Complex & Intelligent Systems (2023) 9:697–717
https://doi.org/10.1007/s40747-022-00828-0

ORIGINAL ARTICLE

Solving arithmetic word problems by synergizing syntax-semantics extractor for explicit relations and neural network miner for implicit relations

Xinguo Yu[1] · Xiaopan Lyu[1] · Rao Peng[1] · Jun Shen[2]

Received: 21 December 2021 / Accepted: 7 July 2022 / Published online: 29 July 2022

Abstract
This paper presents a relation-centric algorithm for solving arithmetic word problems (AWPs) by synergizing a syntax-semantics extractor for extracting explicit relations, and a neural network miner for mining implicit relations. This is the first algorithm that has a specific component to acquire implicit knowledge items for solving AWPs. This paper proposes a three-phase scheme to decompose the challenging task of designing an algorithm for solving AWPs into three smaller tasks. The first phase proposes a state-action paradigm; the second phase instantiates the paradigm into a relation-centric approach; and the third phase implements a relation-centric algorithm for solving AWPs. There are two main steps in the proposed algorithm: problem understanding and symbolic solver. By adopting the relation-centric approach, problem understanding becomes a task of relation acquisition. For conducting the task of relation acquisition, a relaxed syntax-semantics method first extracts a group of explicit relation candidates. In parallel, a neural network miner acquires implicit relation candidates. The miner computes the vectors encoded by BERT to determine which implicit relations should be added. Thus, problem understanding can acquire both explicit relations and implicit relations, which addresses the challenge of building a problem understanding method that can acquire all the knowledge items to find the solution. In the subsequent step of symbolic solver, a fusion procedure forms a distilled set of relations from all the candidates by discarding unnecessary relations. Experimentation on nine benchmark datasets validates the superiority of the proposed algorithm that outperforms the state-of-the-art algorithms.

Keywords Problem solving · State-action paradigm · Relation-centric · Explicit relation · Implicit relation · Syntax-semantics model

Introduction

Problem solving for basic education has become a hotspot research problem again in the recent years because of the confluence of the application demand and the stimulation by artificial intelligence (AI) rapid progress [1,2], in which solving the arithmetic word problems (AWPs) is the most active research problem. The success of "AlphaGo" further enforces the belief that AI will have the ability to solve the exercise problems in an intelligence way like human being does. Besides being the demand technology of building intelligent tutoring systems, algorithms for solving AWPs are useful technologies in fact-checking tasks to validate the veracity of a given claim [3]. In theory, it is still challenging for AI as it cannot properly solve AWPs yet. Now the problem of solving AWPs is complicated and tricky. On the one hand, it seems not to be a difficult research problem because people and even young kids can often solve AWPs with ease. In another hand, as two recent survey papers have also pointed out, so far there are not many satisfactory solvers after over 60 years of efforts [1,2]. The theoretical significance, its application demand and its research challenge, motivate our interest in developing algorithms for solving AWPs.

✉ Xinguo Yu
xgyu@ccnu.edu.cn

Xiaopan Lyu
xiaopanlv@mails.ccnu.edu.cn

Rao Peng
pr3250@mails.ccnu.edu.cn

Jun Shen
jshen@uow.edu.au

[1] Faculty of Artificial Intelligence in Education, Central China Normal University, Wuhan 430079, China

[2] School of Computing and Information Technology, University of Wollongong, Wollongong 2522, Australia

The advances in natural language processing and deep learning now provide a plethora of tools for the development of new algorithms for solving AWPs. Based on these advances, the seq2seq (sequence to sequence) deep neural networks of solving AWPs has been developed in the past years [4–13]. This seq2seq approach solves the problem using single deep neural networks and gives a final answer without intermediate steps. As a result, it cannot provide the intermediate steps of tutoring students as instructors do.

Except the algorithms in the seq2seq approach, most of the existing algorithms for solving AWPs involve two main steps: problem understanding and symbolic solver. Since the two steps run sequentially, the symbolic solver highly depends on the output of problem understanding. Hence, there has been much effort in developing problem understanding. The methods on problem understanding in the literature can be divided into four main families: semantic analysis [14–17], rule based [18–23], machine learning [24–27], and syntax-semantics models [28–33]. A common shortcoming of these four families of algorithms is that they do not have any component to discover implicit knowledge items for solving AWPs. The existing algorithms have a limited ability in solving AWPs because they do not have any effective methods to acquire such implicit knowledge items. Several studies have been involved the investigation of how implicit relations could be acquired [21,23,34], but so far it has not been studied yet how to use both explicit knowledge and implicit knowledge to form the complete knowledge body for solving AWPs. In solving AWPs, the explicit knowledge refers to the quantitative relations stated explicitly in the problem text; the implicit knowledge refers to the knowledge indispensable to solve problems added by the students based on their mastered knowledge. As an example, Fig. 1(b) shows how to add a formula "$perimeter = 2(length + width)$" to solve the given AWP, which is an implicit relation from the formula database. Hence, the implicit knowledge in this paper differs from the implicit relation triples produced by the existing relation triples as in [35,36].

This paper proposes a three-phase scheme of developing the algorithm for solving AWPs to decompose the task of developing an algorithm for solving AWPs into three phases, namely creating a state-action paradigm, instantiating the paradigm into a relation-centric approach, and implementing a solving algorithm by adopting the relation-centric approach. The state-action paradigm lies on the recognition that any algorithm of solving AWPs is a loop of the state of knowledge expression and the algorithm action of transforming states. This paper instantiates the state-action paradigm into a relation-centric approach using a group of relations as the intermediate state of the solving process. The proposed algorithm in this paper is a relation-centric algorithm for solving AWPs. The proposed approach is able to provide explainable intermediate steps, being the trait required by the intelligent tutoring systems. Following the existing algorithms, there are two main steps in the proposed algorithm: problem understanding and symbolic solver. Figure 1 shows that the pipeline and a process of solving a sample AWP by the proposed algorithm. Problem understanding is an acquirer that can acquire both explicit relations and implicit relations. The success of this acquirer lies in synergizing a relaxed syntax-semantics (S^2) extractor and a neural network miner. The symbolic solver first forms a group of relations by fusing explicit relations and implicit relations. Then it produces a distilled group of relations by discarding unnecessary relations. Next it transforms the distilled group of relations into a system of equations. The symbolic solver finally outputs the values of the unknown for a given problem by solving the system of equations. The conducted experiments evaluate the proposed algorithm on a collection of nine datasets; the experimental results show that the proposed algorithm outperforms the state-of-the-art algorithms on these datasets. To the best of our knowledge, this is the first algorithm which uses both the extractor for acquiring explicit relations and the neural network miner for mining implicit relations separately to solve more AWPs. In summary, this paper has three contributions as follows:

(1) It proposes the three-phase scheme of building algorithms for solving AWPs, which consists of building a state-action paradigm, instantiating a relation-centric approach, and implementing a relation-centric algorithm.
(2) It proposes the first relation-centric algorithm that can solve both explicit AWPs and implicit AWPs.
(3) It creates an acquirer that synergizes a relaxed S^2 extractor of acquiring explicit relations and a neural network miner of mining implicit relations.

In the rest of the paper, "Related work" first provides a review of the related work. "State-action paradigm and relation-centric approach" then builds the state-action algorithm paradigm and the relation-centric approach. "A relation–centric algorithm for solving AWPs" presents the proposed algorithm for solving AWPs. "Experiments" presents the algorithm analysis and experimental results. "Conclusions and future work" concludes the paper.

Related work

This section starts with reviewing the solving approaches of the existing algorithms because there is no related work in building the solving algorithm paradigm yet. The existing algorithms for solving AWPs are divided into five approaches according to the different expression forms of the understood state of AWPs. The algorithm proposed in this paper consists in two phases of problem understanding and symbolic solver.

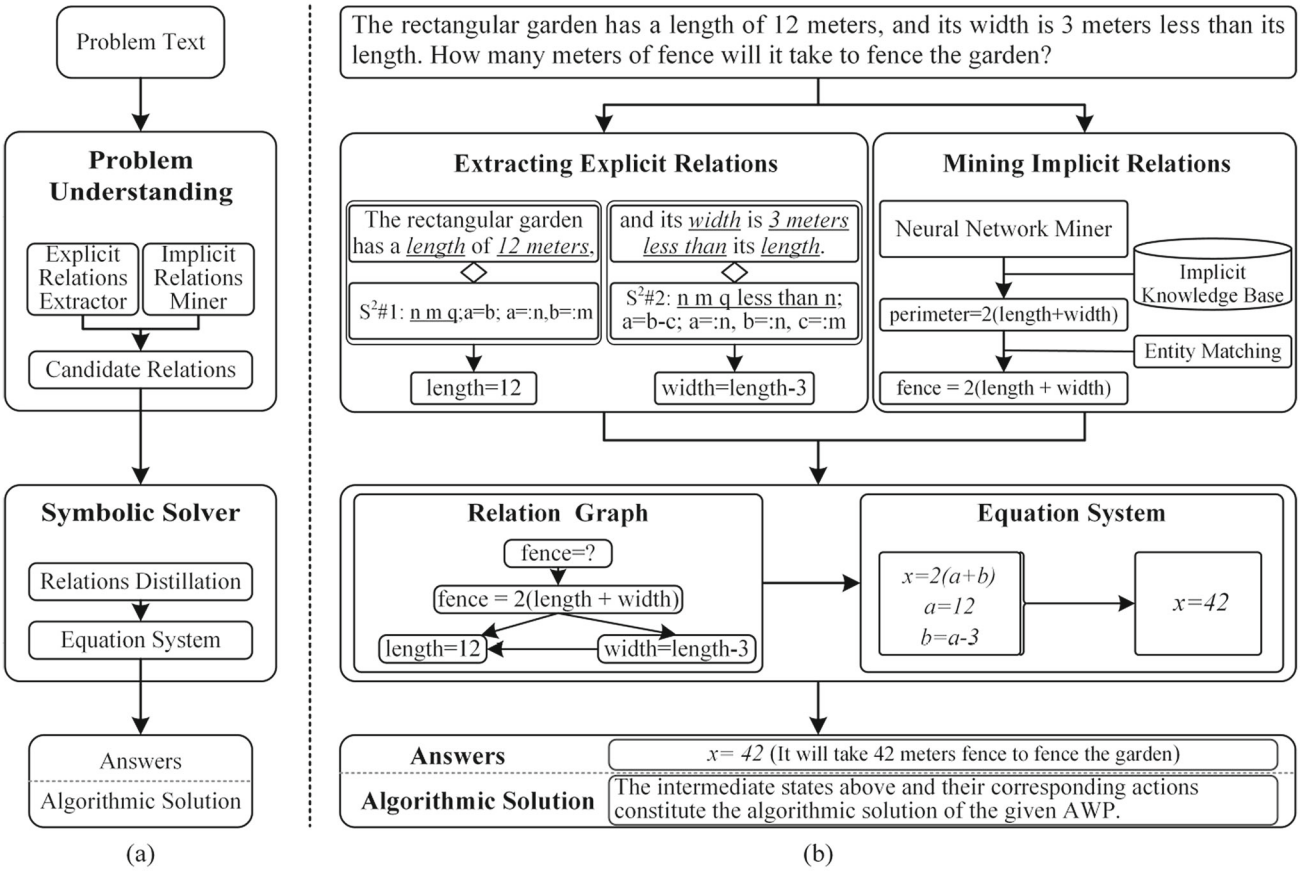

Fig. 1 The pipeline of the proposed algorithm and the process that it solves the given AWP. The pipeline of the proposed relation-centric algorithm is in **(a)**; the process that the proposed algorithm solves the given AWP is in **(b)**

Hence, the related work is also about problem understanding and symbolic solver.

The related work in solving approach

Researchers have proposed many algorithms to solve AWPs. Through analyzing the process of solving problems, one can form the following conclusion: solving problems is to present the knowledge in different states and to use actions to transform knowledge expression from a state to another state. For example, some algorithms use a system of equations as the main intermediate state of AWPs. Their main actions are to build the system and solve the system. Then, this paper proposes to mainly use the knowledge expression forms of the main states to define the approaches of solving algorithms. According to these forms, the existing algorithms can be divided into five approaches: (1) two-frame, (2) equation-centric, (3) answer expression, (4) seq2seq, and (5) relation-centric. The concepts and methods of the five approaches will be the base to build the state-action paradigm proposed in this paper. We review the five approaches as follows.

(1) **Two-frame**: Back in 1985, Kintsch et al. [14] proposed a two-frame approach to solve AWPs. This approach uses two types of frames to store the outcome of problem understanding, namely knowledge frames and solution frames. This approach addressed the representation of outcome of problem understanding and the scheme of knowledge inference. It uses the knowledge frames to store the extracted knowledge items by problem understanding; the solution frames to store the inference schemes. It uses sentence templates to extract knowledge from problem text in order to fill abstract knowledge frames. Subsequently, the researchers proposed semantic sentence templates and deep neural networks to extract knowledge items from problem text [11,15–17]. However, the two-frame approach suffers from a couple of limitations. The first one is that the number of sentence templates can become quickly large due to the large variety of expressions for a given semantic meaning. The other one is that there is no uniform way to infer knowledge frames according to solution frames.

(2) **Equation-centric**: Equation-centric approach is to solve AWPs with methods around equations. Since the math-

ematical tools can solve the system of equations, the main phase of this approach is *problem understanding*, i.e., how to form such a system of equations. Hence, many papers studied various methods to form system of equations. Some methods acquire equations one by one [18,37–41]. Some other methods acquire all equations using equations templates. Kushman et al. [24] first proposed machine learning (ML) method to acquire all the equations together. The ML method prepares a pool of equations templates. The trained ML finds out which template is suitable for the given problem followed by matching entities with elements of the equation template. The ML methods was further studied in [25–27,42]. However, both ways have the limitation of acquiring the proper group of equations. The way of acquiring equations one by one, encounters the difficulty in assigning the same variable to the same entity because the same entity may have different appearances in different equations. The way of acquiring equations using equations templates, encounters difficulty in building a pool of equation templates that can achieve high problem accuracy. Another demerit of the equation-centric approach is that most of the algorithms in this approach have a limited scope because they can produce only linear equations [18,24–27,37,38,41,42].

(3) **Answer Expression**: The approach of answer expression is to solve the problem by forming the answer expression from the extracted knowledge items. An answer expression is an expression of numbers and operands; the answer is the result of evaluating this expression with the assumption that a problem requires a single answer. The algorithms in this approach used various methods to extract knowledge items and form the answer expression. Roy et al. [20] first proposed to use quantity schema and multiple classifiers to map problem text to answer expression. Subsequently, some researchers proposed to use unit dependency graph [21], formula templates [22], or declarative rules [23] to extract knowledge items for forming answer expressions. In the recent years, some researchers proposed to use deep reinforcement learning [43] or deep neural networks [44,45] to form answer expressions. *Answer expression* is an indispensable state of this approach and the methods to extract knowledge items and form this expression are the corresponding actions. However, such actions cross a big span from input to answer expression so they are complicated. Thus, answer expression carries two drawbacks: (i) it can only solve single unknown problems so far; (ii) it can hardly explain the internal reason or inference of the expression generation.

(4) **Seq2Seq**: The seq2seq approach is to solve the problem using the trained machine to obtain the sequence of answer expression, which assumes that a problem requires a single answer so far. This approach is different from the approach of answer expression lying in that it mainly computes vectors, although both approaches target to acquire answer expressions. It first converts problem text into vector sequence and then uses the trained machine to accomplish a transformation from vector sequence of problem text into vector sequence of answer expression. Some papers call this approach as data-driven approach because this approach needs data to train learning machines [46,47]. Wang et al. [4] first proposed a seq2seq approach to design algorithms for solving AWPs. Subsequently, researchers proposed more deep neural networks to translate problem text into answer expression [5–10,12,13,48]. The seq2seq approach suffers a demerit that the actions taken by deep neural networks are not explainable.

(5) **Relation-centric**: After learning from the above-listed four approaches, Yu et al. [28] proposed a relation-centric approach to solve AWPs and proposed to use S^2 method to extract relations from problem text. More researchers paid attention to this approach because it has multiple merits. First, it inherits the merit that the actions are explainable to students because it adopts model-based method. Second, it also demonstrates the advantage of equations because it is easy to convert relations into equations. Third, acquiring relations is easier than acquiring equations. Acquiring equations one by one by S^2 method proves to be a much easier task than directly acquiring a batch of equations. The relation-centric approach was successfully applied in understanding plane geometry problems and direct current circuit problems [31–33], but it has a limited performance when solving AWPs as it cannot acquire implicit relations yet [29], which we have tried to tackle further in this paper.

In a word, this paper adopts the relation-centric approach to develop an algorithm that can broaden the range of solving arithmetic word problems and output explainable solutions that can provide tutoring service to students. Such an algorithm is also applicable to fact-checking task, which validates the veracity of a given claim [3], because it can output an algorithmic solution and a series of algorithmic actions from one state to another. In contrast, the algorithms of adopting some other approaches, especially seq2seq, do not have such ability because they cannot give understandable intermediate results.

The related work in problem understanding

There are four main methods for problem understanding: template matching, rule searching, machine learning, and S^2 model.

The method of template matching uses a pool of templates to extract knowledge items from problem text [14]. A large number of templates are needed because templates are semantic [15]. The rule searching is to search and identify the various scenarios to produce equations. This method also needs a large number of rules because there are many different scenarios [18,23]. The machine learning method leverages the designed features to search the most probable equation template [24,25,27]. This method can hardly explain the internal reason of the equation template selection. Researchers proposed the S^2 model method to work against the variety of semantics and scenarios [28,29]. The S^2 model method has overcome the dimension disaster of semantic expressions by increasing the use of syntax patterns and reducing the use of semantic patterns. However, these four methods mainly extract the *explicit* relations, though they can acquire some *implicit* relations. Hence, this paper develops a method to acquire more implicit relations.

Here we review the methods involving in acquiring implicit relations. In 2016, two studies attempted to address the problem of implicit relation acquisition. The first study, done in Mitra et al. [22], relies on a machine learning method to find equations for three mathematical situations, i.e., 1) part whole, 2) quantity change, and 3) quantity comparison. Among these three situations, only the first one refers to a type of implicit relations as problem text normally does not explicitly state the relation of part whole. The second study, done in Yu et al. [49], focuses on adding an additional formula into outcome of problem understanding. Such a formula corresponds to an implicit relation, as it essentially is a knowledge point requiring students to add an appropriate formula when they are solving problems. To test how well students master this knowledge point, problem text should not explicitly state it. However, problem text expressions will often provide hints of adding the required formula. Based on this fact, Yu et al. [49] proposed an SVM based method to find additional formulas to be used in solving AWPs. The proposed method is not cutting-edge because it relies on bag-of-words rather than vector features. In another study, Dewappriya et al. [34] proposed an ontology-based unit system to resolve unit conflicts. The system mainly aimed at preventing unit conflict before running the main portion of solving algorithm and thus acted as a preprocessing step.

Learning from the existing methods, this study proposes an algorithm for solving AWPs. The algorithm uses a relaxed S^2 method to extract explicit relations and a neural network to acquire implicit relations. More specifically, the proposed algorithm relies on a BERT based neural network to find implicit relations. The phase of problem understanding thus essentially changes to acquiring a group of relations while the symbolic solver conducts an equivalent transform of relations. This is the first algorithm that can solve a large number of AWPs involving both explicit and implicit relations.

The related work in symbolic solver

Researchers put less effort in developing symbolic solvers because it is not difficult to solve the outcome of problem understanding in most cases. For the algorithms of adopting the equation-centric approach, the outputs of their problem understanding are systems of linear equations [18,24–27,37,38,41,42]. Hence, their symbolic solvers are somewhat simplistic because Gaussian elimination can solve such systems. For the algorithms of adopting the approach of answer expression, the outputs of their problem understanding are in various formats of arithmetic expressions so that they do not need any complex symbolic solver [20–23,43–45]. The algorithms adopting approach of two-frame used the heuristic ways to find answers [11,14–17]. The algorithms of adopting the relation-centric approach have a decent symbolic solver [28,30,32]. The reason is that the equations produced from relations might not be linear. Hence, Yu et al. [29] proposed a method leveraged on the idea of recursively solving the linear portion of all equations. This paper will continue to use this method to build the symbolic solver.

State-action paradigm and relation-centric approach

This paper proposes a three-phase scheme to develop algorithms for solving AWPs, which is a scheme to design algorithms from abstract to concreteness. This section presents the first two phases. The first phase is to design a solving paradigm. Concretely, it proposes a state-action paradigm for solving AWPs as solving paradigm. The second phase is to define the states of the state-action paradigm to form a relation-centric approach to solving AWPs.

The goal of this paper is to design high performance algorithms for solving AWPs. Through analyzing the algorithms in the literature, the loop of ***states*** and ***actions*** can explain the process of solving AWPs by algorithms. Hence, this paper proposes a state-action paradigm as depicted in Fig. 2 and uses it as the algorithm paradigm of solving AWPs. In this paradigm, the states are different states of knowledge expression and actions are the algorithm actions to transform from a state to another. This abstract solving paradigm is a simple diagram, but it can guide us to understand what the solving algorithms are doing and what the core issues of solving problems are. It also can explain how we design approaches and algorithms for solving AWPs. More importantly, the approach of solving AWPs can be defined based on the abstract solving paradigm.

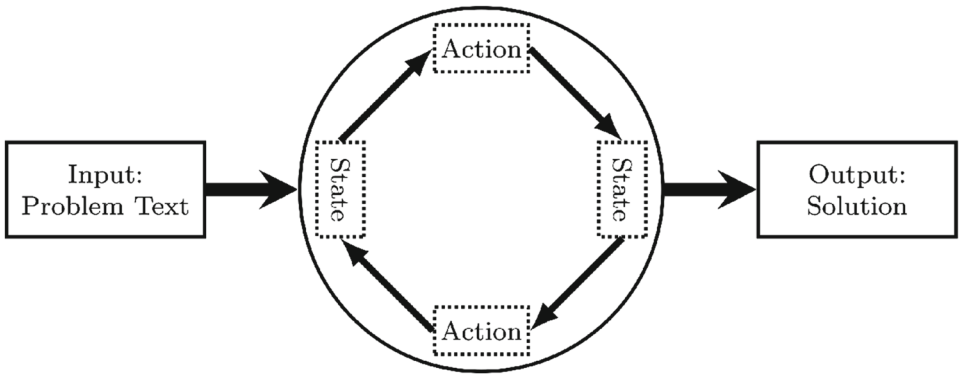

Fig. 2 The state-action paradigm of solving AWPs

Definition 1 (*Approach*) An approach is an instance of the state-action paradigm by determining the main states of the state-action paradigm and the links to change from a state to another.

"The related work in solving approach" has classified the existing algorithms in the literature into five approaches. We take two approaches as examples to show that all the five approaches conform to Definition 1. The first approach is seq2seq. The algorithms in this approach have two related states: a vector sequence and the sequence of answer expression. Thus, they have three common links. The first link is the encoding step to encode the given problem into a vector sequence. The second link is to transform a vector sequence of problem text into a vector sequence of answer expression. The third link is to evaluate answer expression and output the answer. The second approach is relation-centric. The algorithms in this approach have one common state, which is a group of relations. Thus, they have two common links. The first link is to acquiring a group of relations, being another kind of problem understanding. The second link is to solve the group of relations.

Building a link from one state to another means that researchers have found methods to accomplish this state transformation at least for a corpus of problems. Along this state-action solving paradigm, designing algorithms means to propose methods to implement the links of the adopted approach.

All the states in the state-action solving paradigm are the states equivalent to the given problem in terms of solving AWPs. Among these states, researchers want to acquire an *understood state*, which is defined as follows.

Definition 2 (*Understood State*) A state is called an understood state of a given AWP if a symbolic solver can produce the solution from this state without revisiting the given problem.

From Definition 2, all the states except the input state and the vector sequence of problem text in the five approaches are understood states.

Definition 3 (*Problem Understanding*) Problem understanding is to produce an understood state of AWPs.

Definition 4 (*Relation*) A relation is an equation expression of quantifiable entity, where a quantifiable entity can be a number, a variable, or a phrase that describes a quantity.

A relation differs from an equation in that its items may be phrases except variables and numbers. In this paper, a relation is actually a quantity relation because all its elements are about quantity.

Proposition 1 *Assume that $\mathcal{C}$ is a corpus of AWPs. Let $\mathcal{P}$ be an AWP in $\mathcal{C}$. Then there is a group of relations denoted as $Œ = \{r_i : i = 1 \text{ to } k\}$ such that it is an understood state of $\mathcal{P}$.*

A theoretical proof for Proposition 1 is not available so far, but the experimental results in "Experiments" will testify its correctness. Actually, Proposition 1 is the assumption of many existing algorithms for solving AWPs.

We define the relation-centric approach formally because it is a central term of this paper.

Definition 5 (*Relation-Centric Approach*) A relation-centric approach is an instance of the state-action paradigm such that: 1) it has two links of problem understanding and symbolic solver; 2) the link of problem understanding is to produce a group of relations that is an understood state of a problem; 3) the link of the symbolic solver is to find values of the unknowns in the given problem through transforming the group of relations.

Definition 5 shows that relation acquisition and relation transformation are the main operations of the relation-centric approach. This approach benefit from the fact that relation extraction from text is more tractable than equation extraction.

A relation-centric algorithm for solving AWPs

This section presents a relation-centric algorithm for solving AWPs by implementing the relation-centric approach.

Algorithm outline

This section instantiates the relation-centric approach into a three-step algorithm to solve AWPs. The first step is to acquire a group of relations. The second step is a symbolic solver, which solves the group of relations. The last step is to output the unknowns with found values and an algorithmic solution, which is a series of actions of acquiring and transforming relations, recorded along the algorithm execution. These three steps constitute the proposed algorithm, which is shown in Algorithm I. The proposed algorithm uses five procedures, namely Procedure I to V, to implement the five tasks of the algorithm. The process of solving problems by Algorithm I is explainable because people can understand all the actions of acquiring and transforming relations. More importantly, this series of actions can instruct students how the algorithm solves the problem. Figure 1(b) shows the process that Algorithm I solves a given AWP to demonstrate how Algorithm I works.

Algorithm I: The Algorithm for Solving AWPs.

Input: An AWP in text format.

Output: Answers and an algorithmic solution.

Step 1: (acquiring relations) Use Procedure I and II to acquire explicit and implicit relations separately;

Step 2: (symbolic solver)

- **Step 2.1:** Use Procedure III to fuse the acquired relation candidates by Procedure I and II;
- **Step 2.2:** Use Procedure IV and V to solve the group of relations of the given problem;

Step 3: (outputting) Output the unknowns of the problem with their values and an algorithmic solution, the series of algorithmic actions from a state to another, recorded along the algorithm execution; terminate.

Procedure I: Extracting Explicit Relations from Problem Text.

Input: Problem text T.

Output: A group of explicit relations.

Step 1: Load $\Sigma = \{M_i \mid i = 1, 2, ..., m\}$; Initialize Γ as empty;

Step 2: Annotate T, namely parse T into phrases and label each phrase with a POS;

Step 3: **for** $i = 1$ **to** m **do**

M_i match each of the portions of T; For each matched portion, use the entities in the text to instantiate the elements in the model, then put an instance of R_i of M_i into Γ.

Acquiring relations of AWPs

This section presents the method details of acquiring relations from AWPs. It comprises two aspects: (1) extracting explicit relations, and (2) acquiring implicit relations.

Definition 6 (*Explicit and Implicit Relation*) Let $\mathbb{R}$ be a set of relations that is an understood state of a given AWP. A relation in $\mathbb{R}$ is an explicit relation if it explicitly states in problem text; otherwise, it is an implicit relation.

Extracting explicit relations using relaxed S^2 method

This section designs a relaxed S^2 method by enhancing the S^2 method. Compared with the original S^2 method, the relaxed S^2 method relaxes the requirement that each piece of text can match only one S^2 model and delays the action of judging whether an extracted relation is used by the coming symbolic solver. Procedure I uses the proposed relaxed S^2 method to extract explicit relations. The syntax portions of S^2 models are comprised of patterns of POS (part-of-speech) and punctuation while the semantic portions are keyword structures. Procedure I can work only after a pool of S^2 models are prepared appropriately for a natural language.

Definition 7 (S^2 *Model*) A syntax-semantics model, shorted as S^2 model, is defined as a quadruple $M = (K, P, Q; R)$, where K represents semantics keyword structures, P represents POS, Q represents the punctuation, and R is the relation as model output. Let $\Sigma = \{M_i = (K_i, P_i, Q_i; R_i) \mid i = 1, 2, ..., m\}$ denote the set of all the prepared S^2 models, called as a pool of S^2 models for AWPs.

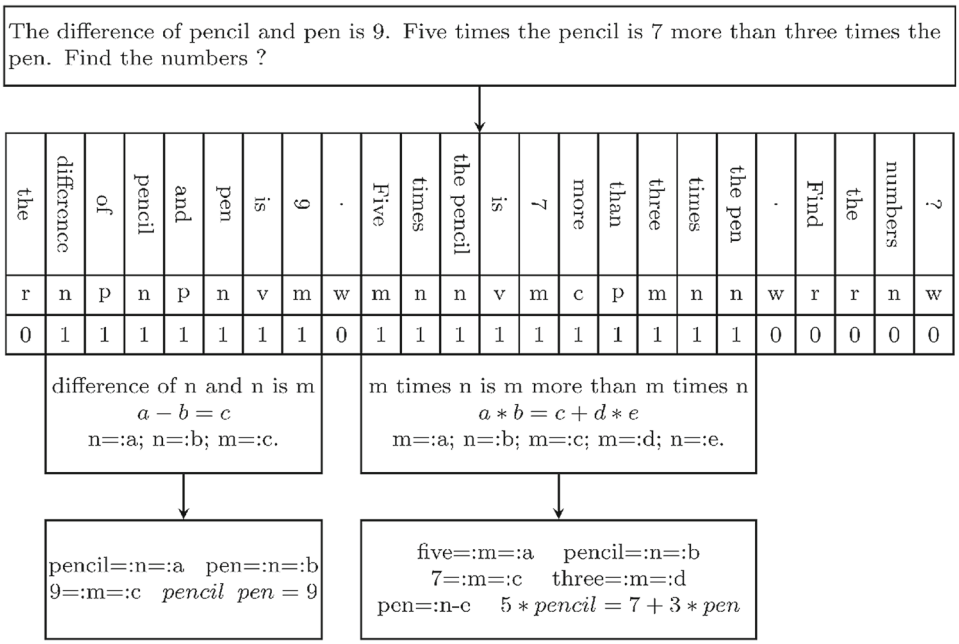

Fig. 3 The process of using S^2 method to extract explicit relations from a sample AWP

Definition 8 (*Matching Action*) A matching action is an action which matches the structure of K, P, Q from a quadruple $M = (K,\ P,\ Q;\ R)$ to a portion of problem text.

After loaded with a pool of S^2 models, Procedure I can extract explicit relations from a given input text. To acquire all explicit relations, it is necessary to match all the models in the pool with all potential portions of text. The crucial point of the procedure is to judge whether a model matches a portion of problem text. The outcome of this procedure is a set of explicit relations. Figure 3 uses an example to illustrate how Procedure I extracts explicit relations from AWPs. This whole process is called as an S^2 method.

When using the S^2 method, the main job of model matching is to match the POS change pattern with a portion of text. There are eight types of frequently used POS in S^2 models in English: n (nouns), v (verbs), a (adjectives), p (pronouns), m (numerals), c (conjunctions), r (particles), and w (punctuation marks). Each POS type corresponds to one of the twelve universal POS tags from natural languages.

This paper prepares pools of S^2 models for Chinese and English respectively. The pools of S^2 models for Chinese and for English consists of 220 and 360 S^2 models respectively. Table 1 lists eight frequently used S^2 models for solving problems in English.

Neural network miner for acquiring implicit relations

This section develops a procedure to acquire the implicit relations from problems. There are two cases of the implicit relations considered in this paper: *unit conversion* and *arithmetic formula*. *Unit conversion* is to add the relations that convert the different units of the same measurement appearing in the same problem. *Arithmetic formula* targets to add appropriate formula about corresponding scenarios. For example, when a problem appears to a scenario of calculating the area of a rectangular object, the component adds the formula of the rectangle area.

For the *unit conversion*, Dewappriya et al. [34] considered it as an issue of unit conflict and proposed a procedure to solve it. This paper adopts this procedure and uses it as a prior process to obtain the entire unit conversion relations for each measurement system involved in a given problem without giving the detail of this process.

For the *arithmetic formula*, the neural network miner can be a useful tool for such types of tasks since it can acquire the hints from the problem text. As it is known, the implicit relation belonging to the arithmetic scenario is highly related with quantity words. Hence, this paper proposes a neural network miner based on *quantity to relation attention neural network* (QRAN) to mine implicit relations (Procedure II). The procedure consists of three steps:

The first step is to encode the given problem into a sequence of vectors. A given problem can be tokenized as $\mathcal{P} = \{w_i\}_{i=1}^{n}$, each token w_i can be represented as a word-context feature vector v_i by BERT [50]. Thus, $\mathcal{P}$ can be denoted by a sequence of vectors as $\mathcal{V} = \{v_i\}_{i=1}^{n}$. The next process is to select the vectors related to quantity, including the numeric words like "100", "1/2" and the descriptive words like "double", "half". Let $\mathcal{N}$ denote the set of the quantity vectors in the problem, and place v_i into $\mathcal{N}$ if w_i is a quantity word. Thus, $\mathcal{N} = \{q_i\}_{i=1}^{k}$ contains all the quantity

Table 1 The list of eight frequently used models from the pool of S^2 models for solving AWPs in English and examples of these eight models extracting explicit relations from problem text

No	S^2 Model	Examples of extracting explicit relations		
		Problem text	Matching	Explicit relation
1	There m n; $a = b$; a =: m, b =: n	There are 14 poplar trees in the school	$\xrightarrow{model1}$	$poplar\_tree = 14$
2	n m times n; $a = b \times c$; a =: n, b =: m, c =: n	Poplar trees are 2 times that of pine trees	$\xrightarrow{model2}$	$poplar\_tree = 2 \times pine\_tree$
3	n m more than n; $a = b + c$; a =: n, b =: m, c =: n	Willow trees are 4 more than pine trees	$\xrightarrow{model3}$	$willow\_tree = 4 + pine\_tree$
4	n is m q; $a = b * c$; a =: n, b =: m, c =: q	The walking time to school is 1.5 hours	$\xrightarrow{model4}$	$time = 1.5 \times hour$
5	sum of n n is m q; $a + b = c * d$; n =: a n =: b m =: c, d =: q	The sum of the upper and lower base of a trapezoid is 7 meters	$\xrightarrow{model5}$	$upper\_base + lower\_base = 7 \times meter$
6	difference of n n is m; $a - b = c$; n =: a n =: b m =: c	The difference of the fifth and fourth grades is 41	$\xrightarrow{model6}$	$fifth\_grade - fourth\_grade = 41$
7	n m q per q; $a = b * c/d$; a =: n, b =: m, c =: q, d =: q	The average harvest of potatoes is 1.5kg per square meter	$\xrightarrow{model7}$	$potato = 1.5 \times kg/square\_meter$
8	n is m times less than n; a = (1-b)*c; a =: n, b =: m, c =: n	The number of roses is 0.2 times less than that of daffodils	$\xrightarrow{model8}$	$rose = (1 - 0.2) \times daffodil$

vectors, where k is the total number of the quantity words in a problem.

The second step is to obtain the goal vector v_g representing implicit relations by adopting the quantity-relation attention mechanism. The concrete computing process is as follows:

$$\mu_i = \alpha \cdot tanh(W_r \cdot [\bar{v}, q_i]) \quad for\ i = 1, 2, ..., k. \tag{1}$$

$$a_i = \frac{exp(\mu_i)}{\sum_{j=1}^{k} exp(\mu_j)} \tag{2}$$

$$v_g = \sum_{i=1}^{k} a_i \cdot q_i \tag{3}$$

where $\bar{v}$ is the average vector of the sequence $\mathcal{V}$, μ_i is the relevance score between the whole problem and a quantity related word, a_i is the attention score of each quantity in a softmax manner, α and W_r are parameters trained for each specific implicit mathematical relation.

The goal vector v_g can be transformed to a indicator $\hat{y}$ through a dense layer, judging whether an implicit relation needs to be added. The $\hat{y}$ is defined as Eq. (4). As $\hat{y}$ is the predicted value of the implicit relation category, we can get the corresponding relation from the prepared implicit relation knowledge base.

$$\hat{y} = \sigma(W_c \cdot v_g + \beta_c) \tag{4}$$

where σ is the sigmoid function, W_c and β_c are trainable parameters.

A knowledge base $\mathbb{D}$ consisting of pairs of implicit category with a corresponding formula (abstract relation) is constructed. We assume that the ground truth label of a problem is $y \in \mathbb{D}^C$, where $y_i = \{0, 1\}$ denotes whether label i appears in the problem or not. The whole network is trained using the multi-label classification loss as follow

Procedure II: Discovering and Adding Implicit Relations.

Input: A problem in text format.

Output: A collection of implicit relations Δ.

Step 1: Initialize Δ as empty;
BERT converts the problem text into the vector sequence;

Step 2: QRAN discovers the abstract relations from the vector sequence to be added;

Step 3: Instantiate the variables in the abstract relations with the entities in problem text. Add all the instanced relations into Δ.

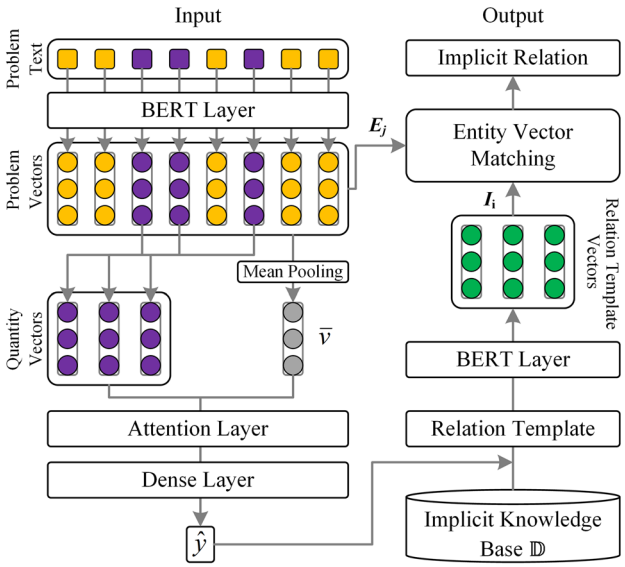

Fig. 4 The architecture of the main part of Procedure II

$$Loss(y, \hat{y}) = -\frac{1}{C}\sum_{i=1}^{C}[y_i log(\hat{y}_i) + (1 - y_i) log(1 - \hat{y}_i)] \tag{5}$$

where C denotes the number of categories of formula (implicit relations).

The third step is to instantiate the variables in the formula by connecting the entities in problem text. Each variable in implicit relation are transformed to word vectors through BERT, represented as a vector I_i. And each entity in the problem text obtained in extraction of explicit relation can be represented as a word vector $E_j = \{\bar{v}\}^l$, where l is the token length of the word. The cosine similarity $cos(I_i, E_j)$ is adopted to calculate the semantic similarity between variable in implicit relation and the entity in text. Once the similarity $cos(I_i, E_j)$ get the max value, the i-th variable in abstract relation would be substituted by the j-th entity in problem text. A instantiated implicit relation would be obtained after all the variables connecting to the corresponding entity. For a problem, all the instantiated implicit relations would be added to the output collection Δ finally.

Procedure II adds implicit relations to complement the problem understanding relying on the trained QRAN to discover the requiring abstract relations. Figure 4 illustrates an architecture of the main part of Procedure II. The implementation details of QRAN are presented in Appendix A.

Symbolic solver

Symbolic solver takes three steps to transform the group of relations to find the values of unknowns. The first step is to acquire the group of distilled relations (Procedure III). The second step is to build a system of equations (Procedure IV)

Procedure III: Acquiring the Fused Set of Relations.

Input: The group of relations candidates, denoted as G_c.

Output: A fused group of relations, denoted as G_f.

Step 1: Add all the unit conversion relations of the unit systems appeared in relation candidates into empty G_f;

Step 2: Identify the unknowns of the problem;

Step 3: Build a graph of relation relevance;

Step 4: Identify the solution graph;
Add all the relations in the solution graph into G_f.

and the third step solves the system of equations (Procedure V).

Distillation of explicit and implicit relations

Procedure I and II together produce a set of candidate relations from a given problem text. These candidate relations might contain relations that symbolic solver does not use and they might lead the solver to produce wrong results. Hence, a procedure needs to discard as many as possible these unnecessary relations. Procedure III proposes to perform the selection from the set of candidate relations based on global and connection characteristics. The main idea lies in identifying the necessary relations according to their link with problem unknowns. The detailed process is as follows: First, it builds a relation graph, whose nodes are all the candidate relations and whose links indicate whether two relations are relevant. We start to build this graph from the relations that contain the unknowns; add a node for a relation and add links if it has the sharing quantity objects with the built nodes in the graph and it is consistent with nodes in terms of arithmetic formulas. In other words, whether two relations are relevant is equivalent to whether two relations have the sharing quantity objects and they are involved in the same arithmetic formula. Then it identifies the solution graph, which is a partial graph, contains all problem unknowns and all nodes connecting to any unknown, and all the links among these nodes. The relations in the solution graph forms a set of distilled relations.

Forming a system of equations

To obtain the system of equations for a given problem, we need to list all entities that appear in the distilled relations. Then we assign a variable to each entity. The assigned variables replace all the entities in the distilled relations. Each relation thus turns into an equation and all the relations form a system of equations. The system of equations and the entity-variable table represent the given problem together. The detailed process is given in Procedure IV.

Procedure IV: Forming a System of Equations.

Input: The group of relations G_f.

Output: A system of equations, denoted as S_E and the correspondence table between entities in the problem text and variables in the equations with their domains $L(E, V)$.

Step 1: Make a list of all the entities denoting it as $L(E)$, declare a list of variables corresponding to $L(E)$ denoting it as $L(V)$, and form a corresponding table between $L(E)$ and $L(V)$ with the domains of variables denoting it as $L(E, V)$;

Step 2: Form a system of equations S_E from G_f by replacing the entities with the corresponding variables listed in $L(E, V)$.

Solving the equation system

The system of equations always contain some linear equations, though the whole system may not be a linear system. Gaussian elimination can solve these linear equations. The new linear equations forms when the solution of the linear equations replaces variables. This recurring manner will solve the whole system of equations. The detailed process is shown in Procedure V.

Experiments

This paper prepares nine datasets from authoritative resources and chooses five baseline algorithms as benchmarks. Then it compares the proposed algorithm with the five baseline algorithms on the nine datasets.

Experimental settings

Datasets: An explicit AWP is an AWP such that a group of explicit relations is its understood state; an implicit AWP is an AWP that is not explicit. Thus, each of the datasets can be divided into an explicit set and an implicit set; the explicit set (the implicit set) contains all the explicit problems (the implicit problems). Assume that X is the name of a dataset.

Procedure V: Solving the System of Equations.

Input: The system of equations S_E.

Output: The values of all the unknowns in the system.

Step 1: Divide S_E into linear part L and non-linear part H;

Step 2: Solve linear part L by Gaussian elimination;
Goto Step 3 if no new variables get their values;
Add values of solved variables into values set;
Substitute values set into H and update S_E;
Goto Step 1 if S_E is not empty;

Step 3: Output all the pairs of the unknowns and their corresponding values.

Then X:E, X:I and X:3 denote three partial sets of X. X:E denotes the explicit set of X that contains only explicit problems; X:I denotes the implicit set of X that contains only implicit problems; X:3 denotes the special set of X that contains problems whose equivalent relation group needs at least one relation of part whole, time distance, and interest. Table 2 gives a list of sample problems corresponding to the three partial sets X:E, X:I, and X:3. In addition, each dataset has its English and Chinese versions, no matter the language in which its original version is.

Table 3 lists the prepared nine datasets: PEP, BNU, and USC are three datasets from textbooks, which include all problems from textbooks for primary students published by People's Education Press in 2018, by Beijing Normal University in 2018 both in China, and used in California of USA in 2018, respectively. EEP is a dataset that includes all problems from entrance exam papers for high school in 34 provinces of China from 2010 to 2019. M23K, Ape5kT, MAWPS, Dol2K, and Arith are five datasets of AWPs from the generic research community. Math23K is a popular dataset in Chinese. Ape210K is currently the largest dataset in Chinese. M23K is from Math23K by discarding pure numerical problems and special symbol calculation because both baseline algorithms and the proposed algorithm are not designed for such problems. Ape5kT is the test set of Ape210K, which is the test portion of Ape210K in [9]. The baseline algorithm in [9] was tested on Ape5kT, after it used Ape210K as training set. Hence, Ape5kT is chosen as test set instead of Ape210K because the existing algorithms have never been tested on Ape210K. MAWPS, Dol2K, and Arith are another three test datasets used by the baseline algorithms.

Baseline algorithms: The algorithm presented in this paper is denoted as **PROPOSED** in this section. This paper compares **PROPOSED** with five baseline algorithms described below. **PROPOSED** fully compares with the first four of five baseline algorithms on the nine datasets because their implementations are provided by the authors of the papers or prepared by us.

- **MaAlg:** MaAlg is an algorithm presented in [15], being the best one among the algorithms of adopting *two-frame* approach. It improved Kintsch's algorithm presented in [14] in two aspects. First, the algorithm extends the definition of knowledge frame so that it can solve problems that require multiple steps. Second, it uses more than 1600 sentence templates to extract knowledge from problem text.
- **RoyAlg:** Roy et al. [20] proposed an *answer expression* approach that lies in mapping obtained quantities and operands to an expression. RoyAlg is reported in [23], being a representative algorithm among the multiple algorithms in this approach [20–23].
- **ZhaoAlg:** ZhaoAlg is a *seq2seq* algorithm proposed in [9]. ZhaoAlg leverages on an open Neural Machine Translation (NMT) tool with a copying mechanism. The largest dataset of AWPs is used to train NMT, however, the trained network is tested only on Ape5kT.
- **LinAlg**: LinAlg is another *seq2seq* algorithm in [12]. It uses a hierarchical structure to enhance the ability to understand math word problems. Unlike the previous seq2seq model that encodes the text into a single vector sequence for decoding [4,7,8,10], this model uses a hierarchical coding structure to encode the contextual semantics of the words in the clauses, the semantic dependencies in the clauses, and the inter-clause relation, respectively. The decoder decodes the vector sequences formed in different levels of hierarchy to enhance the decoder's attention to different levels of semantic information.
- **ShiAlg:** ShiAlg is an *equation-centric* algorithm in [18] and uses the formal language as its intermediate form. The design of this algorithm requires DOL (abbreviation of dolphin language) as an intermediate language. Given a problem, the algorithm first converts it into its intermediate expression in DOL. Then it derives a system of equations from the DOL expression of problems.

Mandal et al. [44] proposed an another *answer expression* algorithm. It uses a BiLSTM to classify operations among four basic operations and an irrelevant-information removal unit to identify the relevant quantities to form an expression to solve AWPs. It conducts experiment on AddSub and SingleOp, being two subsets of MAWPS, where all the problems only require a single equation with a single operation to be solved. This paper does not select the algorithm presented in [44] as a baseline because it targets to solve a limited scope of AWPs.

Table 2 The list of sample problems corresponding to the three partial sets X:E, X:I, and X:3

Name	Problem text	Explicit relation	Implicit relation
X:E	There are 14 poplar trees in the school. Poplar trees are 2 times pines. Willow trees are 4 more than pines. How many willow trees are there?	Poplar = 14 $Poplar = 2 \times pine$ $Willow = 4 + pine$	—
X:I	A rectangular forest with a length of 96 meters and a width of 58 meters. If a forest can produce 75 grams of oxygen per square meter per day, how many oxygen can it produce per day?	$Length = 96$ $Width = 58$ $Oxygen\_yield = 75$	$Rectangle\_area = Length \times Width$ $Production = Yield \times Area$
X:3	Ming walk to university takes 0.5 hours. The average speed is 10.5 kilometers per hour. How many kilometers did he walk?	$Time = 0.5$ $Speed = 10.5$	$Distance = Speed \times Time$
	Tom has 9.0 balloons. Sara has 8.0 balloons. How many balloons do they have in total?	$Tom = 9.0 \times balloon$ $Sara = 8.0 \times balloon$	$Whole = \sum_{i=1}^{j} Part_i$
	How much interest is earned on a principal of $863 invested at an interest rate of 4% for 5 years?	$Principal = 863$ $Interest\_rate = 4\%$ $Time = 5 \times years$	$Interest = Principal \times Interest\_rate \times Time$

Table 3 The prepared nine datasets and their sources

Datasets		Explicit problems		Implicit problems		Original language	Source of collecting problems
Name	Size	Name	Size	Name	Size		
PEP	504	PEP:E	226	PEP:I	278	Chinese	Textbook published by People's Education Press in 2018
BNU	436	BNU:E	293	BNU:I	143	Chinese	Textbook published by Beijing Normal University Press in 2018
EEP	2722	EEP:E	1641	EEP:I	1081	Chinese	Entrance Exam Papers from 2010 to 2019
M23K	20910	M23K:E	16568	M23K:I	4342	Chinese	Cleaned from Math23K in [4]
Ape5kT	4570	Ape5kT:E	3431	Ape5kT:I	1139	Chinese	Cleaned from Ape210K test set in [9]
MWP	2373	MWP:E	2299	MWP:I	74	English	Called as MAWPS in [51]
USC	497	USC:E	436	USC:I	61	English	US California Textbook in 2018
Dol2K	1878	Dol2K:E	1809	Dol2K:I	69	English	Called as Dolphin1878 in [18]
Arith	1541	Arith:E	1053	Arith:I	488	English	Called as AllArith in [23]

The implementation of the baseline algorithms: This study implements MaAlg, ZhaoAlg, and LinAlg algorithms on our own test platform. Since both ZhaoAlg [9] and LinAlg [12] algorithms are a deep learning model, we set the same parameters as their original papers respectively. For RoyAlg, this study uses the source code provided by [23], taking problems in English as its input. Hence, RoyAlg can run on all the English datasets.

Experimental results

An arithmetic word problem (AWP) is an arithmetic exercise problem in the primary mathematics described in natural language. An algorithm is said that it can solve an AWP if it can output the correct pairs of all the unknowns of the problem and their values after a series of correct transforms.

An algorithm also called as a solver of AWPs if it can solve a decent percentage of AWPs from a dataset.

This paper compares **PROPOSED** with the baseline algorithms mainly using two evaluation indicators: (i) The first indicator is the correct accuracy, which is the percentage that an algorithm can correctly solve the AWPs from a dataset. So far the most solving algorithms have mainly focused on generating the correct answer and adopted the correct accuracy as the main evaluation indicator. (ii) The second indicator is the number of models and the corresponding correct accuracy. Both MaAlg and ShiAlg use a pool of templates or rules. The number of models or templates is a key indicator when evaluating these algorithms. Hence, PROPOSED also compares with MaAlg and ShiAlg on the number of templates or models because only these two baseline algorithms use templates or models to extract the knowledge from problem text.

PROPOSED compares with the baseline algorithms on the accuracy of solving problems in three cases for each of datasets: (i) on whole dataset; (ii) on explicit problems only; and (iii) on implicit problems only.

PROPOSED emphasizes on its application to learning systems. Hence, it outputs not only the answers of unknowns but also the understandable algorithmic solution. It shows that **PROPOSED** can output the understandable algorithmic solution that can explain to students how to solve the given AWPs used an example in Fig. 1(b).

Table 4 displays the analytical comparison of accuracy of solving problems between **PROPOSED** and the four baseline algorithms, namely MaAlg, RoyAlg, ZhaoAlg, and LinAlg, on the nine datasets. The analytical comparison is to find out the performances of the involved algorithms on explicit problems, implicit problems and problems with three types of relations. **PROPOSED** is tested on both Chinese and English version of every dataset whereas a baseline algorithm is tested on Chinese (English) version of every dataset as it actually did in the reference paper.

In Tables 4 and 5, "–" indicates that the corresponding item is not available; a number in "Size" column/row indicates the number of the problems in the corresponding dataset; a number in "%" column indicates that the percent of problems that the corresponding algorithm can solve on the involved dataset. Table 4 reveals the four important conclusions:

- First, the component of acquiring implicit relations plays a critical role in enhancing the accuracy of **PROPOSED**. In accuracy, **PROPOSED** is at least 6.5% higher than any of four baseline algorithms on the nine datasets. **PROPOSED** achieves an accuracy of 78.3% on the Chinese version of the union of the nine datasets, whereas LinAlg achieves an accuracy of 71.8%, being the highest accuracy among the four baseline algorithms.
- Second, **PROPOSED** has a much better performance in solving implicit problems than the baseline algorithms. It is at least 18.2% higher than any of baseline algorithms on all the implicit subsets of the nine datasets. **PROPOSED** achieves an accuracy of 81.9% on the union of the Chinese implicit subsets of the nine datasets, whereas LinAlg achieves an accuracy of 63.7%, being the highest accuracy among the four baseline algorithms. On the set of all Chinese implicit problems with three types of relations of the nine datasets, **PROPOSED** is still at least 13.7% higher than any of baseline algorithms is.
- Third, **PROPOSED** has a better performance in solving problems on any particular dataset that even favors the particular algorithm. For example, Arith favors RoyAlg because problems in Arith have only types of the implicit relations discussed by RoyAlg. Hence, RoyAlg achieves an accuracy of 85.5%, being the highest percent among the nine datasets. Even so, **PROPOSED** is 2.8% higher than RoyAlg on the same dataset in accuracy of solving problems, which achieves an accuracy of 88.3%.
- Fourth, **PROPOSED** is just slightly better than LinAlg on the Chinese explicit problems of the nine datasets. Specifically, **PROPOSED** is 77.3% and LinAlg 74.0% for this accuracy. This can be partially explained by the fact that **PROPOSED** is even 0.1% lower than LinAlg in solving the problems in M23K:E, being one of nine explicit datasets.

Among the five baseline algorithms, Ma et al. [15] has listed the sentence templates so that this study implements it on our test platform. Hence, **PROPOSED** can compare with MaAlg on solving explicit problems because the sentence templates can extract only explicit relations. MaAlg claims that it has correctly understood a problem if the frames filled by MaAlg contain all knowledge items needed by symbolic solver, while **PROPOSED** can understand a problem if it can acquire a group of relations as an understood state. Table 5 shows two parts of comparisons. The first part is that **PRO-**

Table 4 The analytical comparison on the accuracy of solving problems considering **PROPOSED** against the four baseline algorithms on the nine datasets

Language of versions		Chinese version				English version	
Algorithms		MaAlg	ZhaoAlg	LinAlg	**PROPOSED**	RoyAlg	**PROPOSED**
Name	Size	%	%	%	%	%	%
PEP	504	35.5	64.3	62.1	**83.3**	45.4	**77.6**
BNU	436	49.1	67.4	67.7	**83.5**	61.2	**79.4**
EEP	2722	42.1	68.9	69.3	**79.1**	53.2	**75.4**
M23K	20910	34.6	73.1	74.1	**77.7**	28.4	**75.4**
Ape5kT	4570	34.9	70.1	70.9	**76.6**	38.2	**75.3**
MWP	2373	54.7	83.2	83.7	**84.2**	74.8	**83.9**
USC	497	57.5	71.4	69.2	**76.3**	73.8	**79.9**
Dol2K	1878	54.5	27.6	31.2	**70.8**	0.3	**84.0**
Arith	1541	46.9	82.9	83.0	**88.3**	85.5	**89.3**
Sum	35431	38.7	70.9	71.8	**78.3**	37.0	**77.2**
PEP:E	226	79.2	74.3	79.2	**84.1**	63.3	**74.3**
BNU:E	293	73.0	72.4	77.1	**82.9**	66.6	**78.8**
EEP:E	1641	69.8	76.4	76.3	**82.6**	64.3	**79.0**
M23K:E	16568	43.7	75.1	75.7	**75.6**	32.3	**74.5**
Ape5kT:E	3431	46.4	75.6	76.8	**78.0**	39.7	**77.0**
MWP:E	2299	56.5	84.2	84.6	**84.9**	74.9	**84.5**
USC:E	436	65.6	70.9	68.6	**75.5**	75.0	**78.7**
Dol2K:E	1809	56.6	27.6	31.1	**70.8**	0.3	**83.6**
Arith:E	1053	68.7	84.1	84.7	**86.2**	85.0	**87.0**
Sum	27756	49.3	73.1	74.0	**77.3**	39.9	**77.1**
PEP:I	278	0.0	56.1	48.2	**82.7**	30.9	**80.2**
BNU:I	143	0.0	57.3	48.3	**84.6**	50.3	**80.4**
EEP:I	1081	0.0	57.6	58.7	**73.8**	36.3	**69.9**
M23K:I	4342	0.0	65.7	68.1	**85.5**	13.6	**78.9**
Ape5kT:I	1139	0.0	53.5	52.9	**72.5**	33.5	**70.1**
MWP:I	74	0.0	52.7	54.1	**59.5**	70.3	**67.6**
USC:I	61	0.0	75.4	73.8	**82.0**	65.6	**88.5**
Dol2K:I	69	0.0	29.0	33.3	**72.5**	0.0	**94.2**
Arith:I	488	0.0	80.3	79.3	**92.8**	86.7	**94.3**
Sum	7675	0.0	62.8	63.7	**81.9**	26.6	**77.5**
PEP:3	75	0.0	66.7	69.3	**90.7**	64.0	**74.7**
BNU:3	76	0.0	73.7	72.4	**86.8**	61.8	**69.7**
EEP:3	585	0.0	63.6	62.4	**78.3**	56.8	**68.7**
M23K:3	3591	0.0	71.1	76.7	**89.8**	15.5	**84.1**
Ape5kT:3	567	0.0	70.9	70.2	**85.7**	64.4	**82.0**
MWP:3	74	0.0	52.7	54.1	**59.5**	70.3	**67.6**

Table 4 continued

Language of versions		Chinese version				English version	
Algorithms		MaAlg	ZhaoAlg	LinAlg	**PROPOSED**	RoyAlg	**PROPOSED**
Name	Size	%	%	%	%	%	%
USC:3	61	0.0	75.4	73.8	**82.0**	65.6	**88.5**
Dol2K:3	0	–	–	–	–	–	–
Arith:3	488	0.0	80.3	79.3	**92.8**	86.7	**94.3**
Sum	5517	0.0	70.9	74.3	**87.9**	33.8	**82.7**

POSED compares with MaAlg on Chinese versions of the nine datasets. The other part is that **PROPOSED** compares with ShiAlg on English versions of Dol2K dataset. We only implements MaAlg whereas the numbers on ShiAlg in Table 5 are cited from [18].

In Table 5, "1600+m" and "220m" mean that the corresponding algorithms use more than 1600 and 220 models respectively. Table 5 mainly presents the accuracy of solving problems on datasets with the number of used models. The accuracy is in the format of X%. For example, the first accuracy 79.2% means that MaAlg achieves 79.2% when it uses 1600+ models to work on PEP:E dataset in Chinese. Table 5 shows that **PROPOSED** is 4.9% higher than MaAlg on the accuracy on PEP:E in Chinese under the condition that the number of S^2 models used in **PROPOSED** is only one seventh of the number of sentence templates used in MaAlg. **PROPOSED** needs a smaller number of models because its models use syntax as much as possible, thus overcoming the variety of semantic expressions. Table 5 also shows that **PROPOSED** is much better than ShiAlg in both solving accuracy and the number of models. ShiAlg needs 9600 rules because these rules are semantics based.

After we have compared **PROPOSED** with the baseline algorithms, we conduct an ablation experiment to show the contribution of the relaxed S^2 method and QRAN to the accuracy of **PROPOSED** solving AWPs. Concretely, we investigate how much the relaxed S^2 method and QRAN contribute to increase the accuracy of the whole **PROPOSED**. Table 6 presents the results of this ablation experiment. In Table 6, "**Not-Relaxed**" means the **PROPOSED** uses the original S^2 method but not the relaxed S^2 method; "**No-QRAN**" means that we run **PROPOSED** without the function of adding implicit relations; "**No-Both**" means the **PROPOSED** does not use either the relaxed S^2 method or the function of adding implicit relations.

Table 6 tells three conclusions. The first conclusion is that the relaxed S^2 method increases the accuracy of **PROPOSED** solving AWPs by at least 6.3% at any dataset. The second one is that **QRAN** increases the accuracy of **PROPOSED** solving AWPs by from 1.9% to 45.6% on the different datasets. The third one is that the relaxed S^2 method and QRAN together increase the accuracy of **PROPOSED** solving AWPs by from 8.8% to 48.6% on the different datasets.

Discussions on experimental results: For a given dataset, the implicit ratio is the ratio of implicit problems to total problems. The implicit ratio of EEP dataset is 39.7%. We use this ratio as the reference ratio because the problems in EEP dataset are from entrance exam papers. The implicit ratios of M23K, Dol2K, and Arith are 20.8%, 3.7%, and 31.7%, respectively. Implicit ratios of M23K and Dol2K can thus be deemed low, compared with the reference ratio. This discloses the fact that most of the existing algorithms focus on solving explicit problems because they are tested only on the datasets with low implicit ratio. Arith is the only dataset that is used to test the algorithm that can extract some implicit relations. Hence, its implicit ratio is higher than the other datasets, though it is still lower than the reference ratio by 8%. Interestingly, RoyAlg achieved high accuracy, being 73.8% and 85.5% on USC and Arith in English, respectively. However, the accuracies that RoyAlg works on other datasets are lower than 62%. It seems that RoyAlg is sensitive to the problem type.

For the evaluation in the baseline algorithms, an algorithm is said that it can correctly solve an AWP if its output can correctly match to the ground-truth, but they do not check whether all the transforms of producing the correct answers are correct. Thus, we discover the phenomenon that some seq2seq algorithms may give correct answers using wrong answer expressions. Table 7 shows two examples of this phenomenon. In the first example, the algorithm does not figure out the concept of "average speed of round trip", and mistakenly takes n_3 as a multiple of total distance. In the second example, the model does not understand the concept of "half of the book", and coincidentally takes n_3 as a multiple of the number of pages read. Such confusion of mathematical concepts would lead to wrong answers while problem statements might have some changes.

For the task of acquiring implicit relation by neural network miner, this paper has tried other models, such as Word2Vec, GLOVE, and FastText, but we find that the BERT model suits the task. The BERT model can capture the over-

Table 5 The comparison on the accuracy of solving AWPs and the number of models considering **PROPOSED** against both MaAlg and ShiAlg on the explicit portions of the nine datasets and Dol2K dataset respectively

Dataset	Language	Chinese									English		
	Name	PEP:E	BNU:E	EEP:E	M23K:E	Ape5kT:E	MWP:E	USC:E	Dol2K:E	Arith:E	–	Name	Dol2K
	Size	226	293	1641	16568	3431	2299	436	1809	1053	–	Size	1878
MaAlg	1600+m	79.2%	73.0%	69.8%	43.7%	46.4%	56.5%	65.6%	56.6%	68.7%	ShiAlg	9600m	68.1%
PROPOSED	**220 m**	**84.1%**	**82.9%**	**82.6%**	**75.6%**	**78.0%**	**84.9%**	**75.5%**	**70.8%**	**86.2%**	**PROPOSED**	**360 m**	**84.0%**

Table 6 The ablation experiments on the nine datasets of showing the contribution of the relaxed S^2 method and QRAN to the accuracy of **PROPOSED** solving AWPs

Language of versions	Algorithms	Source	From textbooks and exam Papers				From published datasets					
		Name	PEP	BNU	USC	EEP	M23K	Ape5kT	MWP	Dol2K	Arith	Sum
		Size	504	436	497	2722	20910	4570	2373	1878	1541	35431
Chinese version	**PROPOSED**	%	83.3	83.5	76.3	79.1	77.7	76.6	84.2	70.8	88.3	78.3
	Not-Relaxed	%	63.7	64.0	65.6	68.1	66.8	68.2	74.7	63.4	78.9	67.8
	No-QRAN	%	37.7	55.7	66.2	49.8	59.9	58.5	82.3	68.2	58.9	60.6
	No-Both	%	34.7	45.9	50.3	37.6	49.2	48.3	73.4	62.0	46.0	50.1
English version	**PROPOSED**	%	77.6	79.4	79.9	75.4	75.4	75.3	83.9	84.0	89.3	77.2
	Not-Relaxed	%	61.1	59.6	69.8	67.5	65.8	66.2	75.4	64.5	83.0	67.2
	No-QRAN	%	33.3	53.0	69.0	47.6	59.1	57.8	81.8	80.5	59.4	60.4
	No-Both	%	31.5	43.8	52.9	36.7	47.9	45.8	73.2	63.5	53.0	49.3

all semantic differences of the problem text and the semantic differences of entities in different scenarios, thus it achieves better experimental results. For example, when the feature word "rectangle" appears in the problem text, it needs to figure out that the arithmetic scene is about the area or the perimeter. There are some other improved models after the BERT model, such as RoBERTa, GPT3, XLNET, and T5. These models maybe improve the accuracy of our algorithm in acquiring implicit relations. However, the main objective of this paper is to verify the effectiveness of the idea of transforming the complex task of solving AWPs into the smaller task of relation acquisition and transformation. The experimental results demonstrate that the relation-centric solving algorithm has achieved better results using the BERT model. Thus, we have not explored the models proposed after BERT. On the other hand, these models are similar in principle to the BERT model, although they may use some techniques for specific tasks and enhance the training data.

Among the five baseline algorithms, the seq2seq algorithms have claimed the highest performances in terms of answer accuracy. However, this accuracy is not a reliable accuracy because these algorithms have some false positive instances. This paper has given two examples of false positive instances in Table 7. Compared with the five baseline algorithms, the proposed algorithm has three common main advantages. The first one is that it has no false positive instances. The second one is that it broadens the range of solving arithmetic word problems, especially in solving implicit problems, outperforming state-of-the-art algorithms on all nine datasets. The third is that it is explainable because its actions of acquiring and transforming relations are understandable. More importantly, this series of actions can instruct students on how to solve the problem.

Conclusions and future work

This paper has taken the decomposition strategy to tackle the research problem of designing algorithms for solving AWPs. Following this notion, it has innovated a three-phase scheme of designing a high performance algorithm for solving AWPs. This three-phase scheme can break the heavy task of developing algorithms for solving AWPs into three smaller tasks. The first phase is to build a solving algorithm paradigm that summarizes and explains all algorithms for solving AWPs. The second phase is to form relation-centric approach by instantiating the solving algorithm paradigm. The third phase is to instantiate the relation-centric approach by proposing the methods to acquire the relations and to transform relations.

The proposed algorithm designed using three-phase scheme has three desired characteristics. The first one is that it is explainable, thanks to the fact that it solves AWPs with explicitly stated rules, models and algorithm actions. It is a

Table 7 Two examples that the seq2seq algorithms have the correct answer through evaluating wrong answer expressions

Problem text	A plane, in a speed of 500 (n_1) km/h, costs 3 (n_2) hours traveling from city A to city B. It only costs 2 (n_3) hours for return. How much is the average speed of the plane during this round-trip?	Qiao read a story book, 8 (n_1) pages a day, half of the book in 12 (n_2) days. Since then, the number of pages she read every day is 2 (n_3) times the original, how many days will she have to finish the rest?
Expression by seq2seq	$x = (n_1 * n_2) * n_3/(n_2 + n_3)$	$x = (n_1 * n_2 * n_3 - n_1 * n_2)/(n_1 * n_3)$
Ground truth	$x = (n_1 * n_2) * 2/(n_2 + n_3)$	$x = (n_1 * n_2)/(n_1 * n_3)$
Answer	$x = 600$	$x = 6$

very important characteristics that the algorithms output the explainable solution because it is an indispensable characteristics for building the tutoring function. In contrast, ZhaoAlg in [9] and LinAlg in [12] are not explainable because they solve the problems within a pure seq2seq approach. The second one is that the proposed algorithm has a neural network miner dedicated to acquiring implicit relations. Thus, it formed a procedure that synergizing the S^2 extractor for extracting explicit relations and the neural network miner for acquiring implicit relations. The last one is that it is a high performance algorithm. It outperforms all the baseline algorithms in terms of the number of used models and accuracy of solving AWPs. In addition to, it outputs an algorithmic solution to show a correct transforming process when it can solve the given AWP.

This paper has four technical contributions. First, it designed and implemented a three-phase scheme of designing solving algorithms. Second, it proposed a state-action algorithm paradigm. Third, it formed the relation-centric approach of solving AWPs. Fourth, it built a procedure of synergizing a relaxed S^2 method for extracting explicit relations and a neural network miner for acquiring implicit relations.

Three research areas among many potential ones could be further investigated in the near future. First, it is expected to adopt the three-phase scheme to design more solving algorithms. Second, it is feasible to develop the vector computing procedure to replace the text processing part of S^2 method. Third, it is anticipated to develop novel intelligent tutoring systems based on the new solving algorithms, which has been the ultimate motivation of our research.

Acknowledgements This work is partially supported by the National Natural Science Foundation of China (61977029) by Xinguo Yu and ARC Discovery Project (DP180101051) by Jun Shen.

Declarations

Conflict of interest On behalf of all authors, the corresponding author states that there is no conflict of interest.

Appendix A Implementation details of QRAN

Here, we first present the parameters setting of the training process of QRAN and then choose the optimal training parameters for our training results. Figure 5 shows the loss and accuracy graph of the training and validation of QRAN. During the training process of QRAN, the Adam optimizer with $learningrate = 2e-4, \beta_1 = 0.9, \beta_2 = 0.999$ is adopted. The batch size of the samples is set to 64, and the epochs are set to 50. The datasets used in training the QRAN are combination of 5 Chinese dataset (29142

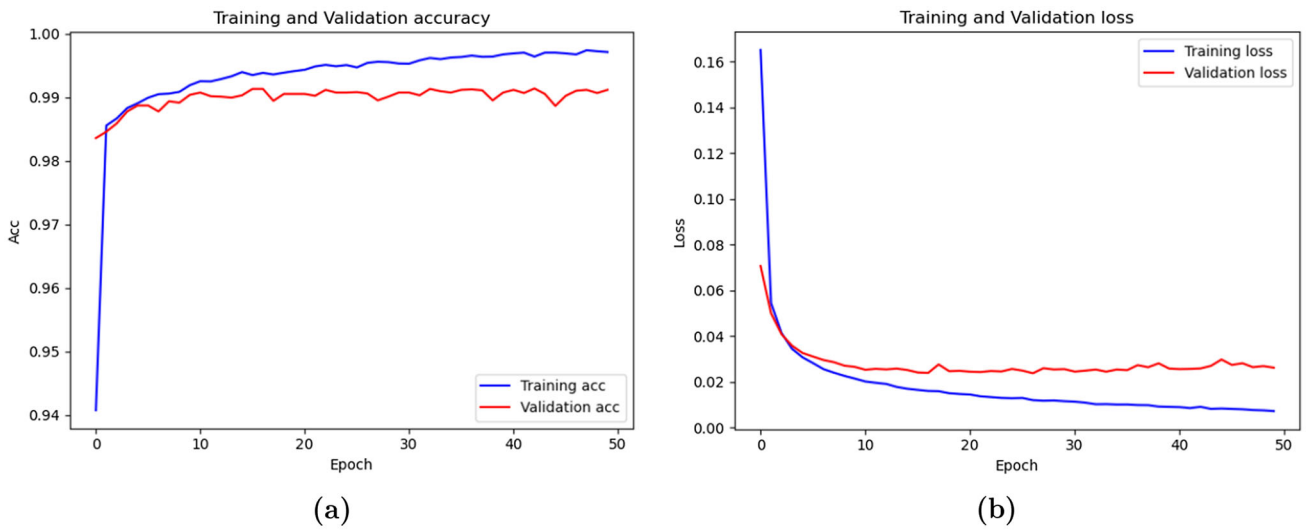

Fig. 5 The loss and accuracy graph of the training and validation of QRAN. Variation of accuracy with epochs is in (**a**); variation of loss with epochs is in (**b**)

Chinese instances) and 4 English dataset (6289 English instances), respectively. The train set contains 80% instances randomly sampled from the whole dataset, while the remaining instances form the test set. From the results of Fig. 5, we set the optimization epochs to 20, because the validation accuracy has converged.

References

1. Zhang D, Wang L, Zhang L, Dai BT, Shen HT (2019) The gap of semantic parsing: A survey on automatic math word problem solvers. IEEE Trans Pattern Anal Mach Intell 42(9):2287–2305
2. Faldu K, Sheth A, Kikani P, Gaur M, Avasthi A (2021) Towards tractable mathematical reasoning: Challenges, strategies, and opportunities for solving math word problems. arXiv:2111.05364
3. Bekoulis G, Papagiannopoulou C, Deligiannis N (2023) A review on fact extraction and verification. ACM Comput Surv 55(1):1–35
4. Wang Y, Liu X, Shi S (2017) Deep neural solver for math word problems. Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing. pp. 845–854
5. Xie Z, Sun S (2019) A goal-driven tree-structured neural model for math word problems. Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI-19), pp. 5299–5305
6. Liu Q, Guan W, Li S, Kawahara D (2019) Tree-structured decoding for solving math word problems. Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pp. 2370–2379
7. Zhang J, Wang L, Lee RK-W, Bin Y, Wang Y, Shao J, Lim E-P (2020) Graph-to-tree learning for solving math word problems. Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pp. 3928–3937
8. Zhang J, Lee RK-W, Lim E-P, Qin W, Wang L, Shao J, Sun Q (2020) Teacher-student networks with multiple decoders for solving math word problem. Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, pp. 4011–4017
9. Zhao W, Shang M, Liu Y, Wang L, Liu J (2020) Ape210k: A large-scale and template-rich dataset of math word problems, 1–10 arXiv:2009.11506v2
10. Hong Y, Li Q, Ciao D, Haung S, Zhu S-C (2021) Learning by fixing: Solving math word problems with weak supervision. The Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI-21, pp. 4959–4967
11. Hong Y, Li Q, Gong R, Ciao D, Huang S, Zhu S-C (2021) SMART: A situation model for algebra story problems via attributed grammar. The Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI-21, pp. 13009–13017
12. Lin X, Huang Z, Zhao H, Chen E, Liu Q, Wang H, Wang S (2021) HMS: A hierarchical solver with dependency-enhanced understanding for math word problem. Thirty-Fifth AAAI Conf Artif 2021:4232–4240
13. Cao Y, Hong F, Li H, Luo P (2021) A bottom-up DAG structure extraction model for math word problems. Thirty-Fifth AAAI Conf Artif 2021:39–46
14. Kintsch W, Greeno JG (1985) Understanding and solving word arithmetic problems. Psycho Rev 92(1):109
15. Ma Y, Ying Z, Cui G, Yun R, Huang R (2010) Frame-based calculus of solving arithmetic multi-step addition and subtraction word problems. Int Workshop Edu Tech Comput Sci 2:476–479
16. Ma Y, Zhou Y (2010) The method of semantic analysis for arithmetic word problems. 5th Int Conf Comput Sci Edu pp. 565–569
17. Ma Y, Tan K, Shao L, Shang X (2011) Constructing the representation model of arithmetic word problems for intelligent tutoring system. 6th International Conference on Computer Science & Education, pp. 250–255
18. Shi S, Wang Y, Lin C-Y, Liu X, Rui Y (2015) Automatically solving number word problems by semantic parsing and reasoning. Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pp. 1132–1142
19. Liang C, Hsu K, Huang C, Li C, Miao S, Su K (2016) A tag-based english math word problem solver with understanding, reasoning and explanation. Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 67–71
20. Roy S, Roth D (2015) Solving general arithmetic word problems. Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP, pp. 1743–1752
21. Roy S, Roth D (2016) Unit dependency graph and its application to arithmetic word problem solving. Association for the Advancement of Artificial Intelligence, pp. 3082–3088
22. Mitra A, Baral C (2016) Learning to use formulas to solve simple arithmetic problems. Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics 1:2144–2153
23. Roy S, Roth D (2018) Mapping to declarative knowledge for word problem solving. Trans Assoc Comput Linguist 6:159–172
24. Kushman N, Artzi Y, Zettlemoyer L, Barzilay R (2014) Learning to automatically solve algebra word problems. Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, pp. 271–281
25. Zhou L, Dai S, Chen L (2015) Learn to solve algebra word problems using quadratic programming. Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pp. 817–822
26. Upadhyay S, Chang M-W, Chang K-W, Yih W-t (2016) Learning from explicit and implicit supervision jointly for algebra word problems. Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pp. 297–306
27. Huang D, Shi S, Lin C-Y, Yin J (2017) Learning fine-grained expressions to solve math word problems. Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pp. 805–814
28. Yu X, Gan W, Wang M (2017) Understanding explicit arithmetic word problems and explicit plane geometry problems using syntax-semantics models. 2017 International Conference on Asian Language Processing (IALP), pp. 247–251
29. Yu X, Wang M, Gan W, He B, Ye N (2019) A framework for solving explicit arithmetic word problems and proving plane geometry theorems. Intern J Pattern Recognit Artif Intell 33(7):1940005–1194000521
30. Gan W, Yu X, Wang M (2019) Automatic understanding and formalization of plane geometry proving problems in natural language: A supervised approach. Int J Artif Intell Tool 28(04):1940003
31. Gan W, Yu X, Zhang T, Wang M (2019) Automatically proving plane geometry theorems stated by text and diagram. Intern J Pattern Recognit Artif Intell 33(07):1940003
32. Jian P, Sun C, Yu X, He B, Xia M (2019) An end-to-end algorithm for solving circuit problems. Intern J Pattern Recognit Artif Intell 33(07):1940004
33. He B, Yu X, Jian P, Zhang T (2020) A relation based algorithm for solving direct current circuit problems. Applied Intelligence, pp. 1–17
34. Dewappriya N, Kankanamge GU, Wellappili D, Hevapathige A, Ranathunga S (2018) Unit conflict resolution for automatic math word problem solving. 2018 Moratuwa Engineering Research Conference (MERCon), pp. 191–196

35. Chen L, Feng Y, Huang S, Luo B, Zhao D (2018) Encoding implicit relation requirements for relation extraction: A joint inference approach. Artif Intell 265:45–66
36. Chen Y, Zhang Y, Hu C, Huang Y (2021)Jointly extracting explicit and implicit relational triples with reasoning pattern enhanced binary pointer network. Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 5694–5703
37. Amnueypornsakul B, Bhat S (2014) Machine-guided solution to mathematical word problems. The 28th Pacific Asia Conference on Language, Information and Computation pages, pp. 111–119
38. Hosseini MJ, Hajishirzi H, Etzioni O, Kushman N (2014) Learning to solve arithmetic word problems with verb categorization. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), 523–533
39. Koncel-Kedziorski R, Hajishirzi H, Sabharwal A, Etzioni O, Ang SD (2015) Parsing algebraic word problems into equations. Trans Assoc Comput Linguist 3:585–597
40. Huang D, Yao J-G, Lin C-Y, Zhou Q, Yin J (2018) Using intermediate representations to solve math word problems. Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Long Papers), pp. 419–428
41. Zaporojets K, Bekoulis G, Deleu J, Demeester T, Develder C (2021) Solving arithmetic word problems by scoring equations with recursive neural networks. Expert Syst Appl 174:114704
42. Upadhyay S, Chang MW (2016) Annotating derivations: A new evaluation strategy and dataset for algebra word problems. Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Long Papers 1, 494–504
43. Wang L, Zhang D, Gao L, Song J, Guo L, Shen HT (2018) MathDQN: Solving arithmetic word problems via deep reinforcement learning. Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), pp. 5545–5552
44. Mandal S, Sekh AA, Naskar SK (2020) Solving arithmetic word problems: A deep learning based approach. J Intell & Fuzz Syst 39(2):2521–2531
45. Mandal S, Naskar SK (2021) Classifying and solving arithmetic math word problems-an intelligent math solver. IEEE Trans Learn Tech 14(1):28–41
46. Robaidek B, Koncel-Kedziorski R, Hajishirzi H (2018) Data-driven methods for solving algebra word problems. CoRR, 1–6 arXiv:1804.10718
47. Harshal K (2019) Girish, Kumar, Patnaik: A review of automatic math word problem solving techniques. Int J Manag technol Eng 9(1):1544–1548
48. Shen Y, Jin C (2020) Solving math word problems with multi-encoders and multi-decoders. Proceedings of the 28th International Conference on Computational Linguistics, pp. 2924–2934
49. Yu X, Jian P, Wang M, Wu S (2016) Extraction of implicit quantity relations for arithmetic word problems in Chinese. International Conference on Educational Innovation Through Technology, pp. 242–245
50. Devlin J, Chang M-W, Lee K, Toutanova K (2019) BERT: Pre-training of deep bidirectional transformers for language understanding. Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies 1:4171–4186
51. Koncel-Kedziorski R, Roy S, Amini A, Kushman N, Hajishirzi H (2016) MAWPS: A math word problem repository. Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 1152–1157