# Learning by Applying: A General Framework for Mathematical Reasoning via Enhancing Explicit Knowledge Learning

**Jiayu Liu**[1, 2], **Zhenya Huang**[1, 2*], **Chengxiang Zhai**[3], **Qi Liu**[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
[3]University of Illinois at Urbana-Champaign
jy251198@mail.ustc.edu.cn, {huangzhy, qiliuql}@ustc.edu.cn, czhai@illinois.edu

## Abstract

Mathematical reasoning is one of the crucial abilities of general artificial intelligence, which requires machines to master mathematical logic and knowledge from solving problems. However, existing approaches are not transparent (thus not interpretable) in terms of what knowledge has been learned and applied in the reasoning process. In this paper, we propose a general Learning by Applying (LeAp) framework to enhance existing models (backbones) in a principled way by explicit knowledge learning. In LeAp, we perform knowledge learning in a novel *problem-knowledge-expression* paradigm, with a Knowledge Encoder to acquire *knowledge* from *problem* data and a Knowledge Decoder to apply *knowledge* for *expression* reasoning. The learned mathematical knowledge, including word-word relations and word-operator relations, forms an explicit knowledge graph, which bridges the knowledge "learning" and "applying" organically. Moreover, for problem solving, we design a semantics-enhanced module and a reasoning-enhanced module that apply knowledge to improve the problem comprehension and symbol reasoning abilities of any backbone, respectively. We theoretically prove the superiority of LeAp's autonomous learning mechanism. Experiments on three real-world datasets show that LeAp improves all backbones' performances, learns accurate knowledge, and achieves a more interpretable reasoning process.

## 1 Introduction

Mathematical reasoning is one of the core abilities and signs of the intelligence level of general artificial intelligence (Zhang, Wang et al. 2020). It requires machines to grasp mathematical knowledge and logical thinking from solving several mathematical problems (Lewkowycz et al. 2022; Seo et al. 2015). Among them, we specifically study math word problems (MWP) in this paper, which is a fundamental reasoning task that has attracted much attention since the 1960s (Feigenbaum, Feldman et al. 1963). Figure 1 illustrates a toy example. Generally, a math word problem is represented as a problem sentence ("Amy has ... have?") that poses a question requesting an unknown quantity. To solve it, the machine needs to understand the verbal description that contains words (e.g.,"Amy") and quantities (e.g.,"2"),

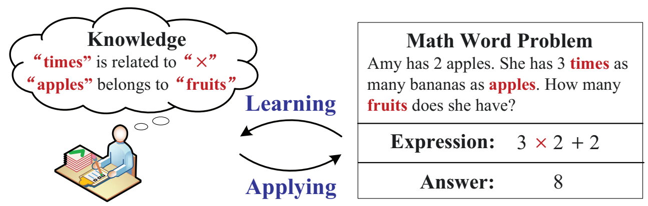


Figure 1: Learning and applying knowledge for MWP.

and then reason a mathematical expression (i.e.,"$3 \times 2 + 2$"), based on which finally get the answer (i.e.,"8").

In the literature, traditional MWP solvers include rule-based, statistic-based, and semantics parsing-based (Zhang, Wang et al. 2020). In recent years, sequence to sequence (Seq2Seq) framework has thrived in MWP task (Wang, Liu, and Shi 2017; Lan et al. 2022), following a *problem-expression* paradigm to translate problems into expressions, where the existing work has focused on improving MWP in two ways, including promoting problem understanding (Lin et al. 2021; Kim et al. 2022) and improving expression reasoning (Zhang et al. 2020a; Cao et al. 2021). However, such a *problem-expression* paradigm is still far from encompassing human-like mathematical reasoning capacity, because it lacks the processes of learning and applying explicit knowledge. On one hand, according to the educational theory of cognitivism (Mowrer 1960; Muhajirah 2020), humans acquire explicit mathematical knowledge from solving problems (Liu et al. 2019), e.g., the word "times" is related to the operator "×" and "apples" belongs to "fruits" from the problem in Figure 1. On the other hand, humans produce solutions for MWP by applying this knowledge in logical thinking (Liang et al. 2018). It is necessary to integrate these two processes organically into machines to build a stronger AI (Tsatsou et al. 2021; de Penning et al. 2011).

From this perspective, existing MWP solvers have some room for improvement. First, they ignore the learning process of explicit knowledge. To be specific, the learned knowledge of existing solvers is implicitly contained in the parameters and network architectures, which is not transparent to humans. Second, the lack of applying explicit knowledge (e.g., "times" is related to "×" in Figure 1) hinders their reasoning ability and interpretability of how they rea-

*Corresponding Author.

son the answers. More importantly, we emphasize that both knowledge learning and applying are general human capabilities (Huang et al. 2020a) that can benefit different MWP solvers. To this end, this paper aims to construct a general framework in which different MWP solvers can learn and apply explicit knowledge to achieve better reasoning ability.

However, many challenges remain to be confronted. First, there is no clear scheme of how to formalize the explicit knowledge that is helpful for MWP and applicable to different solvers. Second, it is challenging to probe a learning mechanism that simulates how humans gain knowledge from solving MWP, which meanwhile should be general to work with different solvers. Third, we need to design a general knowledge application mechanism based on distinct solver architectures, which is underexplored nowadays.

To address these challenges, we propose a novel framework named Learning by Applying (LeAp) for MWP, which adopts a *problem-knowledge-expression* architecture. In LeAp, we define two types of explicit and general mathematical knowledge, including word-word relations and word-operator relations. Then, we implement LeAp by a Variational AutoEncoder (VAE) that contains a Knowledge Encoder and a Knowledge Decoder. Specifically, Knowledge Encoder acquires *knowledge* from *problem* sentences, and Knowledge Decoder applies the learned *knowledge* to reason corresponding *expressions*. The combination of these two components constitutes our novel "learning knowledge by applying it" mechanism. The learned knowledge explicitly forms a knowledge graph in the middle and serves as a bridge connecting the two components, which is transparent. Moreover, for MWP solving, we propose a semantics-enhanced module and a reasoning-enhanced module in Knowledge Decoder, which apply knowledge to promote problem comprehension and symbol reasoning, respectively. Our LeAp is a general framework that benefits existing MWP solvers by improving their reasoning abilities. We conduct extensive experiments by instantiating several backbone solvers in LeAp. The experimental results on three datasets show the improvements of LeAp on answer reasoning, effect of knowledge learning, as well as reasoning interpretability. The contributions of this paper are as follows:

- We propose a general Learning by Applying (LeAp) framework to learn and apply explicit knowledge, where existing MWP solvers can serve as its backbone and benefit from it to improve reasoning ability.
- We design a novel semantics-enhanced module and a novel reasoning-enhanced module for knowledge application, which enhance existing MWP solvers on both answer accuracy and reasoning interpretability.
- We theoretically analyze the superiority of our autonomous knowledge learning mechanism in LeAp and experimentally validate its effectiveness.

## 2 Related Work

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

**Math Word Problems.** Early efforts to solve MWP range from rule-based methods (Bakman 2007; Fletcher 1985), statistic-based methods (Hosseini et al. 2014; Mitra and Baral 2016), to semantics parsing-based methods (Koncel-Kedziorski et al. 2015; Shi et al. 2015). They are characterized by relying on manually crafted rules, machine learning models, and semantic structure of problems, respectively. Recently, Wang et al. (2017) first proposed a seq2seq model that translated the problem sentence into expression, following a *problem-expression* paradigm. Based on such a manner, we summarize advanced methods into two categories: semantics-focused and reasoning-focused. Specifically, semantics-focused methods aim to promote the comprehension of problems with advanced networks (e.g., graph neural networks (Zhang et al. 2020b), pre-trained language models (Liang et al. 2021; Shen et al. 2021; Kim et al. 2022; Yu et al. 2021; Huang et al. 2021)), or additional information (Lin et al. 2021; Wu, Zhang, and Wei 2021; Huang et al. 2020b). For example, Zhang et al. (2020b) proposed Graph2Tree to capture the relationships and order information among quantities. Reasoning-focused methods aim to improve the expression reasoning process (Wang et al. 2019; Shen and Jin 2020; Cao et al. 2021; Jie, Li, and Lu 2022), such as GTS (Xie and Sun 2019) that applied the goal-driven decomposition mechanism to reason an expression tree. Besides, Shen et al. (2020) produced an ensemble of multiple encoders and decoders, combining their advantages in both semantic understanding and reasoning.

**Knowledge Learning and Applying.** One of our most crucial targets is that LeAp can learn and apply explicit knowledge for MWP. Thus, we summarize the related work regarding knowledge learning and knowledge applying, respectively. For knowledge learning, it expects machines to gain knowledge from data (de Penning et al. 2011; Labhishetty et al. 2022). Since our knowledge in LeAp forms an explicit knowledge graph, a relevant task is link prediction (Chen et al. 2021; Cai and Ji 2020) (or knowledge graph completion (Bansal et al. 2019; Cheng et al. 2021)), which generally learns unknown knowledge (edges) in a knowledge graph with the existing edges. For example, Bordes et al. (2013) considered the semantics of knowledge and interpreted it as translation operation. Pei et al. (2019) captured the structure information and the long-range dependencies via a novel geometric perspective. Some researchers also investigate learning other types of knowledge, e.g., background knowledge (Peyrard and West 2020), logic knowledge (Dai and Muggleton 2021), and implicit knowledge in pre-trained language models (Petroni et al. 2019). For knowledge applying, various types of knowledge have been applied in many machine learning tasks, such as conversation generation (Zhou et al. 2018), question answering (Liu et al. 2021), and recommender systems (Xia et al. 2021). Some special forms of knowledge (e.g., logic rules (Qu and Tang 2019), mathematical property (Pei et al. 2020)) have also played an important role in many studies. We refer the readers to a more detailed survey conducted by Laura von Rueden et al. (2021). Especially, Wu et al. (2020) made a basic attempt to incorporate an external manually constructed knowledge base for MWP task.

Different from previous studies on knowledge learning, our LeAp gains mathematical knowledge by applying it to reason answers. Compared with existing work on MWP,

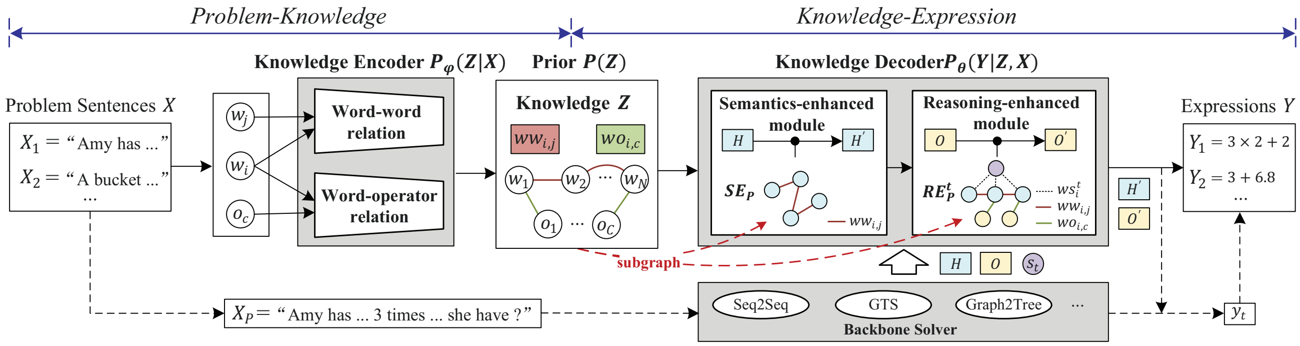


Figure 2: The architecture of LeAp, which operates in a *problem-knowledge-expression* paradigm.

it is a general framework that empowers different solvers with explicit knowledge learning and applying in a novel *problem-knowledge-expression* paradigm, further improving their reasoning abilities. Besides, we propose a semantics-enhanced module and a reasoning-enhanced module in LeAp to apply knowledge for MWP solving, which bring better answer accuracy and reasoning interpretability.

# 3 LeAp: Learning by Applying

In this section, we first formally define our problem and then introduce the architecture of our LeAp in details.

## 3.1 Problem Definition

A MWP dataset is denoted as $D = (X, Y)$, where $X$ is the set of problem sentences and $Y$ is the set of corresponding expressions. Specifically, $X_P = \{w_1, ..., w_n\} \in X$ is a sequence of $n$ word tokens and numeric values of problem $P$, where $w_i$ is either a word token (e.g., "Amy" in Figure 1) or a numeric value (e.g., "2"). $Y_P = \{y_1, ..., y_m\} \in Y$ is a sequence of $m$ symbols. Each symbol $y_i$ comes from a target vocabulary $V_P$ composed of the operator set $V_O$ (e.g., $\{+, \times, -, \div\}$), numeric constant set $V_N$ (e.g., $\pi$), and numeric values $N_P$ in $X_P$, i.e., $V_P = V_O \cup V_N \cup N_P$. Note that different problems may have different $V_P$ since $N_P$ varies with $P$. The goal of MWP is to train a solver that reads the problem sentence $X_P$, generates a valid mathematical expression $Y_P$, and gets a numeric answer $a_P$ based on $Y_P$.

The knowledge $Z$ we consider is explicitly represented as a mathematical knowledge graph. Its vertices include words and operators, and the knowledge we focus on is the existence of its edges. Specifically, we formalize $Z$ as word-word relation $ww_{i,j}$ and word-operator relation $wo_{i,c}$, i.e., $Z = \{ww_{i,j}, wo_{i,c} | i, j = 1, ..., N; c = 1, ..., C\}$, where $N$ and $C$ are the number of words and operators in MWP task. $ww_{i,j}$ describes the relationship between words $w_i$ and $w_j$ (e.g., "apples" belongs to "fruits"), while $wo_{i,c}$ captures the one between word $w_i$ and operator $o_c$ (e.g., "times" is related to "×"). Both $ww_{i,j}$ and $wo_{i,c}$ are set as binary variables, which equal 1 if there exists knowledge between word $w_i$ and word $w_j$ (or operator $o_c$), and 0 otherwise. Please note that our formulation can be easily extended to incorporate multiple relationships (edges) between words and operators, such as hypernymy and antonymy (Shehata 2009).

Our goal is to build a framework that (1) learns mathematical knowledge $Z$ from solving MWP; (2) applies knowledge $Z$ to reason the answers for MWP. These two goals are coupled with each other and achieved collaboratively.

## 3.2 LeAp Architecture

Intuitively, the educational theory of cognitivism (Mowrer 1960; Muhajirah 2020) indicates that a learner fosters a strong connection to the knowledge (e.g., "times" is related to "×") by directly applying it (e.g., solve the problem in Figure 1). Drawing this insight, we construct our LeAp framework with a novel *problem-knowledge-expression* architecture to achieve the mechanism of "learning knowledge by applying it". Specifically, the *problem-knowledge* process acquires *knowledge* from *problem* data, and the *knowledge-expression* process applies this *knowledge* to reason *expressions* for answers, which further guides to learn reasonable knowledge autonomously. To this end, we formalize LeAp as a Variational AutoEncoder (VAE) (Kingma and Welling 2013), which is shown in Figure 2. It consists of three main parts: (1) a Knowledge Encoder $P_\varphi(Z|X)$ that acquires knowledge $Z$ from problem sentences $X = \{X_P\}$ (*problem-knowledge*); (2) a Knowledge Decoder $P_\theta(Y|Z, X)$ that reasons the expressions $Y = \{Y_P\}$ based on $X$ and $Z$ (*knowledge-expression*); (3) a knowledge prior $P(Z)$ of $Z$. The training objective of LeAp is to maximize the Evidence Lower Bound (ELBO):

$$L = \underbrace{\mathrm{E}_{P_\varphi(Z|X)}\left[\log P_\theta(Y|Z,X)\right]}_{L_1} - \underbrace{\mathrm{KL}\left(P_\varphi(Z|X) \| P(Z)\right)}_{L_2}, \quad (1)$$

where the first term $L_1$ optimizes the performance of MWP solving, and the second term $L_2$ regularizes the knowledge learning results. In the following, we will introduce the details of Knowledge Encoder $P_\varphi(Z|X)$, Knowledge Decoder $P_\theta(Y|Z, X)$, and the knowledge prior $P(Z)$ in turn.

**Knowledge Encoder.** Knowledge Encoder $P_\varphi(Z|X)$ aims at acquiring explicit knowledge $Z$ from the problem sentence set $X$ as shown in Figure 2, which operates the *problem-knowledge* process. Since $ww_{i,j}, wo_{i,c} \in Z$ are binary, we map each word $w_i$ and operator $o_c$ to the vector $\boldsymbol{w}_i, \boldsymbol{o}_c \in \mathbb{R}^d$ respectively ($d$ is the dimension), and feed $\boldsymbol{w}_i, \boldsymbol{w}_j, \boldsymbol{o}_c$ into different networks to encode their Bernoulli distributions. Formally, we model Knowledge Encoder as:

$$\begin{aligned} P_\varphi(ww_{i,j}|X) &= \text{Bernoulli}\left(\sigma\left(f_1\left([\boldsymbol{w}_i, \boldsymbol{w}_j]\right)\right)\right), \\ P_\varphi(wo_{i,c}|X) &= \text{Bernoulli}\left(\sigma\left(f_2\left([\boldsymbol{w}_i, \boldsymbol{o}_c]\right)\right)\right), \end{aligned} \quad (2)$$

where $\sigma$ is the sigmoid function, $f_1$ and $f_2$ are neural networks that transform the concatenation $[\cdot]$ of $\boldsymbol{w}_i, \boldsymbol{w}_j$ and $\boldsymbol{w}_i, \boldsymbol{o}_c$, respectively. Note that $f_1, f_2$ can be implemented ranging from MLP to pre-train language models (Petroni et al. 2019). Since we focus more on learning knowledge from MWP, we do not emphasize their difference and adopt MLP (Kipf et al. 2018) for simplicity.

During training, Knowledge Encoder $P_\varphi(Z|X)$ needs to sample $Z$ to estimate $\mathrm{E}_{P_\varphi(Z|X)}\left[\log P_\theta(Y|Z, X)\right]$, i.e., $L_1$ in Eq. (1). However, it is hard to use reparameterization to backpropagate the derivatives since $ww_{i,j}, wo_{i,c} \in Z$ are binary (Kipf et al. 2018). Therefore, we adopt a continuous approximation of Eq. (2) when optimizing Eq. (1):

$$\begin{aligned} ww_{i,j} &= \sigma\left(\left(f_1\left([\boldsymbol{w}_i, \boldsymbol{w}_j]\right) + g_{i,j}\right)/\tau\right), \\ wo_{i,c} &= \sigma\left(\left(f_2\left([\boldsymbol{w}_i, \boldsymbol{o}_c]\right) + g_{i,c}\right)/\tau\right), \end{aligned} \quad (3)$$

where $\{g_{i,j}, g_{i,c}\}$ are i.i.d. sampled from $\mathrm{Gumbel}(0, 1)$ distribution (Jang, Gu, and Poole 2016). $\tau$ is a temperature parameter that controls the degree of approximation.

**Knowledge Decoder.** Knowledge Decoder $P_\theta(Y|Z, X)$ applies the knowledge $Z$ acquired by Knowledge Encoder to reason expressions $Y$, which operates the *knowledge-expression* process in Figure 2. Here, we aim to design a general knowledge application mechanism that benefits different solvers (e.g., Seq2Seq, GTS), rather than propose a special solver architecture. Specifically, we contribute a semantics-enhanced module and a reasoning-enhanced module in Knowledge Decoder, which apply knowledge to improve problem understanding and symbol reasoning, respectively. In the following, we first unify the architecture of most existing solvers into "Backbone Solver". Then, we explain the details of our proposed modules.

**Backbone Solver.** Given problem $P$, a backbone solver first reads the problem sentence $X_P = \{w_1, ..., w_n\}$, and then generates word representations $\mathbf{H} = \{\boldsymbol{h}_1, ..., \boldsymbol{h}_n\}$ and an initial reasoning state $\boldsymbol{s}_1$ by:

$$(\mathbf{H}, \boldsymbol{s}_1) = \text{Sol-Enc}(\{\boldsymbol{w}_i, i = 1, ..., n\}). \quad (4)$$

Sol-Enc conducts problem understanding and captures many existing models in different solvers. It can be formalized ranging from RNN (e.g., GTS, TSN-MD), BERT (e.g., MWP-BERT), to specific MWP encoder (e.g., Graph2Tree). Then, the backbone solver generates the expression $Y_P = \{y_1, ..., y_m\}$ for $P$ step by step. Specifically, at step $t$ ($t = 1, ..., m$), it reasons symbol $y_t$ by:

$$P_\theta(y_t \mid y_1, ..., y_{t-1}, P) = \text{Sol-Dec}(\boldsymbol{s}_t, \boldsymbol{e}(y_t), \textit{options}). \quad (5)$$

In Sol-Dec, $\boldsymbol{s}_t$ is the reasoning state at step $t$ ($\boldsymbol{s}_1$ at step 1 comes from Eq. (4)). $\boldsymbol{e}(y_t)$ is the embedding of symbol $y_t$, which equals $\boldsymbol{o}_c$ if $y_t$ is operator $o_c$, or $\boldsymbol{h}_i$ if $y_t$ is a number $w_i \in N_P$. *options* are some optional terms. Specifically, the meanings of Sol-Dec, $\boldsymbol{s}_t$, and *options* vary with different solvers. For example, in Seq2Seq, Sol-Dec represents the output gate of LSTM, $\boldsymbol{s}_t$ is the hidden state, and *options* $= \emptyset$.

After generating symbol $y_t$, the backbone solver derives the next reasoning state $\boldsymbol{s}_{t+1}$ with another solver-specific network $f_3$ (e.g., $f_3$ represents the forget gate in Seq2Seq):

$$\boldsymbol{s}_{t+1} = f_3(\boldsymbol{s}_t, \boldsymbol{e}(y_t), \textit{options}). \quad (6)$$

Readers can refer to the original papers for more details of different backbone solvers summarized in Section 2.

**Semantics-enhanced Module.** LeAp improves problem comprehension with a semantics-enhanced module for application of word-word knowledge $ww_{i,j} \in Z$ (e.g., understand "how many fruits" in Figure 1 via the knowledge "apples" belongs to "fruits"), which is independent of specific backbone solvers. Specifically, for problem $P$, (1) we construct a graph $SE_P$ as shown in Figure 2 by taking the words in $P$ as vertices and their knowledge $\{ww_{i,j}|i, j = 1, ..., n\}$ from $Z$ as edges. Thus, $SE_P$ can be seen as a subgraph of our overall explicit knowledge $Z$; (2) we get word representations $\mathbf{H}$ by the original backbone according to Eq. (4); (3) we utilize a GCN to pass the message on $SE_P$ to obtain $\mathbf{H}'$ that fuses the relational knowledge:

$$\mathbf{H}' = \mathbf{A_E} \cdot \mathrm{ReLU}(\mathbf{A_E} \cdot \mathbf{H} \cdot \mathbf{W}_1 + b_1) \cdot \mathbf{W}_2 + b_2, \quad (7)$$

where $\mathbf{A_E} \in \mathbb{R}^{n \times n}$ is the adjacency matrix of $SE_P$, $\mathbf{W}_1, \mathbf{W}_2, b_1, b_2$ are trainable parameters.

**Reasoning-enhanced Module.** Now, we introduce how LeAp improves symbol reasoning with our proposed reasoning-enhanced module that applies both word-word knowledge and word-operator knowledge $ww_{i,j}, wo_{i,c} \in Z$. Intuitively, word "times" in a problem sentence can encourage a solver to correctly reason symbol "×" through the relational knowledge $wo_{i,c}$ between "times" and "×", whose location in the expression can be refined based on how focused the solver is on "times". Thus, to guide the reasoning at step $t$, we first establish temporary relationships $ws_i^t$ between current reasoning state $s_t$ and all words in $P$ by:

$$\begin{aligned} ws_i^t &= \frac{\exp\left(f_4\left(\boldsymbol{s}_t, \boldsymbol{h}'_i\right)\right)}{\sum_j \exp\left(f_4\left(\boldsymbol{s}_t, \boldsymbol{h}'_j\right)\right)}, \\ f_4\left(\boldsymbol{s}_t, \boldsymbol{h}'_i\right) &= \boldsymbol{v}^\top \tanh\left(\mathbf{W}_3 \cdot \left[\boldsymbol{s}_t, \boldsymbol{h}'_i\right]\right), \end{aligned} \quad (8)$$

where $\boldsymbol{v}$, $\mathbf{W}_3$ are learnable parameters.

Then, combining $s_t$ and knowledge $Z$, we construct another graph $RE_P^t$ for problem $P$. As depicted in Figure 2, its vertex set contains reasoning state $\boldsymbol{s}_t$, words $\mathbf{H}' = \{\boldsymbol{h}'_1, ..., \boldsymbol{h}'_n\}$ of $P$, and all operators $\mathbf{O} = \{\boldsymbol{o}_1, ..., \boldsymbol{o}_C\}$. The edge set is composed of $ws_i^t$ and knowledge $ww_{i,j}$, $wo_{i,c}$. On $RE_P^t$, we propagate the information from $\boldsymbol{s}_t$ to enhance operator representations by:

$$\begin{aligned} \mathbf{H}^t &= \mathbf{A_E} \cdot \mathrm{ReLU}(\mathbf{A_E} \cdot [ws^t \cdot \boldsymbol{s}_t, \mathbf{H}'] \cdot \mathbf{W}_4 + b_4) \cdot \mathbf{W}_5 + b_5, \\ \hat{\mathbf{H}}^t &= \mathbf{H}^t + \mathrm{LayerNorm}(\mathbf{H}^t), \\ \mathbf{O}^t &= \mathrm{LayerNorm}(\mathrm{ReLU}(\mathbf{A_D} \cdot \hat{\mathbf{H}}^t \cdot \mathbf{W}_6 + b_6)), \\ \hat{\mathbf{O}}^t &= \mathrm{ReLU}([\mathbf{O}^t, \mathbf{O}] \cdot \mathbf{W}_7 + b_7), \\ \mathbf{O}' &= \mathbf{O} + \mathrm{LayerNorm}(\hat{\mathbf{O}}^t), \end{aligned} \quad (9)$$

where $\mathbf{A_D} \in \mathbb{R}^{C \times n}$ is the adjacency matrix of $RE_P^t$, $\mathbf{W}_*, b_*$ are weight matrices and biases.

When reasoning symbol $y_t$ in Eq. (5), $\boldsymbol{e}(y_t)$ is chosen as $\boldsymbol{o}'_c \in \mathbf{O}'$ in Eq. (9) if $y_t$ is operator $o_c$, or $\boldsymbol{h}'_i \in \mathbf{H}'$ in Eq. (7) if $y_t$ is number $w_i$. Finally, the symbol representation $\boldsymbol{e}(y_t)$ enhanced with knowledge $Z$ is used to generate the next reasoning state $\boldsymbol{s}_{t+1}$ by Eq. (6), and $RE_P^t$ evolves to $RE_P^{t+1}$.

**Prior of Knowledge.** The prior $P(Z)$ controls the known information for LeAp. Since knowledge $ww_{i,j}, wo_{i,c} \in Z$ is binary, it is natural to take the following Bernoulli distribution as the prior of $Z$ and set $\delta_1 = 0.1$ for sparsity.

$$ww_{i,j} \sim \text{Bernoulli}(\delta_1),\ wo_{i,c} \sim \text{Bernoulli}(\delta_1). \tag{10}$$

We design the prior $P(Z)$ to simulate a real learner. In practice, a learner may obtain different reasoning performances for MWP with different knowledge backgrounds. For example, a junior high school student has better reasoning ability than a primary school student. Thus, we can set a higher $\delta_1 = 0.5$ for knowledge that a learner has mastered, which we can simulate by introducing part of (e.g., $\alpha = 20\%$) edges of an external knowledge base. With such prior, LeAp can also be guided to learn similar knowledge, thus alleviating the problem of capturing false correlations.

In summary, our LeAp framework has the following advantages. First, LeAp is general to take different MWP solvers as the backbone, enabling them not only to learn reasonable knowledge but also to gain better reasoning ability. Second, LeAp adopts a *problem-knowledge-expression* architecture, where knowledge $Z$ explicitly forms a knowledge graph. Therefore, both its learning results and applying methods are more interpretable compared with previous methods that follow the *problem-expression* paradigm. Third, in the prior of knowledge $Z$, we can set different $\alpha$ to investigate problem solving effects of learners with variable background, which is further visualized in Section 5.3.

## 4 Theoretical Analyses

In brief, our LeAp learns mathematical knowledge from problems and applies it to solve MWP, naturally constructing an explicit knowledge graph (KG) as shown in Figure 2. Comparatively, a straightforward way to learn a KG is the link prediction task (LP) (Chen et al. 2021; Cai and Ji 2020), which predicts the edge between each pair of vertices directly. In this section, we investigate deeper into how LeAp is superior in its autonomous learning mechanism compared with LP. Here, we unify some important notations without loss of generality. Specifically, we use $z_{i,j} \in Z$ to represent the knowledge between vertices $i$ and $j$ in the KG, including word-word relation $ww_{i,j}$ and word-operator relation $wo_{i,c}$. $X$ represents the embeddings of vertices, including words $\{\boldsymbol{w}_i\}$ and operators $\{\boldsymbol{o}_c\}$.

Before detailed derivation, we first assume that the knowledge $Z$ is helpful for solving MWP, because we cannot expect to gain knowledge irrelevant to mathematical reasoning (e.g., chemistry knowledge) from solving mathematical problems. Thus, we give the following definition:

**Definition 1.** *Effective knowledge* $Z$: $\forall z_{i,j} \in Z$,

$$[P_\theta(Y|z_{i,j}=1,X)-P_\theta(Y|z_{i,j}=0,X)]\cdot(2r_{i,j}-1) > 0. \tag{11}$$

$r_{i,j}$ is the ground-truth label of $z_{i,j}$, equalling 1 if there exists true knowledge (i.e., edge) between $i$ and $j$, and 0 otherwise. $P_\theta$ is the Knowledge Decoder of LeAp. In Definition 1, if there exists knowledge between $i$ and $j$ (i.e., $r_{i,j} = 1$), Knowledge Decoder can better reason the expressions $Y$ by applying $z_{i,j} = 1$ than $z_{i,j} = 0$. On the contrary, it will introduce redundancy if we treat the false knowledge (i.e., $r_{i,j} = 0$) as true (i.e., $z_{i,j} = 1$), and thus $P_\theta(Y|z_{i,j} = 1, X) < P_\theta(Y|z_{i,j} = 0, X)$ at this time.

Based on the "Effective knowledge" assumption, we analyze the difference between our LeAp and LP. Recall that LP learns knowledge with a model $P_\varphi(z_{i,j}|X)$, which calculates the posterior probability of $Z$ given $X$ from a Bayesian perspective. Comparatively, LeAp learns knowledge from reasoning expressions $Y$ based on $X$. Thus, we investigate

- (1) **Whether the optimization objective of LeAp in Eq. (1) optimizes the posterior $\boldsymbol{P_\varphi(z_{i,j}|X,Y)}$**
- (2) **If (1) holds, whether $\boldsymbol{P_\varphi(z_{i,j}|X,Y)}$ is larger (i.e., more accurate) than $\boldsymbol{P_\varphi(z_{i,j}|X)}$**

Here, we consider a simplified setting to train LeAp with parameters being initialized with $\varphi^{LP}, X^{LP}$ from a trained LP model. Under such setting, we have two theorems that answer the two questions above, respectively.

**Theorem 1.** Assume $P_\varphi(z_{i,j} = r_{i,j}|X) > \delta(X)$ holds in a neighborhood $U$ of $(\varphi^{LP}, X^{LP})$ and knowledge $Z$ is effective. Then, for each $z_{i,j} \in Z$, maximizing the objective of MWP solving in Eq. (1), i.e., $L_1 = \mathrm{E}_{P_\varphi(z_{i,j}|X)}[\log P_\theta(Y|z_{i,j}, X)]$, is equivalent to maximizing

$$\begin{aligned} L_3 = r_{i,j} \cdot P_\theta(Y|z_{i,j}=1,X) \cdot P_\varphi(z_{i,j}=1|X) + \\ (1-r_{i,j}) \cdot P_\theta(Y|z_{i,j}=0,X) \cdot P_\varphi(z_{i,j}=0|X) \end{aligned} \tag{12}$$

in $U$, where $\delta(X) \triangleq \max\{\frac{1}{1+\frac{\beta(1-r_{i,j},r_{i,j},c)}{\beta(r_{i,j},1-r_{i,j},c)}}\}|_{c=\theta,\boldsymbol{x}_i\in X}$, and $\beta(a,b,c) \triangleq P_\theta(Y|z_{i,j}=a,X)\cdot\|\frac{\partial P_\theta(Y|z_{i,j}=b,X)}{\partial c}\|$.

*Proof.* The basic idea is to verify that the inner product of the derivatives of $L_1$ and $L_3$ is positive for each parameter in LeAp. Thus, under a first order Taylor approximation, the gradient direction of $L_1$ implicitly optimizes $L_3$. On this basis, the entire proof consists of three parts of verification for Knowledge Encoder $\varphi$, Knowledge Decoder $\theta$, and vertex embeddings $X$ respectively. Here $\langle,\rangle$ represents the inner product of two vectors.

1) For $\varphi$ in Knowledge Encoder, $\langle\frac{\partial L_1}{\partial\varphi},\frac{\partial L_3}{\partial\varphi}\rangle =$

$$(2r_{i,j}-1)\cdot\ln\frac{P_\theta(Y|z_{i,j}=1,X)}{P_\theta(Y|z_{i,j}=0,X)}\cdot\|\frac{\partial P_\varphi(z_{i,j}=1|X)}{\partial\varphi}\|^2. \tag{13}$$

According to the definition of "Effective knowledge" in Eq. (11), $\langle\frac{\partial L_1}{\partial\varphi},\frac{\partial L_3}{\partial\varphi}\rangle \geq 0$ always holds.

2) For $\theta$ in Knowledge Decoder, when $r_{i,j} = 1$, we can derive that $\langle\frac{\partial L_1}{\partial\theta},\frac{\partial L_3}{\partial\theta}\rangle =$

$$\begin{aligned} &\|\frac{\partial P_\theta(Y|z_{i,j}=1,X)}{\partial\theta}\|^2\cdot\frac{P_\varphi(z_{i,j}=1|X)^2}{P_\theta(Y|z_{i,j}=1,X)} + \\ &\langle\frac{\partial P_\theta(Y|z_{i,j}=0,X)}{\partial\theta},\frac{\partial P_\theta(Y|z_{i,j}=1,X)}{\partial\theta}\rangle\cdot \\ &\frac{P_\varphi(z_{i,j}=0|X)\cdot P_\varphi(z_{i,j}=1|X)}{P_\theta(Y|z_{i,j}=0,X)} > \\ &\|\frac{\partial P_\theta(Y|z_{i,j}=1,X)}{\partial\theta}\|\cdot P_\varphi(z_{i,j}=1|X)\cdot\beta(1,0,\theta)\cdot \\ &\frac{P_\varphi(z_{i,j}=1|X)\cdot\frac{\beta(0,1,\theta)}{\beta(1,0,\theta)} - P_\varphi(z_{i,j}=0|X)}{P_\theta(Y|z_{i,j}=1,X)\cdot P_\theta(Y|z_{i,j}=0,X)}. \end{aligned} \tag{14}$$

Given the condition $P_\varphi(z_{i,j} = r_{i,j}|X) > \delta(X)$, it is easy to verify that $P_\varphi(z_{i,j}=1|X)\cdot\frac{\beta(0,1,\theta)}{\beta(1,0,\theta)} > P_\varphi(z_{i,j}=0|X)$, and thus $\langle\frac{\partial L_1}{\partial\theta},\frac{\partial L_3}{\partial\theta}\rangle > 0$ in $U$. The proof for $r_{i,j} = 0$ is similar.

3) For $\boldsymbol{x}_i \in X$ , we also report the result of $r_{i,j} = 1$. Since LeAp is initialized with a trained LP model, $(\varphi^{LP}, X^{LP})$ can be seen as a locally optimal solution of $P_\varphi(z_{i,j} = 1|X)$, and thus $\frac{\partial P_\varphi(z_{i,j}=1|X)}{\partial \boldsymbol{x}_i} \approx 0$ holds in $U$. Based on it, we can derive that $\langle \frac{\partial L_1}{\partial \boldsymbol{x}_i}, \frac{\partial L_3}{\partial \boldsymbol{x}_i} \rangle \approx$

$$
\begin{aligned}
&\|\frac{\partial P_\theta(Y|z_{i,j}=1,X)}{\partial \boldsymbol{x}_i}\|^2 \cdot \frac{P_\varphi(z_{i,j}=1|X)^2}{P_\theta(Y|z_{i,j}=1,X)} + \\
&\langle \frac{\partial P_\theta(Y|z_{i,j}=1,X)}{\partial \boldsymbol{x}_i}, \frac{\partial P_\theta(Y|z_{i,j}=0,X)}{\partial \boldsymbol{x}_i} \rangle \cdot \\
&\frac{P_\varphi(z_{i,j}=0|X) \cdot P_\varphi(z_{i,j}=1|X)}{P_\theta(Y|z_{i,j}=0,X)} > \\
&\|\frac{\partial P_\theta(Y|z_{i,j}=1,X)}{\partial \boldsymbol{x}_i}\| \cdot P_\varphi(z_{i,j}=1|X) \cdot \beta(1,0,\boldsymbol{x}_i) \cdot \\
&\frac{P_\varphi(z_{i,j}=1|X) \cdot \frac{\beta(0,1,\boldsymbol{x}_i)}{\beta(1,0,\boldsymbol{x}_i)} - P_\varphi(z_{i,j}=0|X)}{P_\theta(Y|z_{i,j}=1,X) \cdot P_\theta(Y|z_{i,j}=0,X)}. \qquad (15)
\end{aligned}
$$

Similar to Eq. (14), it is easy to verify that Eq. (15) is larger than 0, and thus $\langle \frac{\partial L_1}{\partial \boldsymbol{x}_i}, \frac{\partial L_3}{\partial \boldsymbol{x}_i} \rangle > 0$ in $U$. □

In Theorem 1, we further notice that $L_3$ can be rewritten as $P(Y|z_{i,j} = r_{i,j}, X) \cdot P(z_{i,j} = r_{i,j}|X)$ if $P_\varphi(z_{i,j}|X)$ and $P_\theta(Y|z_{i,j}, X)$ reconstruct the true distribution $P(z_{i,j}|X)$ and $P(Y|z_{i,j}, X)$ behind dataset $D$. Thus, $L_3$ is equivalent to the following posterior, answering our question (1):

$$
\begin{aligned}
&P(z_{i,j} = r_{i,j}|X, Y) \\
&= \frac{P(Y|z_{i,j} = r_{i,j}, X) \cdot P(z_{i,j} = r_{i,j}|X) \cdot P(X)}{P(X, Y)}. \qquad (16)
\end{aligned}
$$

**Theorem 2.** Under the assumption of "Effective knowledge", the following inequality holds:

$$
\frac{P(Y|z_{i,j} = r_{i,j}, X) \cdot P(X)}{P(X, Y)} > 1. \qquad (17)
$$

*Proof.* $P(X, Y)$ can be rewritten as

$$
\begin{aligned}
&P(X) \cdot [P(Y|z_{i,j} = r_{i,j}, X) \cdot P(z_{i,j} = r_{i,j}|X) + \\
&P(Y|z_{i,j} = 1 - r_{i,j}, X) \cdot P(z_{i,j} = 1 - r_{i,j}|X)]. \qquad (18)
\end{aligned}
$$

According to the "Effective knowledge" assumption, $P(Y|z_{i,j} = 1 - r_{i,j}, X) < P(Y|z_{i,j} = r_{i,j}, X)$. Therefore, Eq. (18) is less than

$$
\begin{aligned}
&P(X) \cdot [P(Y|z_{i,j} = r_{i,j}, X) \cdot P(z_{i,j} = r_{i,j}|X) + \\
&P(Y|z_{i,j} = r_{i,j}, X) \cdot P(z_{i,j} = 1 - r_{i,j}|X)] \qquad (19) \\
&= P(X) \cdot P(Y|z_{i,j} = r_{i,j}, X),
\end{aligned}
$$

i.e., inequality. (17) holds. □

Based on Theorem 2 and Eq. (16), $P(z_{i,j} = r_{i,j}|X, Y) > P(z_{i,j} = r_{i,j}|X)$, supporting our question (2). In summary, we conclude that: According to Theorem 1, in LeAp, the autonomous mechanism of learning knowledge from solving MWP is to optimize the posterior $P(z_{i,j} = r_{i,j}|X, Y)$ by $L_1$ in Eq. (1). Such a mechanism is more accurate than a straightforward link prediction model $P(z_{i,j} = r_{i,j}|X)$ since it achieves a higher probability according to Theorem 2. Notably, LeAp's superiority lies in the mechanism to calculate the posterior probability based on the information (i.e., $Y$) provided by solving MWP, instead of obtaining better parameters $\varphi, X$.

# 5 Experiments

## 5.1 Experimental Setup

**Datasets.** We use three datasets in experiments: **Math23K**, **MAWP**, and **SVAMP**. Specifically, **Math23K** (Wang, Liu, and Shi 2017) is a Chinese dataset that contains $23,162$ problems that have only one variable. We use its the published dataset partition for experiments. **MAWPS** (Roy and Roth 2017) is an English dataset. We select $2,373$ problems with only one unknown variable and conduct 5-fold cross-validation. **SVAMP** (Patel, Bhattamishra, and Goyal 2021) is a test dataset that contains $1,000$ more difficult problems than MAWPS. Following Patel et al. (2021), models are trained on the combination of MAWPS and another ASDiv-A dataset and tested on the problems in SVAMP.

**Baselines.** We take the following SOTA models as the baselines, i.e., the basic Seq2Seq, semantics-focused methods (Graph2Tree, HMS), reasoning-focused methods (GTS, TSN-MD), and ensemble-based method (Multi-E/D). Our LeAp is a general framework, and thus we take all of them as the backbones to evaluate its effectiveness and generality.

- **Seq2Seq** (Luong, Pham, and Manning 2015) adopts a BiLSTM encoder and a LSTM decoder with attention to translate a problem sentence into an expression.
- **Graph2Tree** (Zhang et al. 2020b) builds and encodes two quantity-related graphs to enrich problem understanding with quantity information.
- **HMS** (Lin et al. 2021) captures problem semantics following a word-clause-problem hierarchy.
- **GTS** (Xie and Sun 2019) proposes a goal-driven decomposition mechanism to reason the expression tree.
- **TSN-MD** (Zhang et al. 2020a) generates diverse candidate expressions with a multiple-decoder network.
- **Multi-E/D** (Shen and Jin 2020) is an ensemble of sequence-based encoder/decoder with graph-based encoder/decoder to obtain better semantics and reasoning.

**Implementation Details.** For Knowledge Encoder $P_\varphi$, the dimension of embeddings $d$ is 128. The temperature parameter $\tau$ in Eq. (3) decreases from $0.5$ to $0.1$ during training. For Knowledge Decoder $P_\theta$, all backbone solvers are optimized with their original parameter settings. Other parameters are initialized randomly and trained with Adam (Kingma and Ba 2014) with dropout probability $0.5$. Words with less than 5 occurrences are converted to a special token "UNK". All experiments are run on a Linux server with four 2.30GHz Intel Xeon Gold 5218 CPUs and a Tesla V100 GPU. Our codes are available at *https://github.com/bigdata-ustc/LeAp*.

## 5.2 Performance on Answer Reasoning

We verify the effectiveness of LeAp in improving the reasoning ability of backbones for MWP, using answer accuracy as the metric since there may be many correct expressions for the same problem. The predicted expression is considered correct if its calculated value equals the true answer.

In Table 1, we first report the performances of all backbones in their original version ("ORI") and with our LeAp framework ("LeAp"). From Table 1, we observe that LeAp

| | Math23K | | | MAWPS | | | SVAMP | | |
|---|---|---|---|---|---|---|---|---|---|
| | ORI | LeAp | LeAp-EK | ORI | LeAp | LeAp-EK | ORI | LeAp | LeAp-EK |
| Seq2Seq | 0.640 | **0.660**$^{**}$ | 0.652$^{**}$ | 0.797 | 0.803 | **0.807**$^{*}$ | 0.200 | **0.236**$^{***}$ | 0.220$^{***}$ |
| Graph2Tree | 0.774 | 0.779$^{*}$ | **0.782**$^{**}$ | 0.837 | **0.852**$^{**}$ | 0.849$^{**}$ | 0.319 | **0.341**$^{***}$ | 0.325$^{*}$ |
| HMS | 0.761 | **0.769** | 0.765 | 0.803 | **0.812**$^{*}$ | 0.805 | 0.179 | **0.196**$^{**}$ | 0.191$^{**}$ |
| GTS | 0.756 | **0.772**$^{**}$ | 0.767$^{**}$ | 0.826 | **0.834**$^{**}$ | 0.830$^{*}$ | 0.277 | **0.285** | 0.279 |
| TSN-MD | 0.774 | **0.786**$^{**}$ | 0.778$^{*}$ | 0.844 | **0.853**$^{*}$ | 0.848 | 0.290$^{\dagger}$ | **0.302**$^{**}$ | 0.294$^{*}$ |
| Multi-E/D | 0.784 | 0.791$^{*}$ | **0.793**$^{**}$ | / | / | / | / | / | / |

Table 1: Answer accuracy ($***: p \leq 0.001, **: p \leq 0.01, *: p \leq 0.05$). †: implemented by MTPToolkit (Lan et al. 2022).

| LeAp (backbone) | Math23K | | MAWPS | | SVAMP | |
|---|---|---|---|---|---|---|
| | w/o SE | w/o RE | w/o SE | w/o RE | w/o SE | w/o RE |
| Seq2Seq | 0.645 | 0.648 | 0.798 | 0.802 | 0.210 | 0.228 |
| Graph2Tree | 0.778 | 0.776 | 0.846 | 0.845 | 0.335 | 0.328 |
| HMS | 0.766 | 0.762 | 0.809 | 0.801 | 0.193 | 0.189 |
| GTS | 0.770 | 0.761 | 0.830 | 0.830 | 0.265 | 0.284 |
| TSN-MD | 0.782 | 0.783 | 0.847 | 0.852 | 0.293 | 0.300 |
| Multi-E/D | 0.790 | 0.788 | / | / | / | / |

Table 2: Ablation study on reducing semantics-enhanced module ("w/o SE") or reasoning-enhanced module ("w/o RE"). The performance of LeAp is referred in Table 1.

improves the answer accuracy of all backbones, and by applying paired t-test, the improvements are statistically significant with $p \leq 0.001 \sim 0.05$. It demonstrates that LeAp can enhance the mathematical reasoning ability of MWP solvers by enabling them to autonomously learn explicit knowledge from problem solving and apply it to generate answers.

Second, to assess the effect of knowledge application, we design a variation of LeAp, named "LeAp-EK", by directly applying knowledge $Z$ from external knowledge bases in Knowledge Decoder. We select HowNet (Dong and Dong 2006) as the knowledge base for Math23K and ConceptNet (Speer, Chin, and Havasi 2017) for MAWPS and SVAMP. As we can see, the performances of "LeAp-EK" are also significantly better than "ORI". It shows the applicability of Knowledge Decoder that applies knowledge for different solvers and further verifies the reasonability of our "Effective knowledge" assumption in Section 4.

Third, "LeAp" outperforms "LeAp-EK" in almost all cases. That is probably because LeAp learns additional word-operator knowledge and our theoretical results guarantee that effective knowledge for MWP can be acquired. Thus, it reflects the importance and benefits of LeAp's learning mechanism to gain knowledge autonomously.

**Ablation Study.** Here, we highlight the advantages of our semantics-enhanced module in Eq. (7) and reasoning-enhanced module in Eq. (9) for knowledge application. Combing Table 1 and Table 2, the accuracy of reducing semantics-enhanced ("w/o SE") or reasoning-enhanced ("w/o RE") degrades for all models, proving that they are necessary for LeAp to solve MWP while also having superior generality. Besides, the performances of LeAp "w/o SE" are higher than "w/o RE" for semantics-focused methods (e.g., Graph2Tree). It indicates that the learned knowledge may be more useful to reason symbols for methods that have captured strong semantics. The opposite results appear for reasoning-focused solvers (e.g., TSN-MD). Thus, these two modules are suitable for different types of backbones.

| LeAp (backbone) | word-word pairs | word-operator pairs |
|---|---|---|
| Seq2Seq | house-home<br>egg-food<br>add-all | total-"$+$"<br>selling-"$-$"<br>pieces-"$\times$" |
| Graph2Tree | apple-fruit<br>sale-buy<br>give-hold | more-"$+$"<br>times-"$\times$"<br>gave-"$-$" |
| HMS | person-people<br>more-total<br>more-another | leftover-"$+$"<br>other-"$+$"<br>add-"$+$" |
| GTS | potato-food<br>apple-fruit<br>piece-part | earned-"$+$"<br>rate-"$\div$"<br>added-"$+$" |
| TSN-MD | per-each<br>store-sale<br>red-blue | give-"$-$"<br>costs-"$-$"<br>borrowed-"$+$" |

Table 3: Top 3 knowledge (word-word/operator pairs) of LeAp with different backbones on MAWPS.

## 5.3 LeAp Analysis

**Knowledge learning.** Now, we assess the quality of explicit knowledge $Z$ learned by LeAp. Our idea is to rank $ww_{i,j}, wo_{i,c} \in Z$ after training and expect the reasonable one to be at the top. We visualize the top 3 word-word/operator pairs in Table 3 and clearly see that LeAp gains reasonable knowledge with different backbones (e.g., LeAp (Seq2Seq) learns that "house" is related to "home" and "total" is related to "$+$").

Further, we conduct a quantitative evaluation by introducing $\alpha = 20\%$ of external knowledge bases (i.e., HowNet and ConceptNet) as the prior for word-word knowledge $ww_{i,j}$ (there is no external knowledge base for $wo_{i,c}$) and calculating $Precision@50$ based on the rest $80\%$, because we fo-

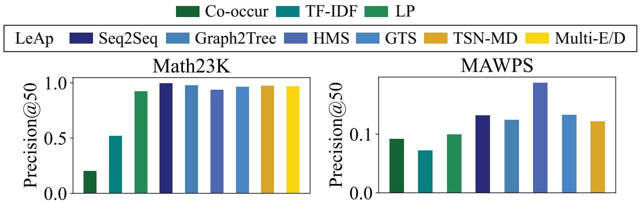


Figure 3: $Precision@50$ of word-word relations $ww_{i,j}$.

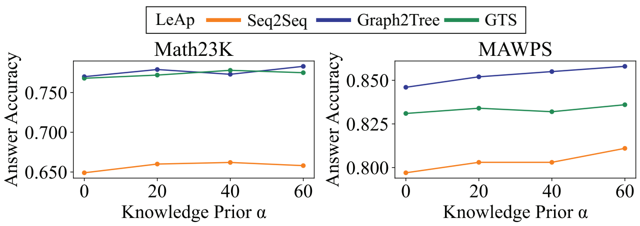


Figure 4: Answer accuracy with $\alpha = 0\%, 20\%, 40\%, 60\%$.

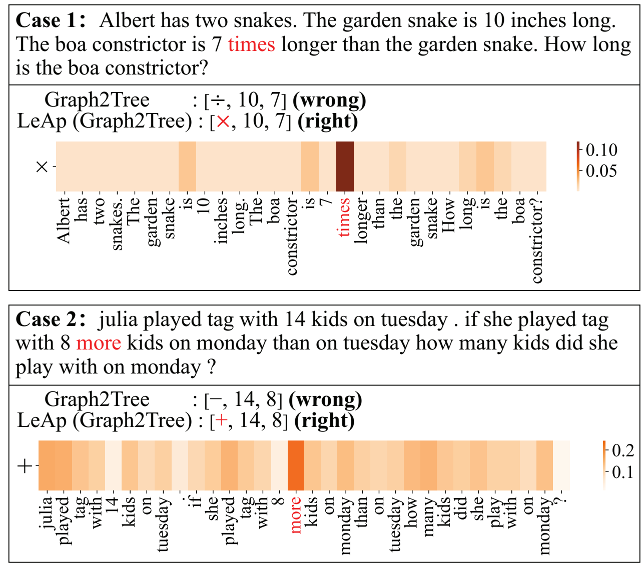


Figure 5: Case study.

cus more on the knowledge accuracy. Here, We introduce three classic baselines. "Co-occur" determines the weight between two words by their co-occurrence frequency in all problems. "TF-IDF" calculates the cosine similarity between words by their TF-IDF values. "LP" is a model with the same architecture as Knowledge Encoder. It is trained on the same $20\%$ knowledge in the prior of LeAp.

From Figure 3, our LeAp performs better than all baselines, no matter taking which backbone solver. It reflects the superiority and robustness of LeAp's autonomous learning mechanism. Especially, LeAp outperforms LP, justifying the rationality of our theoretical analyses in Section 4.

**Knowledge prior.** To simulate learners with different backgrounds, we take $\alpha = 0\%, 20\%, 40\%, 60\%$ of external knowledge bases in the prior $P(Z)$ and evaluate the performance of LeAp on MWP solving, with Seq2Seq, Graph2Tree, and GTS as the backbone. From Figure 4, with the increase of $\alpha$, the answer accuracy of LeAp shows an increasing trend, just as a human learner with a richer knowledge foundation can better carry out mathematical reasoning. Besides, when $\alpha = 0\%$, LeAp still outperforms the original backbones ("ORI") in Table 1. It demonstrates the flexibility of our LeAp to learn knowledge from scratch.

**Case Study.** Further, we conduct case study to illustrate the interpretable reasoning process of LeAp (Graph2Tree as the backbone). In Figure 5, we report the problem sentence, the prefix expressions reasoned by Graph2Tree and LeAp (Graph2Tree), and the word-operator knowledge $wo_{i,c}$ learned by LeAp. For both cases, we can see that the original Graph2Tree makes mistakes at reasoning step $t = 1$. Comparatively, by applying the learned knowledge $wo_{i,c}$ between "times" and "$\times$", "more" and "$+$", LeAp corrects such errors and reasons "$\times$" and "$+$" accurately. Notably, the knowledge $wo_{i,c}$ is fixed during the reasoning process. When to use this knowledge is controlled by how focused LeAp is on the words "times" and "more", measured by $ws_i^t$ in our proposed reasoning-enhanced module (Eq. (8) in Section 3.2). Therefore, we can conclude that in LeAp, not only does the learned explicit knowledge benefit existing MWP solvers on answer accuracy, but the knowledge application mechanism can also explain how to reason the corresponding answers, showing superior interpretability.

## 6 Conclusion and Future Work

In this paper, we proposed a Learning by Applying (LeAp) framework for explicit knowledge learning and applying in a novel general *problem-knowledge-expression* paradigm that can be added to existing solvers to improve their reasoning ability. We also designed semantics/reasoning-enhanced modules in LeAp to strengthen problem understanding and symbol reasoning by applying knowledge effectively. We theoretically proved the superiority of LeAp's autonomous learning mechanism from a Bayesian perspective. Experiments showed the effectiveness of LeAp in answer reasoning, knowledge learning, and interpretability. In the future, we will extend LeAp to other types of knowledge/problems, and explore its potential to enrich external knowledge bases.

**Acknowledgement.** This research was partially supported by grants from the National Natural Science Foundation of China (Grants No. 62106244, and 61922073).

## References

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

Bansal, T.; Juan, D.-C.; Ravi, S.; and McCallum, A. 2019. A2N: Attending to neighbors for knowledge graph inference. In *Proceedings of the 57th annual meeting of the association for computational linguistics*, 4387–4392.

Bordes, A.; Usunier, N.; Garcia-Duran, A.; Weston, J.; and Yakhnenko, O. 2013. Translating embeddings for modeling multi-relational data. *Advances in neural information processing systems*, 26.

Cai, L.; and Ji, S. 2020. A multi-scale approach for graph link prediction. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, 3308–3315.

Cao, Y.; Hong, F.; Li, H.; and Luo, P. 2021. A Bottom-Up DAG Structure Extraction Model for Math Word Problems. In *AAAI*, volume 35, 39–46.

Chen, J.; He, H.; Wu, F.; and Wang, J. 2021. Topology-aware correlations between relations for inductive link prediction in knowledge graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 6271–6278.

Cheng, K.; Yang, Z.; Zhang, M.; and Sun, Y. 2021. UniKER: A Unified Framework for Combining Embedding and Definite Horn Rule Reasoning for Knowledge Graph Inference. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 9753–9771.

Dai, W.-Z.; and Muggleton, S. H. 2021. Abductive knowledge induction from raw data. In *IJCAI*, 1845–1851.

de Penning, H. L. H.; Garcez, A. S. d.; Lamb, L. C.; and Meyer, J.-J. C. 2011. A neural-symbolic cognitive agent for online learning and reasoning. In *Twenty-Second International Joint Conference on Artificial Intelligence*.

Dong, Z.; and Dong, Q. 2006. *Hownet and the Computation of Meaning*. World Scientific.

Feigenbaum, E. A.; Feldman, J.; et al. 1963. *Computers and thought*. New York McGraw-Hill.

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

Hosseini, M. J.; Hajishirzi, H.; Etzioni, O.; and Kushman, N. 2014. Learning to solve arithmetic word problems with verb categorization. In *EMNLP*, 523–533. Citeseer.

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

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

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

Jang, E.; Gu, S.; and Poole, B. 2016. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*.

Jie, Z.; Li, J.; and Lu, W. 2022. Learning to Reason Deductively: Math Word Problem Solving as Complex Relation Extraction. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*, 5944–5955.

Kim, H.; Hwang, J.; Yoo, T.; and Cheong, Y.-G. 2022. Improving a Graph-to-Tree Model for Solving Math Word Problems. In *2022 16th International Conference on Ubiquitous Information Management and Communication (IMCOM)*, 1–7. IEEE.

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

Kingma, D. P.; and Welling, M. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Kipf, T.; Fetaya, E.; Wang, K.-C.; Welling, M.; and Zemel, R. 2018. Neural relational inference for interacting systems. In *International Conference on Machine Learning*, 2688–2697. PMLR.

Koncel-Kedziorski, R.; Hajishirzi, H.; Sabharwal, A.; Etzioni, O.; and Ang, S. D. 2015. Parsing algebraic word problems into equations. *Transactions of the Association for Computational Linguistics*, 3: 585–597.

Labhishetty, S.; Zhai, C.; Xie, M.; Gong, L.; Sharnagat, R.; and Chembolu, S. 2022. Differential Query Semantic Analysis: Discovery of Explicit Interpretable Knowledge from E-Com Search Logs. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*, 535–543.

Lan, Y.; Wang, L.; Zhang, Q.; Lan, Y.; Dai, B. T.; Wang, Y.; Zhang, D.; and Lim, E.-P. 2022. Mwptoolkit: an open-source framework for deep learning-based math word problem solvers. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, 13188–13190.

Lewkowycz, A.; Andreassen, A.; Dohan, D.; Dyer, E.; Michalewski, H.; Ramasesh, V.; Slone, A.; Anil, C.; Schlag, I.; Gutman-Solo, T.; et al. 2022. Solving Quantitative Reasoning Problems with Language Models. *arXiv preprint arXiv:2206.14858*.

Liang, X.; Hu, Z.; Zhang, H.; Lin, L.; and Xing, E. P. 2018. Symbolic graph reasoning meets convolutions. *Advances in Neural Information Processing Systems*, 31.

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

Lin, X.; Huang, Z.; Zhao, H.; Chen, E.; Liu, Q.; Wang, H.; and Wang, S. 2021. Hms: A hierarchical solver with dependency-enhanced understanding for math word problem. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 4232–4240.

Liu, L.; Du, B.; Ji, H.; Zhai, C.; and Tong, H. 2021. Neural-Answering Logical Queries on Knowledge Graphs. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 1087–1097.

Liu, Q.; Huang, Z.; Yin, Y.; Chen, E.; Xiong, H.; Su, Y.; and Hu, G. 2019. Ekt: Exercise-aware knowledge tracing for student performance prediction. *IEEE Transactions on Knowledge and Data Engineering*, 33(1): 100–115.

Luong, M.-T.; Pham, H.; and Manning, C. D. 2015. Effective Approaches to Attention-based Neural Machine Translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 1412–1421.

Mitra, A.; and Baral, C. 2016. Learning to use formulas to solve simple arithmetic problems. In *Proceedings of the*

*54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2144–2153.

Mowrer, O. 1960. *Learning theory and behavior.* John Wiley & Sons Inc.

Muhajirah, M. 2020. Basic of Learning Theory: (Behaviorism, Cognitivism, Constructivism, and Humanism). *International Journal of Asian Education*, 1(1): 37–42.

Patel, A.; Bhattamishra, S.; and Goyal, N. 2021. Are NLP Models really able to Solve Simple Math Word Problems? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2080–2094.

Pei, H.; Wei, B.; Chang, K.; Zhang, C.; and Yang, B. 2020. Curvature regularization to prevent distortion in graph embedding. *Advances in Neural Information Processing Systems*, 33: 20779–20790.

Pei, H.; Wei, B.; Chang, K. C.-C.; Lei, Y.; and Yang, B. 2019. Geom-GCN: Geometric Graph Convolutional Networks. In *International Conference on Learning Representations*.

Petroni, F.; Rocktäschel, T.; Riedel, S.; Lewis, P.; Bakhtin, A.; Wu, Y.; and Miller, A. 2019. Language Models as Knowledge Bases? 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)*, 2463–2473.

Peyrard, M.; and West, R. 2020. KLearn: Background Knowledge Inference from Summarization Data. In *EMNLP (Findings)*.

Qu, M.; and Tang, J. 2019. Probabilistic logic neural networks for reasoning. *Advances in neural information processing systems*, 32.

Roy, S.; and Roth, D. 2017. Unit dependency graph and its application to arithmetic word problem solving. In *AAAI*, volume 31.

Seo, M.; Hajishirzi, H.; Farhadi, A.; Etzioni, O.; and Malcolm, C. 2015. Solving geometry problems: Combining text and diagram interpretation. In *EMNLP*, 1466–1476.

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

Shen, J.; Yin, Y.; Li, L.; Shang, L.; Jiang, X.; Zhang, M.; and Liu, Q. 2021. Generate & Rank: A Multi-task Framework for Math Word Problems. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, 2269–2279.

Shen, Y.; and Jin, C. 2020. Solving math word problems with multi-encoders and multi-decoders. In *Proceedings of the 28th International Conference on Computational Linguistics*, 2924–2934.

Shi, S.; Wang, Y.; Lin, C.-Y.; Liu, X.; and Rui, Y. 2015. Automatically solving number word problems by semantic parsing and reasoning. In *EMNLP*, 1132–1142.

Speer, R.; Chin, J.; and Havasi, C. 2017. Conceptnet 5.5: An open multilingual graph of general knowledge. In *Thirty-first AAAI conference on artificial intelligence*.

Tsatsou, D.; Karageorgos, K.; Dimou, A.; Carbó, J.; López, J. M. M.; and Daras, P. 2021. Towards Unsupervised Knowledge Extraction. In *AAAI Spring Symposium: Combining Machine Learning with Knowledge Engineering*.

von Rueden, L.; Mayer, S.; Beckh, K.; Georgiev, B.; Giesselbach, S.; Heese, R.; Kirsch, B.; Walczak, M.; Pfrommer, J.; Pick, A.; et al. 2021. Informed Machine Learning-A Taxonomy and Survey of Integrating Prior Knowledge into Learning Systems. *IEEE Transactions on Knowledge & Data Engineering*, (01): 1–1.

Wang, L.; Zhang, D.; Zhang, J.; Xu, X.; Gao, L.; Dai, B. T.; and Shen, H. T. 2019. Template-based math word problem solvers with recursive neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 7144–7151.

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

Wu, Q.; Zhang, Q.; Fu, J.; and Huang, X.-J. 2020. A knowledge-aware sequence-to-tree network for math word problem solving. In *EMNLP*, 7137–7146.

Wu, Q.; Zhang, Q.; and Wei, Z. 2021. An edge-enhanced hierarchical graph-to-tree network for math word problem solving. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, 1473–1482.

Xia, L.; Huang, C.; Xu, Y.; Dai, P.; Zhang, X.; Yang, H.; Pei, J.; and Bo, L. 2021. Knowledge-enhanced hierarchical graph transformer network for multi-behavior recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 4486–4493.

Xie, Z.; and Sun, S. 2019. A Goal-Driven Tree-Structured Neural Model for Math Word Problems. In *IJCAI*, 5299–5305.

Yu, W.; Wen, Y.; Zheng, F.; and Xiao, N. 2021. Improving Math Word Problems with Pre-trained Knowledge and Hierarchical Reasoning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 3384–3394.

Zhang, D.; Wang, L.; et al. 2020. 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.

Zhang, J.; Lee, R. K.-W.; Lim, E.-P.; Qin, W.; Wang, L.; Shao, J.; and Sun, Q. 2020a. Teacher-student networks with multiple decoders for solving math word problem. In *IJCAI*.

Zhang, J.; Wang, L.; Lee, R. K.-W.; et al. 2020b. Graph-to-Tree Learning for Solving Math Word Problems. In *Association for Computational Linguistics*, 3928–3937.

Zhou, H.; Young, T.; Huang, M.; Zhao, H.; Xu, J.; and Zhu, X. 2018. Commonsense knowledge aware conversation generation with graph attention. In *IJCAI*, 4623–4629.