a tree-structured decoder to predict the mathematical expression.

3.3 Pre-trained Knowledge Encoder

We want to incorporate implicit external knowledge as well as math-aware knowledge which can be learned from the training set in our model. Language models, and especially transformer-based language models, have shown to contain commonsense and factual knowledge (Petroni et al., 2019; Jiang et al., 2019). We adopt this direction in our model and build an encoder, pre-trained with Roberta (Liu et al., 2019c), which has been pre-trained on the huge language corpora (e.g., BooksCorpus (Zhu et al., 2015), Wikipedia (Remy, 2002)) to capture implicit knowledge. We tokenize a description $\mathcal{Q}$ using WordPiece (Wu et al., 2016) as in BERT (Devlin et al., 2019a), giving us a sequence of $|\mathcal{Q}|$ tokens and embed them with the pre-trained Roberta embeddings and append Roberta's positional encoding, giving us a sequence of d-dimensional token representation $x^{\mathcal{Q}}_1, ..., x^{\mathcal{Q}}_{|\mathcal{Q}|}$. We feed these into the transformer-based pre-trained knowledge encoder, fine-tuning the representation during training. We mean-pool the output of all

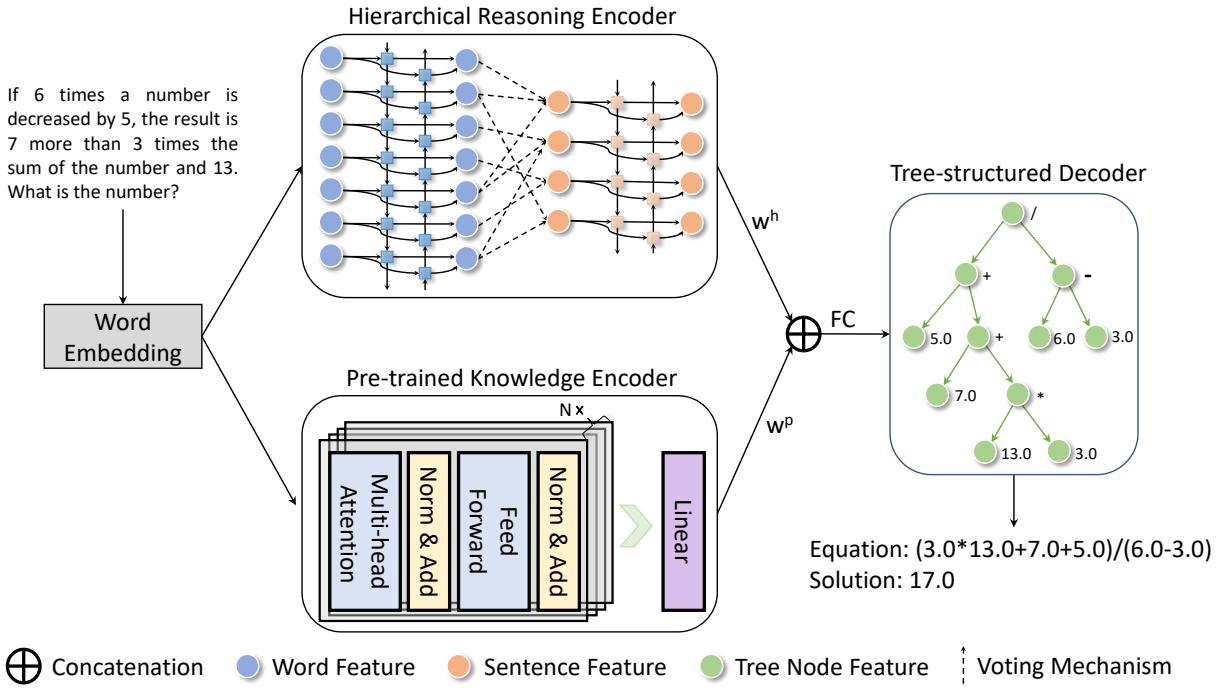


Figure 2: Overview of our Reasoning with Pre-trained Knowledge and Hierarchical Structure (RPKHS). The hierarchical reasoning encoder receives the textual embedding to construct inter-relationship between sentence and word to aggregate semantics among entity and context. The pre-trained knowledge encoder captures a large amount of knowledge about the linguistic world from the pre-trained network weights, and incorporates the implicit knowledge into the input embedding to enrich the input representation. Then we concatenate the results from two encoders as the input of a tree-structured decoder for parsing the target mathematical equation and solution.

transformer steps to get our combined implicit knowledge representation $\mathbf{Y}_p$.

3.4 Hierarchical Reasoning Encoder

The proposed hierarchical reasoning encoder takes into account that the different parts of a math description have no similar relevant information. Moreover, determining the relevant sections involves modeling the interactions among the words, not just their isolated presence in the text. Therefore, to consider this aspect, the model includes two levels of reasoning mechanisms. One reasoning at the word level and the other at the sentence level, which let the model pay more or less attention to individual words and sentences when constructing the whole description representation. The hierarchical reasoning encoder is composed of 2 layers. The first layer is our word-level reasoning layer and the second layer is the sentence-level reasoning layer. In the following sections, we first introduce the GRU-based operation commonly used in our two layers. Then we present the details of the two reasoning layers.

GRU-based Sequence Encoding. The GRU (Bahdanau et al., 2015) uses a gating mechanism to track the state of sequences without using separate memory cells. There are two types of gates: the reset gate r_t and the update gate z_t. They jointly control how information is updated to the state. At time t, the GRU computes the new state as

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \hat{h}_{t-1}. \quad (1)$$

This is a linear interpolation between the previous state h_{t-1} and the current new state $\hat{h}_t$ computed with new sequence information. The gate z_t decides how much past information is kept and how much new information is added. z_t is updated as

$$z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z), \quad (2)$$

where x_t is the sequence vector at time t. The candidate state $\hat{h}_t$ is computed by

$$\hat{h}_t = tanh(W_h x_t + r_t \odot (U_h h_{t-1}) + b_h), \quad (3)$$

where r_t is the reset gate which controls how much the previous state contributes to the candidate state. If r_t is zero, then it forgets the past state. The reset gate is updated by

$$r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r). \quad (4)$$

The W and U mean the learnable matrix weights and the b is the learnable bias vector.

Word-level Reasoning. In this layer, the model uses bidirectional GRU (Bahdanau et al., 2015) to produce representation of words by summarizing information from both directions. Therefore, it incorporates the contextual information in the word-level representation. Given a sentence with words $w_{it}, t \in [1, T]$ and an embedding matrix W_e, a bidirectional GRU contains the forward GRU $\overrightarrow{f}$ which reads the sentence s_i from w_{i1} to w_{iT} and a backward GRU $\overleftarrow{f}$ which reads from w_{iT} to w_{i1} by using

$$x_{it} = W_e w_{it}, t \in [1, T], \tag{5}$$
$$\overrightarrow{h_{it}} = \overrightarrow{f}(x_{it}), t \in [1, T], \tag{6}$$
$$\overleftarrow{h_{it}} = \overleftarrow{f}(x_{it}), t \in [T, 1]. \tag{7}$$

The word-level representation for a given word w_{it} is obtained by concatenating the forward hidden state and backward hidden state, i.e., $h_{it} = [\overrightarrow{h_{it}}, \overleftarrow{h_{it}}]$, which summarizes the information of the whole sentence centered around w_{it}. Not all words contribute equally to the representation of the sentence meaning. Hence, we introduce an attention mechanism to extract such words that are important to the meaning of the sentence and aggregate the representation of those informative words to form a sentence vector. Specifically,

$$u_{it} = \tanh(W_w h_{it} + b_w), \tag{8}$$
$$\alpha_{it} = \frac{\exp(u_{it}^\top u_w)}{\sum_t \exp(u_{it}^\top u_w)}, \tag{9}$$
$$s_i = \sum \alpha_{it} h_{it}. \tag{10}$$

We first feed the word-level feature h_{it} through a one-layer MLP to get u_{it} as a hidden representation of h_{it}. Then we measure the importance of the word as the similarity of u_{it} with a word-level context vector u_w and get a normalized importance weight α_{it} through a softmax function. After that, we compute the sentence vector s_i as a weighted sum of the word representations based on the learnable weights. The word context vector u_w in Eq. 9 can be seen as a high-level representation of a fixed query like "what is the informative word" over the words. It is inspired by the memory networks (Kumar et al., 2016). The word context vector u_w is randomly initialized and jointly learned during the training process.

Sentence-level Reasoning. Given the sentence vectors s_i, we get a problem description vector in an analogical way. We use a bidirectional GRU to encode the sentences:

$$\overrightarrow{h_{it}} = \overrightarrow{f}(s_i), i \in [1, L], \tag{11}$$
$$\overleftarrow{h_{it}} = \overleftarrow{f}(s_i), i \in [L, 1], \tag{12}$$

where $\overrightarrow{f}$ and $\overleftarrow{f}$ mean the forward GRU and backward GRU, respectively. We concatenate $\overrightarrow{h_i}$ and $\overleftarrow{h_i}$ to get the target representation of sentence i, i.e. $h_i = [\overrightarrow{h_i}, \overleftarrow{h_i}]$. The h_i summarizes the neighbor sentences around sentence i but still focus on sentence i. To reward sentences that are relevant to correctly parse the problem description, we again use attention mechanism and introduce a sentence level context vector u_s to measure the importance of the sentences, which can be formulated as

$$u_{it} = \tanh(W_s h_i + b_s), \tag{13}$$
$$\alpha_{it} = \frac{\exp(u_i^\top u_s)}{\sum_i \exp(u_i^\top u_s)}, \tag{14}$$
$$v = \sum \alpha_i h_i, \tag{15}$$

where v is the global text vector that summarizes all the information of sentences in a description. Similarly, the sentence-level context vector u_s can be randomly initialized and jointly learned during the training process.

Merging Mechanism. After getting the results $\mathbf{Y}_p$ and $\mathbf{Y}_h$ from pre-trained knowledge encoder and hierarchical reasoning encoder, respectively, we utilize a parser at the end of two encoders as shown in Figure 2 to adaptively merge $\mathbf{Y}_p$ and $\mathbf{Y}_h$ to get an enhanced representation $\mathbf{Y}$ for final decoding. The parser can be formulated as

$$\mathbf{Y} = F([w^p \mathbf{Y}_p, w^h \mathbf{Y}_h]), \tag{16}$$

where w^p and w^h are derived from the $\mathbf{Y}_p$ and $\mathbf{Y}_h$ to calculate the importance of the task. [.] means concatenation operation. We use a simple dot product to merge the two representations ($\mathbf{Y}_p$ and $\mathbf{Y}_h$). Then we use linear mapping function F such as fully connected (FC) layer to produce the enhanced representation $\mathbf{Y}$ for final decoding. The w^p and

w^h can be calculated as

$$w^p = \frac{\exp(\varphi_p(\mathbf{Y}_p\mathbf{W}_p))}{\exp(\varphi_p(\mathbf{Y}_p\mathbf{W}_p)) + \exp(\varphi_h(\mathbf{Y}_h\mathbf{W}_h))}, \tag{17}$$

$$w^h = \frac{\exp(\varphi_h(\mathbf{Y}_h\mathbf{W}_h))}{\exp(\varphi_p(\mathbf{Y}_p\mathbf{W}_p)) + \exp(\varphi_h(\mathbf{Y}_h\mathbf{W}_h))}, \tag{18}$$

where $\mathbf{W}_p$ and $\mathbf{W}_h$ are both trainable weighted matrices, and φ_p and φ_h indicate different MLPs.

3.5 Decoder and Optimization

Tree-structured Decoder. Following the goal-driven tree structure (GTS) (Xie and Sun, 2019b), we apply a tree-structured decoder as shown in Figure 2 to leverage the outputs of our encoders for generating the tree-structured targets like mathematical equations. The math equation often consists of operators and quantities. Firstly, the quantity is defined as a leaf node and each operator node is required to have two child nodes. Then, the tree-structured decoder parses an equation expression by following the pre-order traversal ordering. Firstly, the most center operator is generated, followed by the left child node. The generation process is recursively used until the final leaf node is completed. Next, we similarly generate the right child nodes.

To achieve the above-mentioned tree generation, our model initializes the root node vector according to the global context representation $\mathbf{Y}$ from two encoders. The expression trees in our decoder contain three types of nodes: math operators V_{op}, constant quantities V_{con} that are those common-sense numerical values encountered in the target expression but not in the problem text (e.g. a rabbit has 4 legs.), and the numbers n_P encountered in problem P. For each token y in the target vocabulary V^{tar}, its token embedding $\mathbf{e}(y|P)$ is defined as

$$\mathbf{e}(y|P) = \begin{cases} \mathbf{M}_{op}(y) & \text{if } y \in V_{op} \\ \mathbf{M}_{con}(y) & \text{if } y \in V_{con} \\ \mathbf{h}^p_{loc}(y, P) & \text{if } y \in n_P \end{cases} \tag{19}$$

where $\mathbf{M}_{op}$ and $\mathbf{M}_{con}$ are two trainable word embedding matrices independent of the specific problem. However, for a numeric value in n_P, we take the corresponding hidden state $\mathbf{h}^p_{loc}$ from encoder as its token embedding, where $loc(y, P)$ is the index position of numeric value y in P. The constant quantities V_{con} and numbers n_P are always set to be in leaf nodes position. The math operators V_{op} take up the non-leaf positions. The representation of n_P is dependent on certain MWP descriptions. Because y should take the corresponding hidden state $\mathbf{h}^p_{loc}$ from the encoder outputs. The representations of V_{op} and V_{con} are independently obtained from by two embedding matrices $\mathbf{M_{op}}$ and $\mathbf{M_{con}}$.

In regard to the tree-structured decoder, we mainly followed the GTS (Xie and Sun, 2019b) to parse the root vector to math equations. Being the same with the decoder of GTS, we have prepared the candidates for operators and numbers in our target vocabulary. Then we used the root vector with trainable vectors iteratively to predict the probability of node token y from the target vocabulary. Then the specific y (operation, number, etc.) with the highest probability will be selected to replace with the tree node according to the rules in Equation (19).

Optimization. Since the MWP task can be formulated as (P, E), we define its loss function as $\mathcal{L}(E, P)$, which can be formulated as a sum of the negative log-likelihoods of probabilities for predicting t-node token y_t. Formally, the objective function of the training optimizer can be

$$\mathcal{L}(E, P) = \sum_{t=1}^{m} -logp(y_t|q_t, \mathbf{Y}_t, P), \tag{20}$$

where m denotes the size of E, q_t and $\mathbf{Y}_t$ are the target vector and its context vector at the t-th node. The p is calculated by distribution computation function in GTS (Xie and Sun, 2019b).

4 Experiments

In this section, we first introduce the data that we use and the state-of-the-art baselines that we compare against. Then we show the implementation details of our experiments. Next, we demonstrate our results in comparison with other methods and provide extensive analyses. Finally, we conduct ablation studies and show some visualizations to investigate the effectiveness of our proposed components of our model (Reasoning with Pre-trained Knowledge and Hierarchical Structure, RPKHS).

4.1 Datasets and Evaluation

Datasets. We evaluate our proposed RPKHS and compare it with other state-of-the-art methods on two commonly-used datasets, namely MAWPS (Koncel-Kedziorski et al., 2016) with 2,373 problems and Math23K (Wang et al., 2018b) containing 23,162 problems.

Evaluation. As other works do (Xie and Sun, 2019b), for two datasets, we also measure the performance of our proposed method via the solution accuracy. For the Math23K dataset, there are two settings for evaluation on the previous methods. One is evaluating the model on the test set (denoted as "Math23K" in Table 2). The other evaluation setting is using 5-fold cross-validation which is expressed in "Math23K*". We evaluate our model compared with other methods in both settings.

Methods	MAWPS (%)	Math23K (%)	Math23K* (%)
DNS	59.5	-	58.1
Math-EN	69.2	66.7	-
T-RNN	66.8	66.9	-
S-Aligned	-	-	65.8
GROUP-ATT	76.1	69.5	66.9
AST-Dec	-	69.0	-
GTS	82.6	75.6	74.3
HMS	76.1	80.3	-
IRE	-	76.7	-
Graph2Tree	83.7	77.4	75.5
EPT-L	84.5	-	-
RPKHS (Ours)	**89.8**	**83.9**	**82.2**

Table 2: Model comparison between our model and other state-of-the-art methods. The Math23K indicates the results on public test set and Math23K* denotes 5-fold cross-validation.

4.2 Implementation Details

We implement our proposed RPKHS via PyTorch (Paszke et al., 2019) and python3.6 to train and test our RPKHS in math word problems. All experiments are conducted on the Ubuntu 18.04 from a server with 4 Tesla V100 GPUs. The Nvidia CUDA of 10.1 and cuDNN of 7.5 are utilized for acceleration. Unless noted otherwise, settings are the same for all experiments.

We set the dimension of the word embedding to 128 and use the dimension of all hidden states for the other layers in our hierarchical reasoning encoder with 512. For our pre-trained knowledge encoder, we strictly follow the setting in (Liu et al., 2019c) and use their pre-trained weights as the initial weights in our pre-trained knowledge encoder. We utilize the aforementioned objective function $\mathcal{L}(E, P)$ for all experiments. We set batch size to be 64 for 4 GPUs with 0.5 dropout (Hinton et al., 2012) rate, and set the weight decay as 1e-5 to prevent overfitting. We use Adam optimizer (Kingma and Ba, 2015) with an initial learning rate set to 0.0001 on pre-trained knowledge encoder and set to 0.001 on other parts of our model. The β_1 and β_2 in our optimizer are set as 0.94 and 0.99, respectively. We adopt plateau learning rate scheduler that reduces the learning rate by half every 20 epoch. Our model is trained for 80 epochs. The beam size is set to be 5 in beam search to generate expression trees, which is inspired by the GTS (Xie and Sun, 2019b).

Word-level	Sentence-level	pre-trained	Math23K (%)
			74.9
√			75.8
	√		76.1
		√	80.1
√	√		79.8
√		√	81.4
	√	√	82.3
√	√	√	83.9

Table 3: Ablation studies on Math23K test set. The 'word-level', 'sentence-level' and 'pre-trained' means the word-level reasoning, sentence-level reasoning and pre-trained knowledge encoder, respectively.

4.3 Results and Analyses

Comparison. We report experimental results on two benchmark datasets and compare these results with several state-of-the-art methods, which are DNS (Wang et al., 2017), Math-EN (Wang et al., 2018b), T-RNN (Wang et al., 2019a), S-Aligned (Chiang and Chen, 2019), GROUP-ATT (Li et al., 2019), AST-Dec (Liu et al., 2019a), GTS (Xie and Sun, 2019b), HMS (Lin et al., 2021), IRE (Sahu et al., 2019), Graph2Tree (Zhang et al., 2020b) and EPT-L (Kim et al., 2020). As shown in Table 2, our proposed RPKHS consistently and considerably outperforms other state-of-the-art MWP methods by 6.1%, 6.5% and 6.7% performance gains, respectively on MAWPS, Math23K and Math23K* when compared with Graph2Tree. Our RPKHS performs 3.6% improvement better than HMS on Math23K, and even outperform EPT-L by around 5.3% accuracy. The superior performance further demonstrates the effectiveness of our model on the math word problems.

4.4 Ablation Studies

The effect of our proposed components. As shown in Table 3, we use a word embedding layer, a LSTM layer and a tree-structured decoder as our baseline model, which achieves 74.9% accuracy on Math23K test set. After adding our word-level reasoning, it can boost the accuracy by 0.9% from baseline. We analyze the effect of sentence-level reasoning and observe that it can promote the

Example 1: There are 22 students in a classroom, 12 like grape juice, 15 like orange juice, and 10 like both juices. How many students like neither juice? **GTS:** 22.0-(12.0+15.0)-10.0;(error) **Graph2Tree:** (12.0+15.0)-22.0+10.0;(error) **RPKHS (Ours):** 22.0-(12.0+15.0)+10.0;(correct)
Example 2: While playing at the arcade, Edward won 3 tickets playing 'whack a mole' and 5 tickets playing 'skee ball'. If he was trying to buy candy that cost 4 tickets a piece, how many could he buy? **GTS:** 3.0+5.0/4.0;(error) **Graph2Tree:** (5.0-3.0)/4.0;(error) **RPKHS (Ours):** (3.0+5.0)/4.0;(correct)
Example 3: How many liters of a blue dye that costs 1.80 dollars per liter must be mixed with 20 liters of Anil, which costs 2.60 dollars per liter, to make a mixture that costs 2.12 dollars per liter? **GTS:** (20.0*2.6-20.0*2.12)/(2.12-2.6);(error) **Graph2Tree:** (20.0*2.6-20.0*2.12)/(2.6-1.8);(error) **RPKHS (Ours):** (20.0*2.6-20.0*2.12)/(2.12-1.8);(correct)

Table 4: Three examples of solving MWP with GTS, Graph2Tree and RPKHS models.

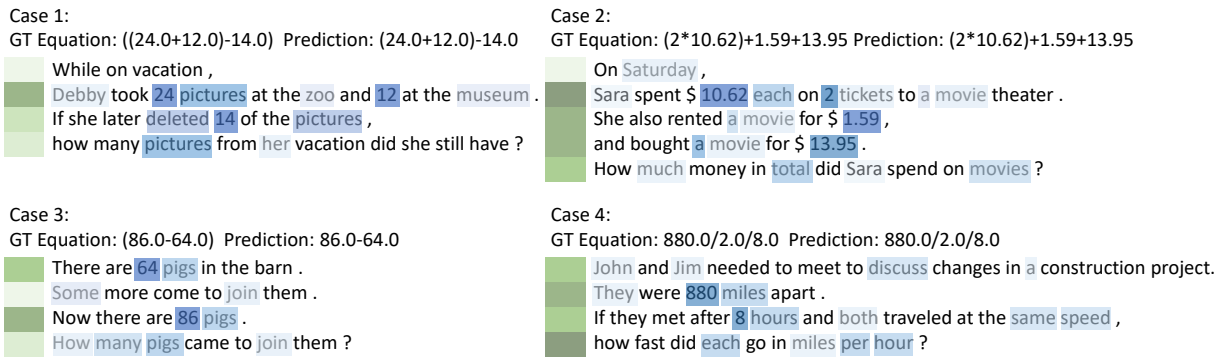


Figure 3: Visualizations of weighted connections of our word-level reasoning (blue) and sentence-level reasoning (green) in our hierarchical reasoning encoder. It provides insight into which words and sentences contribute to the final prediction, which can be of value in applications and interpretable analysis.

Pre-trained Knowledge Variants	Math23K (%)
Bert-base	78.9
Bert-large	80.0
Roberta-base	82.1
Roberta-large	83.9

Table 5: The effects of various pre-trained knowledge on math word problems.

baseline by 1.2% performance. Furthermore, after combing both of the reasoning processes, it can achieve 79.8% performance, which can validate the availability and superior ability of the hierarchical reasoning encoder. When it comes to the pre-trained knowledge encoder, our model can reach a significant improvement from 74.9% to 80.1%, which strongly supports the feasibility of using implicit knowledge from pre-trained neural network on math word problems. Furthermore, the ability of combination between word-level reasoning and pre-trained knowledge gets great scores of 81.4%. The sentence-level reasoning collaborated with the pre-trained knowledge encoder increases accuracy by 2.2% compared with purely using pre-trained knowledge encoder.

The effect of different pre-trained knowledge. As shown in Table 5, we explore the effect of language-based knowledge from different pre-trained transformer-based variants on the MWP task, which are BERT-base (Devlin et al., 2019b), BERT-large, Roberta-base (Liu et al., 2019c) and Roberta-large. We observe that more powerful pre-trained linguistic models can achieve better performance (78.9%→83.9%). One of the reasons for these gains comes from the commonsense and factual knowledge in the transformer-based models, which has been pre-trained on large-scale corpora to capture the implicit knowledge. These experimental results can also support the effectiveness of using outside knowledge to assist in the MWP task.

4.5 Case Study

In Table 4, we perform a case study on the solution expressions generated by GTS, Graph2Tree and our RPKHS. Previous methods wrongly predict the operator (e.g., GTS in 1^{st} example, Graph2Tree in 2^{nd} example.) and calculation order (e.g., Graph2Tree in 1^{st} example and GTS in 2^{nd} example.). For the last example, GTS and Graph2Tree predict wrong quantities (e.g., '2.12-2.6 on GTS, '2.6-1.8' on Graph2Tree.) while our RPKHS is able to handle this situation better than them. We believe it is because our model encodes the MWP in

richer representation by reasoning with pre-trained knowledge and hierarchical structure.

4.6 Visualizations

To validate that our model is able to select informative words and sentences in a problem description, we visualize the hierarchical attention weights in Figure 3 for four examples. Every line is a sentence (segment). Green denotes the sentence weight and blue denotes the word weight. Due to the hierarchical structure, we normalize the word weight by the sentence weight to make sure that only important words in important sentences are emphasized.

After looking through the four examples, we observe that our model can select the quantity words (positions) carrying strong contribution to the equation like *24*, *12* and *14* in the 1^{st} case, *64* and *86* in the 3^{rd} case. Besides, our model usually can accurately localize the relationship between the quantities and their semantics, such as *2 tickets* in 2^{nd} case and *880 miles* in the 4^{th} case.

Moreover, our model can deal with complex across-sentence contexts by building the correlation between different sentences. For instance, the 1^{st} sentence *John and Jim...* in the 4^{th} case seems to be unconsidered for solving the problem due to no quantity words inside it. However, our model figures out the 1^{st} sentence containing important quantity information when parsing the 4^{th} sentence (e.g., how fast did each...) via sentence-level reasoning. Through detailed visualized illustrations throughout the hierarchical reasoning process, we can reasonably interpret our results with concrete facts to show the effectiveness of our design.

5 Conclusion

We propose reasoning with pre-trained knowledge and hierarchical structure to jointly incorporate implicit knowledge and hierarchical representation into the neural network, which can be achieved by two encoders. A pre-trained knowledge encoder uses implicit knowledge for enhancing textual representation. A hierarchical reasoning encoder bridges the entity and context domain on MWP by building hierarchical reasoning between word-level and sentence-level reasoning. Extensive experiments show that the proposed model achieves a new state-of-the-art performance.

6 Acknowledgments

We thank all reviewers for providing the thoughtful and constructive suggestions. This work was supported in part by National Natural Science Foundation of China under Grant No.U1811461, in part by Natural Science Foundation of Guangdong Province, China under Grant No.2018B030312002, and in part by the Major Program of Guangdong Basic and Applied Research No.2019B030302002.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Yefim Bakman. 2007. Robust understanding of word problems with extraneous information. *arXiv: General Mathematics*.

Ting-Rui Chiang and Yun-Nung Chen. 2019. Semantically-aligned equation generation for solving and reasoning math word problems. 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 2656–2668. Association for Computational Linguistics.

J. Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019a. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019b. 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, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Geoffrey E. Hinton, Nitish Srivastava, A. Krizhevsky, Ilya Sutskever, and R. Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *ArXiv*, abs/1207.0580.

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 Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI-21*.

D. Huang, Shuming Shi, Chin-Yew Lin, and J. Yin. 2017. Learning fine-grained expressions to solve math word problems. In *EMNLP*.

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, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Zhengbao Jiang, F. F. Xu, J. Araki, and Graham Neubig. 2019. How can we know what language models know? *Transactions of the Association for Computational Linguistics*, 8:423–438.

Bugeun Kim, Kyung Seo Ki, Donggeon Lee, and G. Gweon. 2020. Point to the expression: Solving algebraic word problems using the expression-pointer transformer model. In *EMNLP*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

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: Human Language Technologies*, pages 1152–1157.

A. Kumar, Ozan Irsoy, Peter Ondruska, Mohit Iyyer, James Bradbury, Ishaan Gulrajani, Victor Zhong, Romain Paulus, and R. Socher. 2016. Ask me anything: Dynamic memory networks for natural language processing. In *ICML*.

Nate Kushman, Luke Zettlemoyer, R. Barzilay, and Yoav Artzi. 2014. Learning to automatically solve algebra word problems. In *ACL*.

Jierui Li, Lei Wang, Jipeng Zhang, Y. Wang, B. Dai, and D. Zhang. 2019. Modeling intra-relation in math word problems with different functional multi-head attentions. In *ACL*.

Qing Li, Siyuan Huang, Yining Hong, Y. Chen, Y. Wu, and S. Zhu. 2020a. Closed loop neural-symbolic learning via integrating neural perception, grammar parsing, and symbolic reasoning.

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

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*.

Wang Ling, Dani Yogatama, Chris Dyer, and P. Blunsom. 2017a. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In *ACL*.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017b. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 158–167. Association for Computational Linguistics.

Qianying Liu, Wenyv Guan, Sujian Li, and Daisuke Kawahara. 2019a. 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 (EMNLP-IJCNLP)*, pages 2370–2379, Hong Kong, China. Association for Computational Linguistics.

Qianying Liu, Wenyv Guan, Sujian Li, and Daisuke Kawahara. 2019b. Tree-structured decoding for solving math word problems. In *EMNLP/IJCNLP*.

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

A. Mitra and Chitta Baral. 2016. Learning to use formulas to solve simple arithmetic problems. In *ACL*.

Adam Paszke, S. Gross, Francisco Massa, A. Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Z. Lin, N. Gimelshein, L. Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*.

F. Petroni, Tim Rocktäschel, Patrick Lewis, A. Bakhtin, Yuxiang Wu, Alexander H. Miller, and S. Riedel. 2019. Language models as knowledge bases? In *EMNLP*.

M. Remy. 2002. Wikipedia: The free encyclopedia200214wikipedia: The free encyclopedia. 2001 – updated daily. gratis http://www.wikipedia.com.

Subhro Roy and D. Roth. 2015. Solving general arithmetic word problems. *ArXiv*, abs/1608.01413.

Subhro Roy and D. Roth. 2018. Mapping to declarative knowledge for word problem solving. *Transactions of the Association for Computational Linguistics*, 6:159–172.

Sunil Kumar Sahu, Fenia Christopoulou, Makoto Miwa, and S. Ananiadou. 2019. Inter-sentence relation extraction with document-level graph convolutional neural network. *ArXiv*, abs/1906.04684.

Shuming Shi, Y. Wang, Chin-Yew Lin, Xiaojiang Liu, and Y. Rui. 2015. Automatically solving number word problems by semantic parsing and reasoning. In *EMNLP*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

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

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

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

Lei Wang, Dongxiang Zhang, Jipeng Zhang, Xing Xu, Lianli Gao, Bing Tian Dai, and Heng Tao Shen. 2019b. Template-based math word problem solvers with recursive neural networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):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. Association for Computational Linguistics.

Y. Wu, M. Schuster, Z. Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, M. Krikun, Yuan Cao, Q. Gao, Klaus Macherey, J. Klingner, Apurva Shah, M. Johnson, X. Liu, Lukasz Kaiser, Stephan Gouws, Y. Kato, Taku Kudo, H. Kazawa, K. Stevens, George Kurian, Nishant Patil, W. Wang, C. Young, J. Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, G. Corrado, Macduff Hughes, and J. Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *ArXiv*, abs/1609.08144.

Zhipeng Xie and Shichao Sun. 2019a. 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.

Zhipeng Xie and Shichao Sun. 2019b. 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.

Ma Yuhui, Zhou Ying, Cui Guang-zuo, Ren Yun, and Huang Rong-huai. 2010. Frame-based calculus of solving arithmetic multi-step addition and subtraction word problems. *2010 Second International Workshop on Education Technology and Computer Science*, 2:476–479.

Jipeng Zhang, Lei Wang, R. Lee, Yi Bin, Yan Wang, J. Shao, and Ee-Peng Lim. 2020a. Graph-to-tree learning for solving math word problems. In *ACL*.

Jipeng Zhang, Lei Wang, Roy Ka-Wei Lee, Yi Bin, Yan Wang, Jie Shao, and Ee-Peng Lim. 2020b. 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.

Y. Zhu, Ryan Kiros, R. Zemel, R. Salakhutdinov, R. Urtasun, A. Torralba, and S. Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 19–27.

Yanyan Zou and Wei Lu. 2019. Quantity tagger: A latent-variable sequence labeling approach to solving addition-subtraction word problems. In *ACL*.