A Cognitive Solver with Autonomously Knowledge Learning for Reasoning Mathematical Answers

Jiayu Liu[1,2], Zhenya Huang[1,2,*], Xin Lin[1,2], Qi Liu[1,2], Jianhui Ma[1,2], Enhong Chen[1,2]
[1]Anhui Province Key Laboratory of Big Data Analysis and Application, School of Data Science & School of Computer Science and Technology, University of Science and Technology of China
[2]State Key Laboratory of Cognitive Intelligence
{jy251198, linx}@mail.ustc.edu.cn, {huangzhy, qiliuql, jianhui, cheneh}@ustc.edu.cn

2022 IEEE International Conference on Data Mining (ICDM) | 978-1-6654-5099-7/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICDM54844.2022.00037

***Abstract*—Reasoning answers to mathematical problems requires machines to think and operate like a human to learn knowledge from mathematical data, which is one of the fundamental tasks for exploring general artificial intelligence. Most solutions focus on mimicking how humans understand problems, which generate the necessary expressions for answers. However, they are still far from enough since they ignore the core ability of humans to acquire knowledge from experience. In this paper, we propose a Cognitive Solver (CogSolver) that is capable of autonomously learning knowledge from scratch to solve mathematical problems, inspired by two cognitive science theories. Specifically, we draw one insight from the dual process theory to establish an intelligent *BRAIN-ARM* framework, and refer to another information processing theory to summarize the knowledge learning process into *Store-Apply-Update* steps. In CogSolver, the *BRAIN* system *stores* three types of mathematical knowledge, including semantics knowledge, relation knowledge, and mathematic rule knowledge. Then, the *ARM* system *applies* the knowledge in *BRAIN* to answer the problems. Specifically, we design a knowledge-aware module and a commutative module in *ARM* to improve its reasoning ability, where the knowledge is organically integrated into answer reasoning process. After solving the problems, *BRAIN updates* the stored knowledge according to the feedback of *ARM*, where we develop knowledge filters to eliminate the redundant ones and further form a more reasonable knowledge base. Our CogSolver carries out the above three steps iteratively, which behaves more like a human. We conduct extensive experiments on real-world math word problem datasets. The experimental results demonstrate the improvement in answer reasoning and clearly show how CogSolver gains knowledge from the problems, leading to superior interpretability. Our codes are available at https://github.com/bigdata-ustc/CogSolver.**

***Index Terms*—knowledge learning, mathematical reasoning**

I. Introduction

Automatically solving mathematical problems is a crucial step towards general artificial intelligence. It requires machines to learn knowledge from data, understand mathematical logic, and conduct cognitive reasoning like a human [15]. Therefore, the ability to reason mathematical answers is viewed as a sign of the level AI achieves [39]. Among various types of problems, we specify math word problems (MWP) in this paper, which is an important branch and has attracted much attention since the 1960s [39]. Figure 1 shows a toy example. The MWP can be expressed as a short narrative that describes a problem and poses a question about an unknown quantity. The

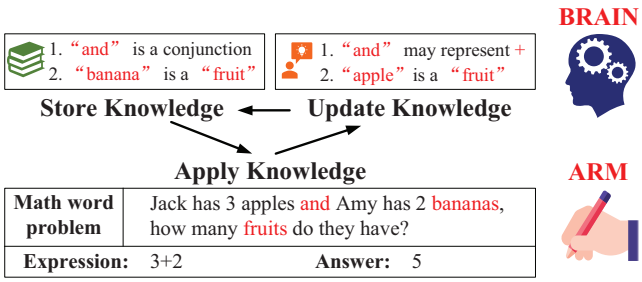


Fig. 1. Example of human learning process.

machine is required to understand the verbal description (Jack has 3...) which involves words (e.g., Jack, has) and quantities (i.e., $3, 2$), and reason the answer for the unknown quantity (i.e., 5). To solve this problem, one needs to understand and translate the problem sentence into an expression (e.g., $3+2$) composed of numbers (e.g., $3, 2$) and operators (e.g., $+$).

In the literature, existing efforts in solving MWP can be categorized into rule-based, statistic-based, semantics parsing-based, and deep learning-based [39]. Recently, inspired by language translation research, Seq2Seq method [36] has emerged, which inputs the problem sentence and outputs the expression as a sequence directly. Furthermore, advanced work, including Seq2Tree [38], Graph2Tree [40], and Seq2DAG [5], decodes the problem as a tree or a direct acyclic graph to improve the validity of expressions. Though previous work has achieved great success, there is still a certain gap with human-like intelligence manifested in two aspects. On one hand, humans can naturally learn knowledge from solving mathematical problems [14]. For example, we acquire the knowledge that "banana" (or "apple") is a kind of "fruit" from the problem in Figure 1. This knowledge is highly interpretable to humans and can be expressed explicitly. However, existing methods mainly focus on training models to improve the comprehension ability, including better understanding the semantic meaning and sentence structure in problems. Their learning results are often represented as neural networks, which lack the interpretability for the above relation knowledge. On the other hand, humans can apply the learned knowledge to answer unseen problems [33]. For example, the learned knowledge ("banana" is a kind of "fruit") is the basis for correctly reasoning the expression $2+4$ to another problem "John has 2 bananas and Lisa has 4 pears, how many fruits do they have?".

* Corresponding Author.

978-1-6654-5099-7/22/$31.00 ©2022 IEEE
DOI 10.1109/ICDM54844.2022.00037

Such ability is important to build a model with high generalization performance. In summary, we argue it is essential to empower machines with the ability to autonomously learn and apply knowledge as human cognitive processes do, which we specifically explore under MWP scenario in this paper.

To this end, we turn to some cognitive science theories for inspiration. We draw the first insight from dual process theory [10], [16], [22], which states that human cognitive processes require two systems: System 1 and System 2. System 1 retrieves relevant information in a fast and instructive way, while System 2 conducts deeper reasoning in an analytic and sequential manner. Therefore, we propose a novel Cognitive Solver (CogSolver) that contains *BRAIN-ARM* systems as the foundation of human cognition. They work in a similar way to System 1 and 2 respectively. We specify that *BRAIN* retrieves and provides knowledge related to a problem, based on which *ARM* reasons the answer step by step. Overall, *BRAIN* manages shared knowledge of all problems, while *ARM* solves a specific problem, both of which are necessary.

One step further, inspired by information processing theory [1], [6], [32], we operate the learning process of our CogSolver as a circular procedure consisting of three steps: *Store-Apply-Update* iteratively. To be more specific, in Figure 1, our CogSolver first *stores* the knowledge that "and" is a conjunction and "banana" is a kind of "fruit". Then it *applies* this knowledge to answer the problem "Jack has...". After figuring out this problem, it might *update* the knowledge by realizing that "and" may also represent "+" and "apple" is another kind of "fruit", which is useful for answering other problems. The above three steps are performed in the two systems respectively, where *BRAIN* controls the storage and update steps, and *ARM* conducts the application step.

However, it is non-trivial to carry out these three steps due to the following challenges. First, there are various types of knowledge in MWP, such as the semantics of tokens (words and operators), relationships between tokens (e.g., "apple" with "fruit", "add" with "+"), along with mathematical properties (e.g., commutative law). It is challenging to represent and store this different knowledge altogether. Second, we need to design targeted mechanisms to apply different types of knowledge, while also combining knowledge with contextual information when reasoning answers [13]. For example, "and" implies "+" in the problem "Jack has..." in Figure 1, while may represent as a conjunction in other sentences. Therefore, knowledge about the relationship between "and" and "+" should be applied differently under different problem contexts. Third, the mechanism of knowledge updating in the human brain remains underexplored at present. It is coupled with the manner of knowledge storage, which makes it more challenging to design suitable update methods.

To address these challenges, we implement CogSolver as follows. We first define and store three types of necessary mathematical knowledge in *BRAIN*, including *semantics knowledge*, *relation knowledge*, and *mathematic rule knowledge*. Then, when applying knowledge to solve a problem in *ARM*, we propose a knowledge-aware module to combine *semantics knowledge* and *relation knowledge* with contextual information, where a novel Hierarchical Graph Convolutional Network (HGCN) is designed to simulate the human reasoning process. Meanwhile, we propose a commutative module to apply *mathematic rule knowledge* to further enhance the reasoning ability. After solving the problem, *BRAIN* updates the *semantics knowledge* through back-propagation and *relation knowledge* with knowledge filters to reduce redundancy. The updated knowledge is further stored, and the cycle of *Store-Apply-Update* repeats in the subsequent learning process to gradually form a mathematical knowledge base in *BRAIN*.

We conduct extensive experiments on two real-world datasets. The experimental results show that our CogSolver can autonomously and effectively learn reasonable knowledge from MWP, and further utilize them to produce better answers with interpretable reasoning processes. To the best of our knowledge, this work is the first attempt that autonomously learns knowledge to solve MWP by constructing *BRAIN-ARM* systems and specifying the learning process into *Store-Apply-Update* steps, whose main ideas are quite general and can be potentially applicable to other mathematical problems.

II. Related Work

In this section, we summarize the related work as follows.

Math Word Problems. In the literature, existing MWP solvers can be classified into rule-based, statistic-based, semantics parsing-based, and deep learning-based [39]. Specifically, rule-based methods rely on manually crafted schemas and pattern matching [2], while statistic-based methods leverage traditional machine learning models like SVM to select and complete pre-defined templates [27]. Semantic parsing-based methods focus on the semantic structure of the textual sentences and derive math expressions after constructing them into logic forms [31]. However, these methods require handcrafted templates, rules, representation language, and so on, which limits their applications on large-scale datasets and causes low generality. Due to the advantage of deep learning in feature extraction, Wang et al. [36] first used a recurrent neural network to translate math word problems into equation templates. Then, approaches including Seq2Tree [23], [37], [38], Graph2Tree [40], Seq2DAG [5], and reinforcement learning [12] have been developed. For example, Xie et al. [38] proposed a top-down goal decomposition process to generate expression trees, motivated by the goal-driven mechanism in human problem solving. Zhang et al. [40] constructed graphs to enrich quantities' representations with the relationships and order information. Wu et al. [37] incorporated external knowledge and built an entity graph for each problem to obtain better problem understanding. Besides, pre-trained language models such as BERT [21], RoBERTa [17], BART [30] have also been utilized to promote comprehension of problems. Apart from arithmetic problems with only one variable, other difficult mathematical problems like equation set problems [5], geometric word problems [28], and new conjectures discovery [7] also attract great attention.

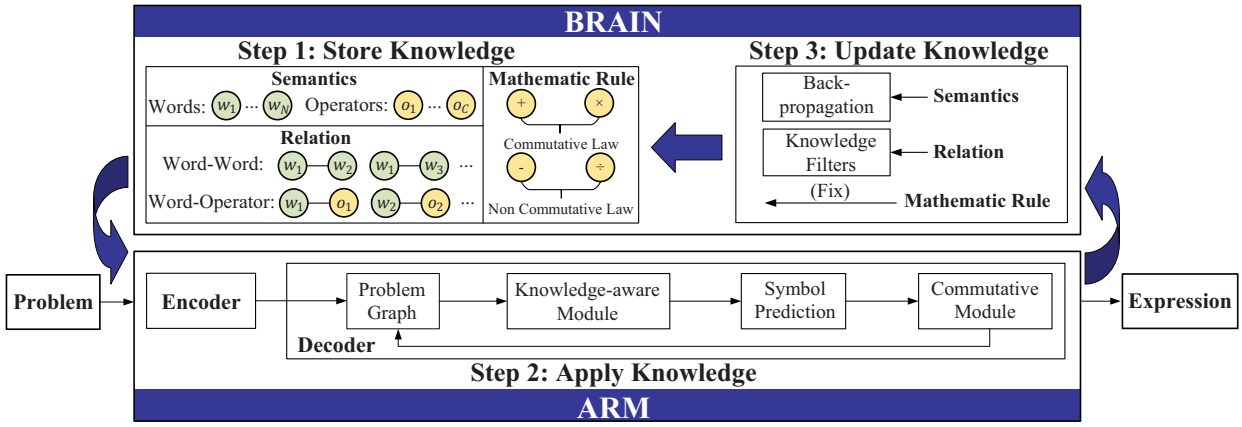


Fig. 2. Framework of CogSolver. The top-level *BRAIN* stores and updates knowledge. The bottom-level *ARM* applies knowledge.

Cognitive Science Theories. Cognitive science investigates the human brain by bringing studies in psychology, linguistics, anthropology, philosophy, and so on [3]. Here, we introduce two theories about the human cognitive structure and the reasoning process related to our work. First, *dual process theory* [10], [16], [22] points out that two separate cognitive systems are underlying human thinking and reasoning process. System 1 includes instinctive behaviors that are innately programmed and retrieves information in a fast, unconscious, and implicit way. System 2 conducts a slow, conscious, and explicit sequential thinking process. Based on this, previous attempts have been made for some machine learning tasks, such as question answering [8] and knowledge graph reasoning [9]. In our CogSolver, *BRAIN* resembles System 1 due to its role as a guider and knowledge provider, while *ARM* operates similarly to System 2 since it produces the answers. Second, *information processing theory* [1], [6], [32] states that the mind's machinery includes bringing information in, manipulating information, and holding information, which are conducted in sensory memory, short-term memory, and long-term memory respectively. Further, it defines cognitive processes as mental activities that help information transfer from one memory to another, such as attention, perception, repetition, coding, and retrieving [6]. As we focus on how machines gain knowledge from experience, mechanisms associated with long-term memory should be emphasized. We summarize the related activities into three steps. First, the ability to hold information refers to knowledge storage. Second, repetition and coding are activities that transfer information from short-term memory to long-term memory, which means summarizing the commonly used methods or steps in MWP as knowledge and is thus relevant to knowledge update. Third, retrieving information from long-term to short-term is taking advantage of what already know, coincident with the process of knowledge application for answer reasoning.

Our work differs from previous studies as follows. First, existing approaches mainly focus on training models to get a better understanding of problems, while our work investigates how humans learn knowledge and designs corresponding reasoning mechanisms to apply learned knowledge for improving answer generation results. Second, the recent method [37] tries to utilize external knowledge bases to solve MWP, and thus is limited when manually constructed knowledge is unavailable. Comparatively, our work can autonomously learn knowledge from scratch, which is more in line with the goal of general artificial intelligence and achieves better flexibility. Last but not least, we construct a superior framework of human cognition by specifying three steps in the learning process summarized from information process theory and unifying them with intelligent systems stated in dual process theory.

III. Cognitive Solver: Two Systems

In this section, we first give formal definition of MWP task. Then we introduce the *BRAIN-ARM* systems of our CogSolver.

A. Problem Definition

The input of a math word problem P is a sequence of n word tokens and numeric values: $P = \{p_1, ..., p_n\}$ where p_i is either a word token (e.g., "banana") or a numeric value (e.g., "3"). We define the numeric values in P as N_P.

The output of P is a numeric answer s_P derived from an expression $E_P = \{y_1, ..., y_m\}$ that is a sequence of m symbols. Each symbol y_i comes from the decoding target vocabulary V_P. V_P is composed of the operator set V_O (e.g., $\{+, \times, -, \div\}$), numeric constant set V_C (e.g., $1, 2, \pi$), and N_P, i.e., $V_P = V_O \cup V_C \cup N_P$. Note that different problems may have different V_P since N_P varies with the input.

The goal of MWP is to train a model that reads the input problem P and calculates the numeric answer s_P based on a generated valid mathematical expression E_P.

B. BRAIN-ARM Systems

We establish the CogSolver with two systems: *BRAIN-ARM* as the foundation as shown in Figure 2, referring to the idea of dual process theory [10], [16], [22] to model human cognition. Specifically, the top-level system is *BRAIN* which guides the bottom-level *ARM* to solve a problem. They work similarly to System 1 and 2 in the theory respectively.

The autonomously learning process for MWP is as follows. Given a problem, *BRAIN* first retrieves *stored* knowledge to *ARM*, and then *ARM applies* the knowledge and conducts

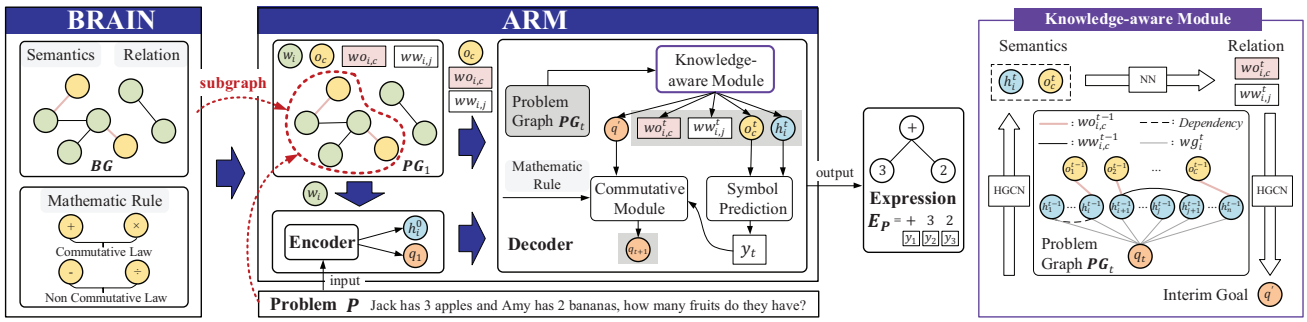


Fig. 3. Knowledge application. Left part shows the solving process. Right part illustrates details of knowledge-aware module.

cognitive reasoning to figure out the problem (i.e., generate the expression). Finally, *BRAIN updates* the knowledge according to the feedback of *ARM* for future applications.

IV. Cognitive Solver: three steps

We draw insights from information processing theory [1], [6], [32] and carry out the knowledge learning process of CogSolver in three steps: *Store-Apply-Update* iteratively. They are operated in *BRAIN-ARM* systems respectively. In this section, we describe the implementation details of these steps.

A. Knowledge Storage

In CogSolver, *BRAIN* stores three types of knowledge for MWP as shown in Figure 2, including *semantics knowledge*, *relation knowledge*, and *mathematic rule knowledge*. They are all explicit and interpretable.

Semantics knowledge refers to the meanings of mathematical tokens (words and operators) and can be represented as feature vectors in the conceptual space [22]. We denote the semantic vectors of tokens, including words as $W = \{w_1, ..., w_N, w_i \in R^d\}$, and operators as $O = \{o_1, ..., o_C, o_c \in R^d\}$, where N, C are the number of words and operators in the field of MWP, and d is the dimension.

Relation knowledge describes the relationships between tokens, which is further divided into two types. The first is word-word relation or the so-called common-sense [4]. Note that in this paper, we focus more on the types of knowledge instead of the exact meaning of such relationship (e.g., hypernymy, antonymy) [29]. Therefore, we adopt a real value $ww_{i,j} \in [0, 1]$ to store the overall relation strength between words i and j instead of a binary one because one word can have multiple meanings, not all of which have the relationship [24]. For instance, "apple" belongs to "fruit" when it is considered as food, while does not when it represents the corporation. The second is word-operator relation (e.g., "add" is highly related to "+"). Similarly, we denote the relation strength between word i and operator c as $wo_{i,c} \in [0, 1]$.

Mathematic rule knowledge indicates mathematical rule. In this work, we consider the commutative law for simplicity. Commutative law is a defined property of operators. We set the rule for machines that "+" and "×" satisfy the commutative law, while "−" and "÷" do not. This setup is similar to a teacher introducing commutative law directly to students, which can avoid confusion of basic mathematical concepts.

Specifically, the *semantics knowledge* and *relation knowledge* together constitute a knowledge graph in *BRAIN*, denoted as BG, whose node representations and edge strength are initialized randomly and will be updated as illustrated in section IV-C to achieve learning from scratch autonomously.

B. Knowledge Application

The process of solving a specific problem is completed by the application step conducted in *ARM* as shown in Figure 3. It follows the encoder-decoder manner, where we propose a novel knowledge-aware module and a novel commutative module in decoder to enhance *ARM*'s ability to reason answers based on knowledge retrieved from *BRAIN*. Generally, given a problem $P = \{p_1, ..., p_n\}$, we first retrieve relevant *semantics knowledge* into *ARM*'s encoder. Then we combine its output with *relation knowledge* to construct a problem-specific graph PG_t in *ARM*'s decoder. After that, we conduct cognitive reasoning on PG_t $(t = 1, ..., m)$ in the knowledge-aware module to predict the expression $E_P = \{y_1, ..., y_m\}$ (represented as tree) symbol by symbol, during which the *mathematic rule knowledge* is applied in the commutative module to improve the reasoning ability implicitly.

1) Encoder: ARM's encoder reads problem P and produces hidden representations of words $H = (h_1^0, ..., h_n^0)$ and the initial goal q_1 of the decoder for reasoning. Specifically, we initialize word p_i in P with *semantics knowledge* w_i from *BRAIN* and feed $\{w_1, ..., w_n\}$ into a neural network f_θ as (1). Note that f_θ can be implemented ranging from LSTM, BERT, to specific MWP encoders (e.g., HMS [23]). We do not emphasize their differences and adopt the hierarchical encoder in HMS [23] as it simulates human reading habits.

$$H \in \mathbb{R}^{n \times d_1}, q_1 \in \mathbb{R}^{d_1} = f_\theta(w_1, ..., w_n). \quad (1)$$

2) Decoder: ARM's decoder takes the general goal-driven mechanism [38] to reason the expression E_P, which generates one token y_t at time t based on problem P and all the previous outputs $\{y_1, ..., y_{t-1}\}$. Generally, given problem P and target expression E_P in the training set, the prediction loss is:

$$Loss_P = \sum_{t=1}^{m} -\log \mathbf{P}(y_t \mid y_1, ..., y_{t-1}, P). \quad (2)$$

As shown in Figure 3, our decoder consists of four main parts: **Problem Graph**, **Knowledge-aware Module**, **Symbol Prediction**, and **Commutative Module**. Generating each y_t requires a process conducted in these four parts. Specifically, the

problem graph PG_t first gets existing *semantics knowledge*, *relation knowledge*, and goal q_t. Then the knowledge-aware module computes context-related semantics and relation in PG_t, and derives an interim goal q'. Based on the semantics, a pointer-generator network [12] is used to predict y_t. Then, the next goal q_{t+1} is generated with q' by the commutative module which applies *mathematic rule knowledge*. Consequently, the problem graph PG_t evolves into PG_{t+1}, which is used to conduct the next reasoning step for generating y_{t+1}.

In particular, we introduce commutative loss $Loss_C$ in the commutative module to implicitly ensure that the reasoning results conform to the *mathematic rule knowledge* (i.e., commutative law). Overall, the loss to be minimized is:

$$Loss = Loss_P + \lambda Loss_C. \tag{3}$$

where λ is a hyper-parameter that keeps a balance between symbol prediction loss and commutative loss.

3) Implementation of decoder: In the following, we describe the details of the four parts in our *ARM*'s decoder.

Problem Graph. Given problem P, we maintain a problem graph $PG_t = (V, E)$ to organize relevant *semantics knowledge* and *relation knowledge*, which will evolve into PG_{t+1} after completing current reasoning step at time t that generates y_t. V is a node set containing words $\{h_i^{t-1}, i = 1, ..., n\}$ from P, all operators $\{o_c^{t-1}, c = 1, ..., C\}$ and current goal q_t. E is a set of undirected weighted edges between words, operators and goal. We denote the edges in E between words and words, words and operators as $ww_{i,j}^{t-1}, wo_{i,c}^{t-1}$, respectively. We also build dependency edges in E between words in a clause to capture the semantic structure of problem P by *stanford corenlp toolkit* [25]. Besides, the edge $wg_i^t \in E$ between goal q_t and word i that describes how focused q_t is on i is calculated by the hierarchical attention mechanism [23].

In summary, PG_t is composed of three types of nodes and four types of edges. The superscript $t-1$ indicates that $h_i^{t-1}, o_c^{t-1}, ww_{i,j}^{t-1}, wo_{i,c}^{t-1}$ are knowledge prior to time step t. To be notice, at the first step $t = 1$, $o_c^0, ww_{i,j}^0, wo_{i,c}^0$ in PG_1 are initialized with *semantics knowledge* o_c and *relation knowledge* $ww_{i,j}, wo_{i,c}$ retrieved from *BRAIN* respectively. Thus, PG_t can be seen as a subgraph of the overall BG stored in *BRAIN* that is fine-tuned for current reasoning step.

Knowledge-aware Module. Directly applying knowledge including semantics h_i^{t-1}, o_c^{t-1} and relation $ww_{i,j}^{t-1}, wo_{i,c}^{t-1}$ in PG_t may be unsuitable to current reasoning step (recall the example of "and" in different contexts). Thus, this module aims at integrating this knowledge with contextual information implied in current goal q_t for subsequent symbol prediction. It processes semantics, relation and derives the interim goal successively, illustrated in (K1.), (K2.) and (K3.) respectively.

(K1.) For semantics in PG_t, we propose a hierarchical graph convolutional network (HGCN) to propagate the information from current goal q_t to the upper levels of PG_t to simulate the human reasoning habits. For example, to solve the goal q_t representing "How many fruits do they have", humans first notice the word "and" in the problem (Figure 3), and then get the correct symbol by associating it with "+". Therefore, information in q_t should be first propagated to words, and then to operators, following a hierarchical manner.

Specifically, HGCN first updates the word semantics h_i^t by:

$$\begin{aligned} AGG_i &= [wg_i^t \cdot q_t, \sum\nolimits_{j=1}^{n} ww_{i,j}^{t-1} \cdot h_j^{t-1}, \sum\nolimits_{j \in \mathcal{N}_d(i)} h_j^{t-1}], \\ h_i' &= ReLU(\mathbf{W}_u \cdot AGG_i + b_u), \\ f_i &= \sigma(\mathbf{W}_{fw} \cdot [h_i^{t-1}, AGG_i] + b_{fw}), \\ h_i^t &= f_i \cdot h_i^{t-1} + (1 - f_i) \cdot h_i', \end{aligned} \tag{4}$$

where σ is the sigmoid function, $\mathbf{W}_u, \mathbf{W}_{fw}, b_u, b_{fw}$ are learnable parameters. AGG_i aggregates word i's neighbors in different relationships with concatenation $[\cdot]$ ($\mathcal{N}_d(i)$ are neighbors in dependency edges), except its upper-level neighbors (operators). The forget gate f_i controls the extent to which existing knowledge is forgotten. Then the context-related operator semantics o_c^t is calculated similarly by:

$$\begin{aligned} o_c' &= ReLU(\mathbf{W}_o \cdot \sum\nolimits_{i=1}^{n} wo_{i,c}^{t-1} \cdot h_i^t + b_o), \\ f_c &= \sigma(\mathbf{W}_{fo} \cdot [o_c^{t-1}, \sum\nolimits_{i=1}^{n} wo_{i,c}^{t-1} \cdot h_i^t] + b_{fo}), \\ o_c^t &= f_c \cdot o_c^{t-1} + (1 - f_c) \cdot o_c'. \end{aligned} \tag{5}$$

(K2.)[1] For relation in PG_t, we consider common-sense remains unchanged when solving a problem (e.g., "banana" always belongs to "fruit" in a problem), i.e., $ww_{i,j}^t = ww_{i,j}^{t-1}$, and only adjust the word-operator relation $wo_{i,c}^t$ by:

$$\begin{aligned} wo_{i,c}' &= softmax(ReLU(\mathbf{W}_e \cdot [h_i^t, o_c^t, wo_{i,c}^{t-1}] + b_e)), \\ f_{i,c} &= \sigma(\mathbf{W}_{fe} \cdot [h_i^t, o_c^t] + b_{fe}), \\ wo_{i,c}^t &= f_{i,c} \cdot wo_{i,c}^{t-1} + (1 - f_{i,c}) \cdot wo_{i,c}'. \end{aligned} \tag{6}$$

(K3.) To derive an interim goal q' with the knowledge of words and operators that helps the generation of the next goal q_{t+1} in the below commutative module, we propagate down from the top of PG_t to compute q' by another HGCN:

$$\begin{aligned} AGG_i &= [\sum\nolimits_{c=1}^{C} wo_{i,c}^t \cdot o_c^t, \sum\nolimits_{j=1}^{n} ww_{i,j}^t \cdot h_j^t, \sum\nolimits_{j \in \mathcal{N}_d(i)} h_j^t], \\ h_i' &= ReLU(\mathbf{W}_u' \cdot AGG_i + b_u'), \\ q_a &= \sum\nolimits_{i=1}^{n} wg_i^t \cdot h_i', \\ f &= \sigma(\mathbf{W}_{fg} \cdot [q_a, q_t] + b_{fg}), \\ q' &= f \cdot q_t + (1 - f) \cdot ReLU(\mathbf{W}_g \cdot q_a + b_g), \end{aligned} \tag{7}$$

Symbol Prediction. The obtained $\{h_i^t, i = 1, ..., n\}$, $\{o_c^t, c = 1, ..., C\}$ have integrated existing knowledge with contextual information, and thus can better achieve current goal q_t and reason y_t. Therefore, we input them into a pointer-generator network [12] to generate y_t at time t since y_t needs to be inferred from the external vocabulary (i.e., $V_O \cup V_C$) or derived from the problem itself (i.e., N_P). In brief, we first predict the probability of y_t belonging to the external vocabulary:

$$\mathbf{P}_{gen} = \sigma(\mathbf{W}_{gen} \cdot [q_t, c_t] + b_{gen}), \tag{8}$$

[1]Dependency edges represent the semantic structure of P, and thus remain unchanged at different reasoning steps. Edges wg_i^t are recomputed by the hierarchical attention mechanism [23] at the beginning of each step t.

where c_t is a context vector fusing word-clause levels semantics [23]. Then we calculate the distribution of y_t on the external vocabulary (denoted as $\mathbf{P}_g(y_t)$) and N_P (denoted as $\mathbf{P}_p(y_t)$) by feeding $\{o_c^t\}$ and $\{h_i^t\}$ into different networks respectively. In sum, the probability $\mathbf{P}(y_t \mid y_1, ..., y_{t-1}, P)$ is:

$$\mathbf{P}(y_t) = \begin{cases} (1 - \mathbf{P}_{gen}) \cdot \mathbf{P}_p(y_t) & \text{if } y_t \in N_P, \\ \mathbf{P}_{gen} \cdot \mathbf{P}_g(y_t) & \text{if } y_t \in V_O \cup V_C. \end{cases} \tag{9}$$

Commutative Module. After reasoning symbol y_t, our CogSolver needs to generate the next goal q_{t+1} to support the next reasoning step at time $t+1$. Specifically, if y_t is an operator, the commutative module will first decompose the interim goal q' from (7) into a left sub-goal q^l (i.e., q_{t+1}) and infer the left child tree t^l. Then, the right sub-goal q^r of q' will be generated using q' and t^l. The left and right sub-goal q^l, q^r are calculated by networks proposed in [38] and we summarize them briefly as $q^l = Decompose1(q')$ and $q^r = Decompose2(q', t^l)$.

To further improve the goal decomposition results, we notice that when the commutative law is satisfied, it is equivalent to generate the left sub-goal first (e.g., 3 in Figure 3) and the right sub-goal first (e.g., 2). Thus, we apply *mathematic rule knowledge* by inversely generating an extra left sub-goal q^l_{inv} using the right child tree t_r. If y is $+$ or $\times$, q^l_{inv} should represent the same meaning as q^l. In this case, we define the commutative loss $Loss_C$ as their distance in (10). In other cases (e.g., y is $-$ or $\div$), we do not compute $Loss_C$.

$$Loss_C = || \underbrace{Decompose2(q', t^r)}_{q^l_{inv}} - \underbrace{Decompose1(q')}_{q^l} ||. \tag{10}$$

C. Knowledge Update

Note that the knowledge in *BRAIN* may contain a lot of redundancy. For example, after initialization (IV-A), the edge strength $wo_{i,c}$ between word "the" and operator "+" in BG may be very strong, which is unreasonable. Thus, we need to simulate in CogSolver how humans update knowledge after answering problems while ensuring knowledge rationality.

Specifically, note that *Semantics knowledge* (i.e., $\{w_i, o_c\}$ in BG) is naturally updated by back-propagation when minimizing (3), and *Mathematic rule knowledge* (commutative law) is fixed. Thus, here we focus on updating *relation knowledge*, i.e., relation strength $ww_{i,j}, wo_{i,c}$ in BG.

As for words i, j, we imply their word-word relation $ww_{i,j}$ by the distance between semantic vectors as follows:

$$ww_{i,j} = \sigma(-||w_i - w_j|| + Mean_{dis}), \tag{11}$$

where $Mean_{dis}$ is the average distance between all pairs of words and is used to widen the distribution of $ww_{i,j}$. To reduce redundancy, we further introduce a knowledge filter with threshold δ_1 to eliminate relationships with weak strength:

$$ww_{i,j} = \begin{cases} ww_{i,j} & \text{if } ww_{i,j} > \delta_1, \\ 0 & \text{else.} \end{cases} \tag{12}$$

For $wo_{i,c}$, we calculate the word-operator relation between each word and all operators using softmax function:

$$wo_{i,c} = softmax(-||w_i - o_c||). \tag{13}$$

TABLE I
THE STATISTICS OF DATASETS.

Dataset	Math23K	MAWPS
Num. Problems	23,162	2,373
Num. Operators	5	4
Avg. problem length	28.02	30.08

Moreover, if a word does not associate with any operator (e.g., "apple" is not related to $\{+, \times, -, \div\}$), its weights with different operators could be thought almost equal (e.g., $wo_{i,c} \approx 0.25$). Thus, we also include another knowledge filter with threshold δ_2 to filter out $wo_{i,c}$ by:

$$wo_{i,c} = \begin{cases} wo_{i,c} & \text{if } wo_{i,c} > \delta_2, \\ 0 & \text{else.} \end{cases} \tag{14}$$

It should be noticed that the sum of the relationships between a word and all operators is not 1. For example, the updated relationship between "apple" and each operator may equal 0, meaning that "apple" is unrelated to any operator, and thus will not directly contribute to predicting the probability of an operator by $wo_{i,c}$, which is reasonable in reality.

In summary, before learning, the knowledge in *BRAIN* is initialized randomly without rational meanings. However, with the *Store-Apply-Update* steps (autonomously learning) being iteratively carried out when solving problems, it is constantly improved and can finally form a long-term mathematical knowledge base. We will demonstrate it in section V-E. Meanwhile, as the knowledge becomes more accurate, the effect of *ARM* in answering problems will also improve.

V. EXPERIMENTS

A. Dataset Description

- **Math23K:** Math23K [36] is a well-known benchmark dataset that contains 23,162 Chinese math word problems. Wu et al. [37] have also published an external common-sense knowledge graph for Math23K.
- **MAWPS:** MAWPS [19] is a dataset that contains problems with one or more unknown variables. We select 2,373 problems with only one unknown variable.

Table I summarizes the basic statistics of two datasets, and Figure 4 shows the distribution of operators occurring in target expressions. We can find that except for $\wedge$ (power), $\{+, \times, -, \div\}$ are distributed more evenly in the Math23K, while $+$ accounts for nearly 40% in MAWPS. It will bring the challenge of learning knowledge related to these operators.

B. Experimental Setup

Implementation Details. For knowledge storage in section IV-A, the dimension d of semantic vectors in *BRAIN* is 128. Words' vectors are initialized with pre-trained word2vec [26] learned from the training set, while operators' vectors are initialized randomly. Relation knowledge is initialized by (11)-(14). For knowledge application in section IV-B, the dimension d_1 of hidden vectors is 512, and other parameters in *ARM* are initialized with Kaiming initialization [11]. λ in (3) is set to 0.001. For knowledge update in section IV-C, the filter thresholds δ_1, δ_2 in (12), (14) are set as $0.7, 0.3$ respectively,

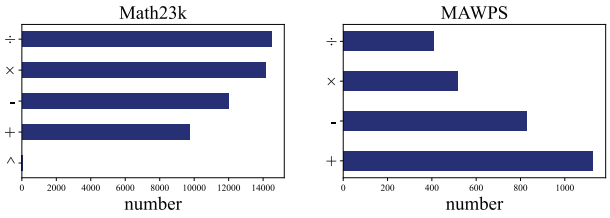


Fig. 4. The distribution of operators.

which are important in *BRAIN* as further discussed in section V-E. The number of iterations for *Store-Apply-Update* steps (which we call $learning\ iteration$) is set to 120. During each $iteration$, parameters are also trained in the knowledge update step for one epoch with mini-batch 64 and dropout probability 0.5. The learning rate of Adam optimizer [18] is initialized with 0.001 and will be halved every 20 $learning\ iterations$.

For dataset preprocessing, words with less than 5 occurrences are converted to a special token "UNK". For Math23K, we follow its original published partition. For MAWPS, the models are evaluated with 5-fold cross-validation. All experiments are run on a Linux server with four 2.30GHz Intel Xeon Gold 5218 CPUs and a Tesla V100 GPU.

Evaluation metric. We use answer accuracy as the evaluation metric because there may be multiple correct and equivalent expressions to the same problem. If the calculated value equals the answer, the predicted expression is thought of correct.

C. Baseline Approaches

We select the following representative and state-of-the-art methods as baselines. For a fair comparison, we do not include methods that are based on pre-trained language models.

- **DNS** [36]: uses a vanilla seq2seq model to translate MWP to equation templates directly.
- **Math-EN** [34]: utilizes an equation normalization method to address the problem of duplicated equations.
- **T-RNN** [35]: applies recursive neural networks to infer unknown operator nodes in the predicted template.
- **GROUP-ATT** [20]: extracts multiple features in MWP by a group attention mechanism.
- **GTS** [38]: generates expression trees by a tree-structured neural network in a goal-driven manner.
- **Graph2Tree** [40]: captures the relationships and order information among quantities by a graph-based encoder.
- **KA-S2T** [37]: incorporates the manually constructed external knowledge to obtain better problem representation.
- **HMS** [23]: makes use of the hierarchical word-clause-problem relation to better exploit the problem semantics.

D. Experimental Results

1) Answer Performance: Table II reports the answer accuracy of all models[2], and we find several key observations. First, our CogSolver outperforms all the baselines, and by applying paired t-test, its improvements over the SOTA Graph2Tree are statistically significant with $p \leq 0.01$ on both datasets (marked with "$*$"). It demonstrates that CogSolver benefits from applying learned knowledge to reason more accurate answers and verifies the effectiveness of the knowledge learning process that we construct by *BRAIN-ARM* systems and *Store-Apply-Update* steps. Second, the improvements of CogSolver over HMS highlight the advantage of the knowledge-aware module and commutative module in enhancing the reasoning ability of *ARM*'s decoder (recall CogSolver uses the same encoder as HMS). Third, CogSolver performs better than KA-S2T which incorporates external common-sense. It reflects the ability of our CogSolver to learn knowledge from scratch and form a reliable knowledge base autonomously, which has better flexibility. Moreover, it shows the potential of *BRAIN* to achieve the effect of manually constructed knowledge bases, which is important for building a general AI. Last, the accuracy of CogSolver on MAWPS is higher than Math23K. That is probably because "+" takes a larger proportion on MAWPS (Figure 4). Thus, it is easier for CogSolver to learn and apply the relation knowledge between "+" with words for reasoning, suggesting that training data affects knowledge learning results greatly, which can be further witnessed in section V-E2.

TABLE II
ANSWER ACCURACY ($* : p \leq 0.01$).

	Math23K	MAWPS
DNS	0.581	0.595
Math-EN	0.667	0.692
T-RNN	0.669	0.668
GROUP-ATT	0.695	0.761
GTS	0.743	0.786
KA-S2T	0.763	/[3]
HMS	0.755	0.804
Graph2Tree	0.764	0.820
CogSolver	**0.773***	**0.829***

TABLE III
ABLATION STUDY ON REDUCING *Store-Apply-Update* STEPS.

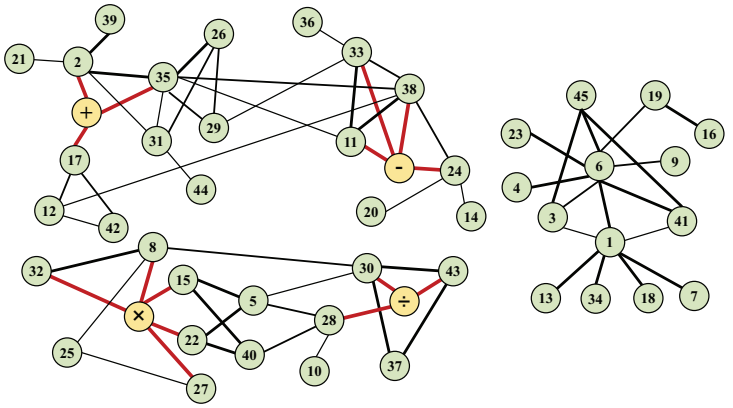
Model			Math23K	MAWPS
CogSolver			**0.773**	**0.829**
Store	w/o Relation		0.754	0.809
	w/o Mathematic Rule		0.766	0.820
Apply	w/o Knowledge-aware Module	Semantics	0.758	0.820
		Relation	0.761	0.818
Update	w/o update Relation		0.741	0.796

2) Ablation Study: To verify the effectiveness of the three steps in CogSolver, we conduct the ablation study. Table III reports the results of each case. Specifically, for knowledge storage, "w/o Relation" omits the *relation knowledge* in *BRAIN*, thus without problem graph and knowledge-aware module, while "w/o Mathematic Rule" ignores $Loss_C$ in the commutative module during training. For knowledge application, "w/o Knowledge-aware Semantics" and "w/o Knowledge-aware Relation" do not calculate context-related semantics by (4)(5) and relationships by (6) in the knowledge-aware module, respectively. For knowledge update, "w/o update Relation" fixes the *relation knowledge* after initialization.

We conclude the results as follows. First, the accuracy of CogSolver degrades when any component is missing, showing the necessity of our designed components for answer reason-

[2]Due to different Pytorch versions, for a fair comparison, we rerun the Graph2Tree and HMS with Pytorch version 1.8.1.

[3]KA-S2T [37] does not provide knowledge base and conduct experiment on MAWPS.



1 fruit	16 clothes	31 earn
2 total	17 remain	32 multiplier
3 pepper	18 pear	33 subtractor
4 dessert	19 goods	34 peach
5 everyone	20 lighten	35 add
6 food	21 bring	36 decimals
7 orange	22 per	37 quotient
8 multiply	23 sesame	38 minus
9 cream	24 lost	39 whole
10 half	25 ratio	40 everyday
11 remainder	26 improve	41 muskmelon
12 others	27 times	42 save
13 banana	28 average	43 divider
14 break	29 exceed	44 include
15 each	30 divide	45 vegetable

Fig. 5. Part of BG in *BRAIN* of CogSolver after autonomously learning on Math23K (Result on MAWPS is similar and we omit it due to space limit). The green and yellow nodes represent words and operators respectively. The black and red line represent word-word relation $ww_{i,j}$ and word-operator relation $wo_{i,c}$ respectively. The width of line reflects the relation strength, i.e., the thicker the line, the greater the strength of the relation.

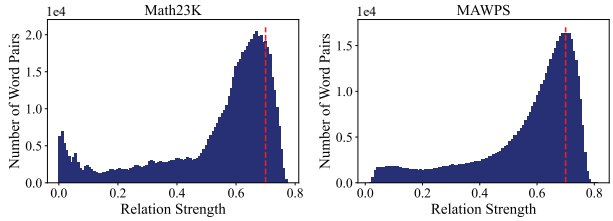


Fig. 6. Word-word relation $ww_{i,j}$ (distribution). The red dash line represents the threshold $\delta_1 = 0.7$ in (12).

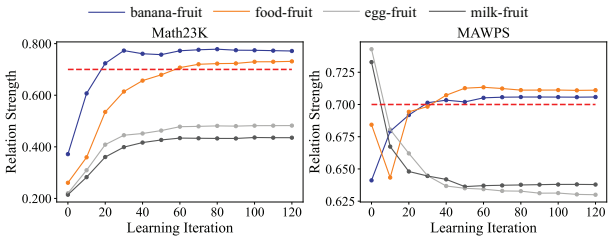


Fig. 7. Word-word relation $ww_{i,j}$ (examples). The red dash line represents the threshold $\delta_1 = 0.7$ in (12).

ing. Second, the performance suffers the greatest damage when the knowledge is not updated (i.e., "w/o update Relation"). It indicates that knowledge update plays the most crucial role in CogSolver, which can eliminate wrong knowledge that harms the reasoning process. Third, compared with "w/o Mathematic Rule", "w/o Relation" diminishes the result more greatly, suggesting *relation knowledge* is more effective than commutative law for correctly conducting reasoning. Last, context-related semantics and relation are almost equally important as there is no significant difference between "w/o Knowledge-aware Semantics" and "w/o Knowledge-aware Relation".

E. Analysis of BRAIN

Part of BG in *BRAIN* after autonomously learning from scratch is shown in Figure 5. We observe that CogSolver has formed reasonable mathematical knowledge bases that contain word-word relation and word-operator relation on both datasets. For example, on Math23K, it learns that "+" is related to word "add" and "add" is related to "exceed", while '+" is not related to "multiply". The line width reflects the relation strength (e.g., the relationship between "add" and "total" is stronger than "add" and "earn"). In this section, we deeply analyze these two types of knowledge to demonstrate the effectiveness and interpretability of CogSolver.

1) Word-word relation: We compute the relation strength $ww_{i,j}$ for any two words after learning by (11) and visualize the distribution for all word pairs in Figure 6. As we can see, the strength of most relationships is distributed around 0.6-0.7 on both datasets. With threshold $\delta_1 = 0.7$ in (12), only 15% and 20% of word pairs (right of the red dash line) are remained (have common-sense knowledge) on Math23K and MAWPS respectively. Furthermore, for Math23K, compared with the external knowledge source published by KA-S2T [37] (not provided for MAWPS), approximately 77.2% of true common-sense knowledge has been automatically acquired by CogSolver, which further proves its ability to learn this knowledge correctly while maintaining superior flexibility.

Moreover, we take several pairs of words as examples and show their relation strength during learning in Figure 7. As we know, there are real relationships between "banana" and"fruit", "food" and "fruit", while there are false between "egg" and "fruit", "milk" and "fruit". For real relationships, we observe they gradually strengthen and finally exceed the threshold 0.7. For false relationships, although initialized differently on Math23K (≈ 0.20) and MAWPS (≈ 0.74), their strength is indeed lower than "banana"-"fruit", "food"-"fruit" and the threshold after learning, thus not be retained in *BRAIN*. These results show that CogSolver is able to distinguish between different relationships and filter out invalid knowledge carefully.

2) Word-operator relation: We count the number of words related to each operator after learning and select the top 2 with the greatest relation strength $wo_{i,c}$ by (14) in Table IV. For instance, 121 words are related to "$-$" on Math23K, with the top 2 being "minus" and "remainder". We can see that these words do imply the meanings of related operators, showing the relationships acquired by CogSolver are reasonable. Besides, there are only $5, 60$ and 8 words that remain related to $-, \times, \div$ respectively on MAWPS, compared to 569 for $+$. Combining

TABLE IV
NUMBER OF WORDS AND TOP 2 WITH GREATEST RELATION STRENGTH FOR EACH OPERATOR.

	Math23k		MAWPS	
	number	top 2	number	top 2
+	3	total,add	569	total,more
−	121	minus,remainder	5	lost,sell
×	16	per,each	60	group,percent
÷	7	divide,average	8	half,split

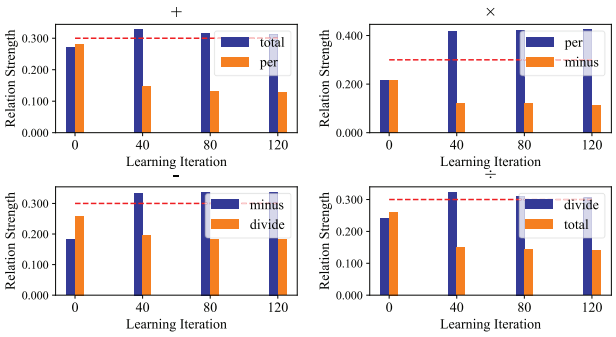


Fig. 8. Word-operator relation $wo_{i,c}$ (examples) on Math23K. The red dash line represents the threshold $\delta_2 = 0.3$ in (14).

it with the result in section V-D1, we can conclude that the amount of data has an important impact on the learning effect.

To better understand CogSolver's competence in capturing the knowledge, we select examples for $\{+, \times, -, \div\}$ on Math23K, since its distribution of operators (except $\wedge$) is relatively uniform (Figure 4). From Figure 8, relationships between "total" and "+", "per" and "×", "minus" and "−", "divided" and "÷" strengthen and are stored in *BRAIN* after learning since they exceed the threshold δ_2. Comparatively, relationships between "per" and "+", 'minus" and "×", and so on weaken gradually. It indicates that CogSolver can learn the knowledge between words and operators correctly.

F. Case Study

To illustrate how our CogSolver benefits from learned knowledge to reason answers, we select two problems for case study in Figure 9. We first report the descriptions of these problems and the predicted prefix expressions generated by our CogSolver and Graph2Tree. Notice that "and" appears in each problem, we then visualize the relationships between "and" and "+", "and" and goal at each reasoning step of CogSolver.

From the black line, the relation strength between "and" and "+" is quite high ($wo^t_{i,c} \approx 1$) at the first reasoning step (i.e., $t = 1$) for both problems. That is probably because "and" appears at a position that implies the meaning of summation (sum of "kittens", "Ruby's"). Moreover, the strength is still high at $t = 2, ..., 5$ for problem 1, which needs to do summation by "and" ($n1+n2$), while decreasing for problem 2 whose goal is irrelevant to summation. These results indicate that CogSolver can adaptively apply the knowledge ("and"-"+") based on contextual information, which further prove the importance and effectiveness of the knowledge-aware module.

The blue bar represents the relationship wg^t_i between "and" and goal q_t, reflecting how focused the model is on "and" at step t. For problem 1, it increases significantly when predicting "+" at $t = 2$ that starts to complete $n1 + n2$. Comparatively, the weight is kept at a low level ($<1e-6$) in problem 2, showing that the model considers "and" to be irrelevant to the target and ignores it at each reasoning step.

Combining the above analysis, by paying more attention to "and" and relating it with "+", CogSolver correctly predicts "+" in problem 1. However, Graph2Tree can not use such information and thus makes a mistake at step 2, further resulting in a wrong answer. Meanwhile, CogSolver takes less into account "and" in problem 2, and therefore does not overestimate the probability of "+". From these observations, we can conclude that our CogSolver has the advantage of applying learned knowledge to different problems. Meanwhile, by incorporating such knowledge into the goal-driven decoder, it achieves interpretable reasoning processes.

VI. CONCLUSION AND FUTURE WORK

In this paper, we presented a novel Cognitive Solver (CogSolver) that autonomously learned knowledge to reason mathematical answers by constructing two systems (*BRAIN-ARM*) and three steps (*Store-Apply-Update* iteratively) inspired by cognitive science theories. In CogSolver, *BRAIN* stored three types of knowledge, and *ARM* applied such knowledge to reason the answers. To improve *ARM*'s reasoning ability, we proposed a knowledge-aware module and a commutative module to apply knowledge effectively. When updating knowledge in *BRAIN*, we developed knowledge filters to reduce redundancy. From the experimental results, we first showed the effectiveness of CogSolver to solve MWP. Then, we verified the reasonability of learning results of CogSolver. Finally, we showed how CogSolver benefited from learned knowledge to achieve interpretable reasoning processes.

There are still some important issues that can be explored in the future. First, as our CogSolver is a general framework, we will test its performance on other kinds of mathematical problems. Second, we will include pre-trained language models into CogSolver to further promote its comprehension ability. Third, we will consider various knowledge in other fields (e.g., physics) and explore the specific meaning of knowledge (e.g., hypernymy) to improve the precision of learning results.

Acknowledgement. This research was partially supported by grants from the National Key Research and Development Program of China (No.2021YFF0901005), the National Natural Science Foundation of China (Grants No. 62106244, 61922073, and U20A20229). The authors would like to thank Prof. Chengxiang Zhai from the University of Illinois at Urbana-Champaign (UIUC) for his constructive advice.

REFERENCES

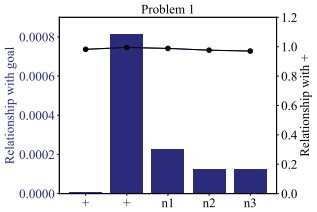
[1] R. C. Atkinson and R. M. Shiffrin. Human memory: A proposed system and its control processes. In *Psychology of learning and motivation*, volume 2, pages 89–195. Elsevier, 1968.

[2] Y. Bakman. Robust understanding of word problems with extraneous information. *arXiv preprint math/0701393*, 2007.

[3] M. Bloch. Language, anthropology and cognitive science. *Man*, pages 183–198, 1991.

Problem 1: Tim 's cat had kittens . He gave $n1$ to Jessica and $n2$ to Sara . He now has $n3$ kittens . How many kittens did he have to start with ?

Our model: $+ + n1\ n2\ n3$ (**correct**)
Graph2Tree: $+ n1\ n3$ (**wrong**)



Problem 2: Ruby has $n1$ candies and $n2$ bananas. If she shares the candies among $n3$ friends, how many candies does each friend get ?

Our model: $\div\ n1\ n3$ (**correct**)
Graph2Tree: $-\ n1\ n3$ (**wrong**)

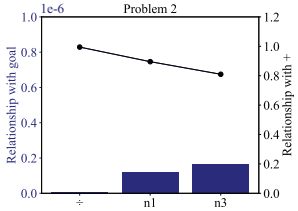


Fig. 9. Two cases of reasoning processes of CogSolver. The x-axis presents the expression generation process. The black line and blue bar represent the relationship between "and" and "+", goal in each generation step respectively.

[4] E. Cambria, Y. Song, H. Wang, and A. Hussain. Isanette: A common and common sense knowledge base for opinion mining. In *2011 IEEE 11th International Conference on Data Mining Workshops*, pages 315–322. IEEE, 2011.

[5] Y. Cao, F. Hong, H. Li, and P. Luo. A bottom-up dag structure extraction model for math word problems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 39–46, 2021.

[6] N. Çeliköz, Y. Erisen, and M. Sahin. Cognitive learning theories with emphasis on latent learning, gestalt and information processing theories. *Online Submission*, 9(3):18–33, 2019.

[7] A. Davies, P. Veličković, Buesing, et al. Advancing mathematics by guiding human intuition with ai. *Nature*, 600(7887):70–74, 2021.

[8] M. Ding, C. Zhou, Q. Chen, H. Yang, and J. Tang. Cognitive graph for multi-hop reading comprehension at scale. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2694–2703, 2019.

[9] Z. Du, C. Zhou, M. Ding, H. Yang, and J. Tang. Cognitive knowledge graph reasoning for one-shot relational learning. *arXiv preprint arXiv:1906.05489*, 2019.

[10] J. S. B. Evans. Dual-processing accounts of reasoning, judgment, and social cognition. *Annu. Rev. Psychol.*, 59:255–278, 2008.

[11] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[12] D. Huang, J. Liu, C.-Y. Lin, and J. Yin. Neural math word problem solver with reinforcement learning. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 213–223, 2018.

[13] Z. Huang, X. Lin, H. Wang, Q. Liu, E. Chen, J. Ma, Y. Su, and W. Tong. Disenqnet: Disentangled representation learning for educational questions. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 696–704, 2021.

[14] Z. Huang, Q. Liu, Y. Chen, L. Wu, K. Xiao, E. Chen, H. Ma, and G. Hu. Learning or forgetting? a dynamic approach for tracking the knowledge proficiency of students. *ACM Transactions on Information Systems (TOIS)*, 38(2):1–33, 2020.

[15] Z. Huang, Q. Liu, W. Gao, J. Wu, Y. Yin, H. Wang, and E. Chen. Neural mathematical solver with enhanced formula structure. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1729–1732, 2020.

[16] D. Kahneman. *Thinking, fast and slow*. Macmillan, 2011.

[17] H. Kim, J. Hwang, T. Yoo, and Y.-G. Cheong. Improving a graph-to-tree model for solving math word problems. In *2022 16th International Conference on Ubiquitous Information Management and Communication (IMCOM)*, pages 1–7. IEEE, 2022.

[18] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[19] R. Koncel-Kedziorski, S. Roy, A. Amini, N. Kushman, and H. Hajishirzi. 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, 2016.

[20] J. Li, L. Wang, J. Zhang, Y. Wang, B. T. Dai, and D. Zhang. 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, 2019.

[21] Z. Liang, J. Zhang, J. Shao, and X. Zhang. Mwp-bert: A strong baseline for math word problems. *arXiv preprint arXiv:2107.13435*, 2021.

[22] A. Lieto, D. P. Radicioni, and V. Rho. Dual peccs: a cognitive system for conceptual representation and categorization. *Journal of Experimental & Theoretical Artificial Intelligence*, 29(2):433–452, 2017.

[23] X. Lin, Z. Huang, et al. Hms: A hierarchical solver with dependency-enhanced understanding for math word problem. In *AAAI*, volume 35, pages 4232–4240, 2021.

[24] D. Ma, Y. Chen, C. Wang, H. Pei, Y. Zhai, G. Zheng, and Q. Chen. Definition-augmented jointly training framework for intention phrase mining. In *International Conference on Database Systems for Advanced Applications*, pages 331–339. Springer, 2022.

[25] C. D. Manning, M. Surdeanu, J. Bauer, J. R. Finkel, S. Bethard, and D. McClosky. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, pages 55–60, 2014.

[26] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[27] S. Roy and D. Roth. Solving general arithmetic word problems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1743–1752, 2015.

[28] M. Seo, H. Hajishirzi, A. Farhadi, O. Etzioni, and C. Malcolm. Solving geometry problems: Combining text and diagram interpretation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1466–1476, 2015.

[29] S. Shehata. A wordnet-based semantic model for enhancing text clustering. In *2009 IEEE International Conference on Data Mining Workshops*, pages 477–482. IEEE, 2009.

[30] J. Shen, Y. Yin, L. Li, L. Shang, X. Jiang, M. Zhang, and Q. Liu. Generate & rank: A multi-task framework for math word problems. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 2269–2279, 2021.

[31] S. Shi, Y. Wang, C.-Y. Lin, X. Liu, and Y. Rui. 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, 2015.

[32] H. A. Simon. Information-processing theory of human problem solving. *Handbook of learning and cognitive processes*, 5:271–295, 1978.

[33] S. Tong, Q. Liu, W. Huang, Z. Hunag, E. Chen, C. Liu, H. Ma, and S. Wang. Structure-based knowledge tracing: an influence propagation view. In *2020 IEEE International Conference on Data Mining (ICDM)*, pages 541–550. IEEE, 2020.

[34] L. Wang, Y. Wang, D. Cai, D. Zhang, and X. Liu. 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, 2018.

[35] L. Wang, D. Zhang, J. Zhang, X. Xu, L. Gao, B. T. Dai, and H. T. Shen. Template-based math word problem solvers with recursive neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 7144–7151, 2019.

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

[37] Q. Wu, Q. Zhang, J. Fu, and X.-J. Huang. A knowledge-aware sequence-to-tree network for math word problem solving. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7137–7146, 2020.

[38] Z. Xie and S. Sun. 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, 2019.

[39] D. Zhang, L. Wang, et al. The gap of semantic parsing: A survey on automatic math word problem solvers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(9):2287–2305, 2020.

[40] J. Zhang, L. Wang, R. K.-W. Lee, Y. Bin, Y. Wang, J. Shao, and E.-P. Lim. Graph-to-tree learning for solving math word problems. pages 3928–3937. Association for Computational Linguistics, 2020.