SMART: A Situation Model for Algebra Story Problems via Attributed Grammar

Yining Hong, Qing Li, Ran Gong, Daniel Ciao, Siyuan Huang, Song-Chun Zhu
University of California, Los Angeles, USA.
yininghong@cs.ucla.edu, {liqing, nikepupu, danielciao, huangsiyuan}@ucla.edu, sczhu@stat.ucla.edu

Abstract

Solving algebra story problems remains a challenging task in artificial intelligence, which requires a detailed understanding of real-world situations and a strong mathematical reasoning capability. Previous neural solvers of math word problems directly translate problem texts into equations, lacking an explicit interpretation of the situations, and often fail to handle more sophisticated situations. To address such limits of neural solvers, we introduce the concept of a *situation model*, which originates from psychology studies to represent the mental states of humans in problem-solving, and propose *SMART*, which adopts attributed grammar as the representation of situation models for algebra story problems. Specifically, we first train an information extraction module to extract nodes, attributes and relations from problem texts and then generate a parse graph based on a pre-defined attributed grammar. An iterative learning strategy is also proposed to further improve the performance of SMART. To study this task more rigorously, we carefully curate a new dataset named *ASP6.6k*. Experimental results on ASP6.6k show that the proposed model outperforms all previous neural solvers by a large margin, while preserving much better interpretability. To test these models' generalization capability, we also design an out-of-distribution (OOD) evaluation, in which problems are more complex than those in the training set. Our model exceeds state-of-the-art models by 17% in the OOD evaluation, demonstrating its superior generalization ability.

Introduction

Algebra Story Problems, depicted by Hinsley, Hayes, and Simon (1977) as "twentieth-century fables", remain an important challenge in modern artificial intelligence. An algebra story problem typically describes a real-world situation and inquires about an unknown fact in the situation. It goes beyond arithmetic since one has to first comprehend the situation and recognize the goal in the problem, and then come up with a solution for it (Nathan, Kintsch, and Young 1992). Psychology studies (Bjork and Bowyer-Crane 2013; Abedi and Lord 2001) also indicate that algebra story problems can serve as a test of children's cognitive skills to perform arithmetic reasoning on real-world tasks. However, although algebra story problems are distinguished *per se*, related works from the community of artificial intelligence and natural language processing often mix them with other types of problems, such as number problems and geometry problems, into one whole task called *Math Word Problems* (MWP) (Wang, Liu, and Shi 2017a; Huang et al. 2016; Amini et al. 2019)

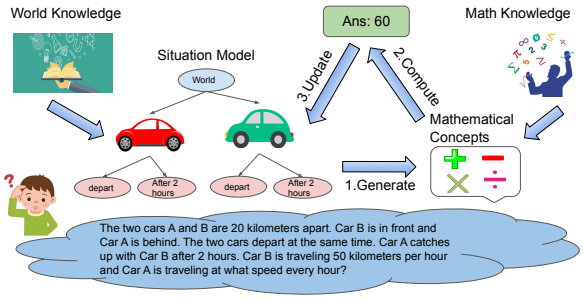


Figure 1: The process of human solving algebra story problems: We first hallucinate a situation model from the text and then perform arithmetic reasoning on the situation model to compute an answer. If we fail to generate a correct solution, we can adjust our situation model until we get one.

Recent works on Math Word Problems (Wang, Liu, and Shi 2017a; Huang et al. 2018a; Wang et al. 2018; Xie and Sun 2019; Hong et al. 2021) have focused on using end-to-end neural networks (*e.g.*, Seq2Seq, Seq2Tree) to directly translate a problem text into an expression, which is then executed to get the final answer. Although they seem to obtain decent performance, such end-to-end neural models suffer from the following drawbacks:

- Lack of interpretability. The expressions generated by neural networks are hard to interpret without the intermediate problem-solving process. An exemplary expression from Figure 3 is "(24+60)/[1-(1-(2/5))*(3/10)-(2/5)*(3/4)-(2/5)]", which makes no sense to humans, even though it generates the correct answer.
- Lack of generalization ability. These neural solvers usually fail in scenarios that are more sophisticated than those in training.

To address the confusions and drawbacks in current re-

search on Math Word Problems, we make the following efforts in this work.

First, we curate a new benchmark named *ASP6.6k*, which contains four canonical types of algebra story problems: motion, price, relation, and task. We build our dataset upon Math23K (Wang, Liu, and Shi 2017b), the most frequently-used MWP dataset in recent years, and categorize algebra story problems by following precisely the criteria set by Mayer (1981).

Second, we introduce the cognitive concept of a *situation model* (van Dijk and Kintsch 1983), which is widely used in psychology studies to model the mental states of humans in problem-solving (Reusser 1990; Greeno 1989; Nathan and Young 1990; Coquin-Viennot and Moreau 2007; Leiss et al. 2010). It is believed that problem-solving techniques, such as mathematics and logic, are applied to the hallucinated situation model instead of the problem text. As shown in Figure 1, the situation model interacts with mathematical concepts to derive a solution for the problem.

To computationally represent the concept of a situation model, we propose *SMART*, which utilizes attributed grammar (Knuth 1990; Liu, Zhao, and Zhu 2014; Park and Zhu 2015; Park, Nie, and Zhu 2016b) as the representation of a situation model in algebra story problems. More specifically, the world, agents, and events depicted in an algebra story problem are represented as nodes in a hierarchical parse graph, derived from the problem text by a context-free grammar. The nodes in the parse graph are further augmented with attributes to represent quantities in the problem, and the relations between these quantities are encoded as numerical constraints on the corresponding attributes. The construction of these constraints usually requires both commonsense knowledge and mathematical knowledge. Therefore, the parse graph generated by an attributed grammar is able to capture the desired characteristics of the situation model for algebra story problems. Equipped with the parse graph, the problem solving is equivalent to seeking one unknown attribute in the graph and can be formulated as an attribute propagation process, which is guided by the constraints on these attributes. To automatically construct parse graphs for problems, we first train an information extraction module to extract nodes, attributes and relations from problem texts and then generate parse graphs based on a carefully designed attributed grammar.

The learning of SMART is nontrivial. Since the grammar parsing and the problem solving are non-differentiable, we cannot use back-propagation to learn SMART in an end-to-end fashion under the supervision of final answers. Therefore, we propose a two-stage learning strategy: First, we manually design a text parser to generate initial supervision on parse graphs and use them to train the information extraction module in SMART. Second, we adopt an iterative learning method to reinforce the information extraction module, where pseudo-gold parse graphs at each iteration augment the supervision for the next learning iteration.

We conduct experiments on the newly curated benchmark ASP6.6k, and the proposed SMART model outperforms all neural network baselines by a large margin. Moreover, it demonstrates stronger generalization ability in an out-of-distribution evaluation, where the test problems are more complex than those in the training set. A qualitative study also suggests that SMART achieves better interpretability and generalization ability than the neural models.

Related Works

Math Word Problems Solving math word problems has attracted researchers for decades. Early solvers (Fletcher 1985; Bakman 2007; Yu-hui et al. 2010) use rule-based methods which are generally fixed and only work on single-step word problems for one category of problems. The next stage of solvers use semantic parsing techniques (Hosseini et al. 2014; Koncel-Kedziorski et al. 2015; Shi et al. 2015; Huang et al. 2017). These methods attempt to parse the problem text into an intermediate structured representation, usually annotated in the training set. Specifically, Shi et al. (2015) uses Context Free Grammar to solve number problems, which is quite different from our grammar for story problems. Statistical learning methods (Kushman et al. 2014; Zhou, Dai, and Chen 2015; Mitra and Baral 2016; Roy and Roth 2017; Huang et al. 2016) attempt to boost semantic parsing techniques, like choosing the most probable template to use (Mitra and Baral 2016). However, these templates are still fixed before training, leading to inflexibility in solving more sophisticated problems.

Researchers recently focus on solving math word problems using neural networks (Ling et al. 2017; Wang, Liu, and Shi 2017a; Huang et al. 2018a; Robaidek, Koncel-Kedziorski, and Hajishirzi 2018; Wang et al. 2018, 2019; Chiang and Chen 2019; Xie and Sun 2019; Zhang et al. 2020; Hong et al. 2021). The mere translation from a text to an equation neglects the intermediate process required by problem solving, thus lacking interpretability. In this paper, we seek to combine the strengths of both symbolic reasoning and neural networks, where we use neural modules to update symbolic representations.

Situation Model Situation models have been proven crucial in human discourse comprehension and problem solving (Zwaan, Magliano, and Graesser 1995; Nesher, Hershkovitz, and Novotna 2003). Researchers have long believed that text comprehension is a process of construction and integration (Gernsbacher 2013; Kintsch and Walter Kintsch 1998). Hegarty, Mayer, and Monk (1995) indicate that without a situation model, problem solvers with a direct translation approach are more likely to fail for math problems. A recent study (Raudszus, Segers, and Verhoeven 2019) also shows that the ability of building a situation model is a strong indicator of cognitive and linguistic skills.

There is a history of situation model construction for algebra story problem solving (Reusser 1990; Greeno 1989; Nathan and Young 1990; Coquin-Viennot and Moreau 2007; Leiss et al. 2010). Kintsch and Greeno (1985) use a situation model to analyze processing requirements and difficulties of algebra word problems. Nathan, Kintsch, and Young (1992) build a situation model to predict student mental state and predict how to tutor students based on interaction history. In contrast, this paper builds a situation model for the machine,

which uses attributed grammar to model the problem solving for algebra story problems. A situation model should satisfy the following properties (van Dijk and Kintsch 1983) :

- Reference: The situation model should represent the world the text is stating about.
- Coherence: All facts, implicit or explicit, need to be connected as long as the relations are indicated by the text.
- Situational Parameters: It includes the parameters and attributes about the world and events in the text.
- Event Independence: It needs to be invariant regardless of the number of events and their order.

We argue that a parse graph derived from attributed grammar can capture the above properties of a situation model.

Attributed Grammar Attributed grammar is proposed by Knuth to handle the semantics of programming languages (Knuth 1990). In recent years, researchers use attributed grammar to represent hierarchical structures such as (Han and Zhu 2005; Wang et al. 2013), video events (Lin et al. 2009), human poses (Park, Nie, and Zhu 2016a), indoor scene understanding (Qi et al. 2018; Jiang et al. 2018; Huang et al. 2018b; Chen et al. 2019), *etc*. Authors often assign attributes to terminal or non-terminal nodes of a grammar based on commonsense knowledge. Attributes propose constraints between terminal nodes and non-terminal nodes. These constraints can be soft constraints or hard constraints, depending on the task at hand. This paper also utilizes hard constraints for mathematical reasoning. In addition, attributes in a parse graph can be propagated in a controlled and formal way.

The *ASP6.6k* Dataset

We curate the new dataset *ASP6.6k* from the widely used Math23K dataset. To select and categorize algebra story problems, we first compute the term frequency–inverse document frequency (TF-IDF) features for each problem in the dataset and then use k-means clustering to group problems into different categories. We use the elbow method (Thorndike 1953) to find the optimal K for the clustering. To further remove noise, we manually select some keywords to filter out problems that do not belong to the group. We select the groups of problems by following the criteria inspired by (Mayer 1981; Nathan, Kintsch, and Young 1992):

- The problem uses words rather than equations.
- The problem has a story-line consisting of characters, objects, and/or actions.

As a result, we obtain a dataset of 6666 problems spanning from four typical types of algebra story problems: motion, price, relation, and task.

Here, we provide a brief summary of the problem types:

- **Motion**: problems involve traveling and require understanding of per time rate.
- **Task**: problems involve completion of tasks and require understanding of the relations between fractions.
- **Price**: problems involve purchasing items and require understanding of unit price and total price.
- **Relation**: problems involve a description of relationship between two objects.

	Motion	**Task**	**Relation**	**Price**	**Total**
Problems	1687	1158	1915	1908	6666
Avg. Length	37.0	33.6	28.8	26.7	31.1
Avg.Agents	1.85	1.13	2.34	1.96	1.90
Avg. Events	1.93	2.21	2.58	2.30	2.27
Avg. Relations	3.07	3.32	3.53	2.98	3.22

Table 1: Dataset Statistics. Length is number of tokens.

Details of the dataset statistics are listed in Table 1. Examples for each problem type are shown in Table 2. See supplementary materials for more details about dataset preprocessing and more statistics.

SMART

In this section, we introduce SMART, a situation model for algebra story problems via attributed grammar.

Attributed Grammar

Inspired by Qi et al. (2018), an attributed grammar is carefully designed for the domain of algebra story problems, as shown in Table 3.

$G = (S, V, A, E, R)$
S is the start symbol.
V = {S, World, Agents, Agent, Events, Event}
A = {rate, amount, total}
E = {e: e is a valid equation on attributes.}
R = {$S \rightarrow$ World
World $\rightarrow$ Agents
Agents $\rightarrow$ Agents Agent | Agent
Agent $\rightarrow$ Events
Events $\rightarrow$ Events Event | Event}

Table 3: The attributed grammar for algebra story problems.

In the definition of attributed grammar, the production rules R are inspired by the following observation: a problem usually depicts a *world*, in which several *agents* perform several *events*.

Inspired by (Roy and Roth 2017; Nathan, Kintsch, and Young 1992; Mayer 1981), we design three types of attributes to augment the nodes: i) *rate*: a quantity which is some measure corresponding to one unit of some other quantity, indicated by phrases like "A per B" and "each A has B" (*e.g.*, speed, price). ii) *amount*: a measurement of units of rate quantities (*e.g.*, hour). iii) *total*: a quantity which equals to the multiplication of rate and amount (*e.g.*, distance).

The relations E represent possible constraints on the attributes, in the form of equations. These constraints can be either explicitly stated in the text, such as "it travels 1/3 of the distance", or implied by commonsense knowledge, such as "distance = speed $\times$ time". See Figure 2 for an exemplary parse graph generated from the attributed grammar.

Problem Type	Sample Problem
Task	The engineering team built a viewing trail and completed 30% of the full length in the first week and 45% of the full length in the second week. 150 meters in two weeks, how long is the length of this trail?
Motion	Mingming's family went to travel, they took a 14-hour train ride, and then a 5-hour car ride before reaching their destination. It is known that the speed of the train is 120 kilometers/hour and the speed of the car is 60 kilometers/hour. How long is this journey?
Relation	Xiaogang's weight is 28.4 kg, Xiaoqiang's weight is 1.4 times that of Xiaogang, Xiaoqiang's weight = how many kilograms?
Price	The school bought 45 sets of desks and chairs at 128 yuan per desk and 52 yuan per chair. How much did it spend?

Table 2: An example of each problem type.

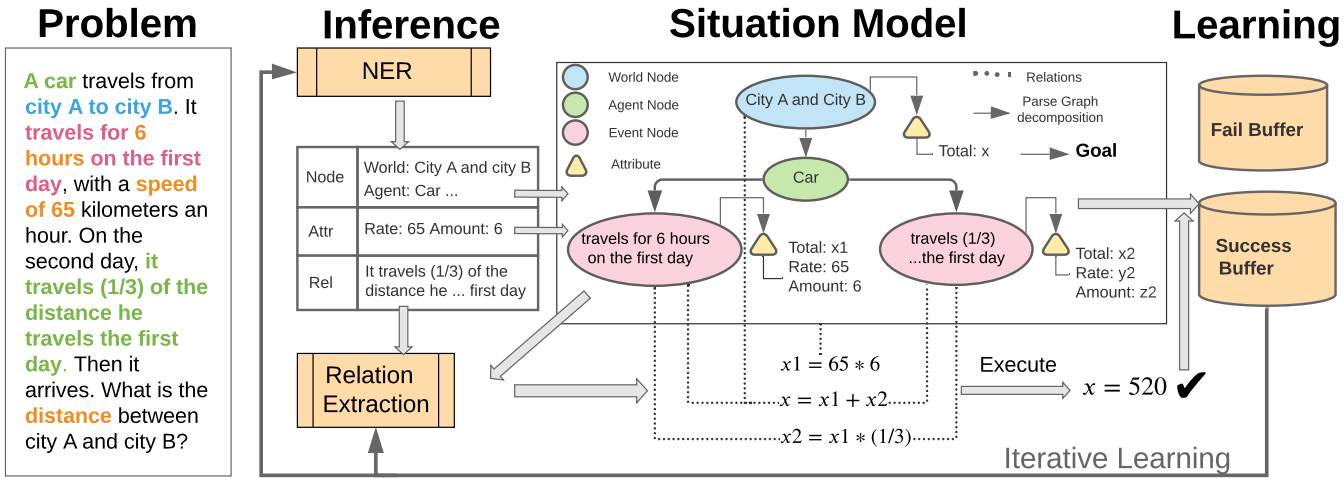


Figure 2: Overview of our SMART model. The Named Entity Recognition (NER) module extracts the spans of nodes, attributes, as well as relations from the text, and construct a parse graph using Attributed Grammar. The Relation Extraction module uses the relation spans and the parse graph already constructed to embed some relations into the parse graph. In the updated graph parser, Relation Extraction corresponds to Seq2Seq. The relations are then executed to get the final answer. If the answer is correct, it is added to the buffer of pseudo-gold parse graphs to train NER and Seq2Seq. If not, it is added to the failure set to be updated in the following iterations.

Grammar Parsing

The construction of the situation model for an algebra story problem is equivalent to parsing the problem text into a parse graph. More formally, given the problem x and the attributed grammar G, the parsing process is formulated as:

$$pg^* = \arg\max_{pg \in \mathcal{L}(G)} p(pg \mid x), \tag{1}$$

where $\mathcal{L}(G)$ denotes the language of the attributed grammar. The probability of a parse graph pg given x can be written as a joint probability of its nodes V_{pg}, attributes A_{pg} and relations E_{pg}:

$$p(pg \mid x) = p(V_{pg}, A_{pg}, E_{pg}|x) \tag{2}$$

$$= p(V_{pg}|x) \cdot p(A_{pg}|x) \cdot p(E_{pg}|x) \tag{3}$$

Here we assume the independence of these nodes, attributes, and relations to simplify our model and leave the exploration of their dependency for future works.

The extraction of nodes, attributes and relations is achieved by a three-step process.

First, we define seven categories of entities: nodes (WORLD, AGENT, EVENT), attributes (RATE, AMOUNT, TOTAL), REL (which denotes a text span that indicates relation). We train a named entity recognition (NER) system to recognize these entities from the text. Specifically, we have one NER model to extract the attributes, and another one for the extraction of the nodes and REL. We use Nested NER (Straková, Straka, and Hajic 2019) for the second model. We use BERT-chinese-base pre-trained model and fine-tune it on our NER task.

$$p(V_{pg}|x) = \frac{1}{|V_{pg}|} \sum_{w \in V_{pg}} p_{ner}(w)$$

$$p(A_{pg}|x) = \frac{1}{|A_{pg}|} \sum_{w \in A_{pg}} p_{ner}(w)$$

where $|V_{pg}|$ is the length of a node span, $|A_{pg}|$ is the length of an attribute span, and w is a word in the node or attribute span. $p_{ner}(w)$ is the probability of a word being labelled a specific category.

Second, we connect these nodes and attributes into a parse graph based on two kinds of distance: the word distance between two nodes in the problem text, and the distance (number of links) between them in the dependency parse. Some

constraints in the dependency parsing are also imposed, *e.g.*, the noun word representing an agent is preferred to be the NSUBJ of the verb word in the event node. Please refer to the supplementary materials for the complete list of rules and constraints used in the parsing. Attributes not extracted by NER are marked as an unknown.

Third, we train a Seq2Seq model to translate the REL entity from a text span into an equation e, which consists of attributes in the parse graph and arithmetic operators ($\{+, -, \times, \div, \wedge, =\}$). To include attributes in the equation, the input to the Seq2Seq model is the concatenation of REL, nodes and the attributes of each node. See supplementary for the examples of inputs and outputs of the Seq2Seq model. We have the probability of the relation as

$$p(E_{pg}|x) = \sum_{e \in E_{pg}} p_{seq2seq}(e|\text{REL}, V_{pg}, A_{pg})$$

where $p_{seq2seq}$ is the Seq2Seq output probability.

Goal Recognition

The goal of a problem is usually an interrogative word extracted by NER (*e.g., how many*). It can be in an attribute or in REL. We transform the interrogative word into an unknown in the equation.

Problem Solving

After building the parse graph, the problem can be easily solved by feeding the relation equations and the goal into a mathematical solver (Meurer et al. 2017). Although seemingly trivial, this stage actually requires mathematical reasoning skills, which are perfectly provided by a stand-alone solver.

The Learning of SMART

The learning objective of SMART is to optimize the information extraction module including both the NER system and Seq2Seq models for relation and goal. Since the grammar parsing and the problem solving are non-differentiable, we cannot use back-propagation to learn SMART in an end-to-end fashion under the supervision of final answers. Therefore, we propose a two-stage learning strategy: First, we manually design a text parser to generate initial supervision on parse graphs and use them to train the information extraction module in SMART. Second, we adopt an iterative learning method to reinforce the information extraction module.

Initial Supervision

Since we do not have the ground-truth parse graphs, we generate parse graph proposals using a hand-designed text parser. The parser extracts nodes, attributes and relations from the text to construct a parse graph.

Attribute Extraction We first extract all numbers of *rate*, *amount* and *total* using pos Tagger by spaCy[1]. Similar to Roy and Roth (2017), we refer to the unit of attribute *total* to be "Num Unit" (short for Numerator Unit), and the unit of attribute *amount* to be "Den Unit" (short for Denominator Unit). The unit of attribute *rate* is therefore "Num Unit per Den Unit". Table 4 shows the attributes and attribute units of an exemplar problem.

Problem: Each kilogram of pears cost 3.65 dollars. How many dollars does mom have to pay for 13 kilograms of pears?	***rate***	***amount***	***total***
	3.65	13	how many
	Num Unit	**Den Unit**	
	dollar	kilometer	

Table 4: The attributes (*rate*, *amount*, *total*), Num Unit, Den Unit of an exemplar problem

Generally, when we have a word marked as "NUM" or "X" (in the case of fractions) by pos tagger, or when the word is "how many" (in the case of interrogative phrase), we check if there's word such as "per" or "each" nearby. If so, we mark the number as *rate* and extract the Num Unit and Den Unit. We then extract *amount* and *total* which are followed by Den Unit and Num Unit respectively.

Node Extraction The next step is to extract the nodes and link attributes to nodes, based on the POS tagging and the dependency parsing. Specifically, we extracts the nouns in the text and treat them as agent nodes. We extract the verbs and their dependents as event nodes. The world node represents the scope of a problem, and is associated with a *total* attribute denoting the total quantities to be covered in the problem. The extraction is conditioned on the problem type. For the type of motion, it means the total distance within the scope of the problem; for the type of price, it is the total money that can be spent; for the type of task completion, it is usually the total amount of tasks to be completed. The world node is extracted based on rules using POS tagger, dependency parser and regular expressions. If there is no "scope" information in the problem, we just place a default world node in the parse graph.

We connect nodes and attributes based on distance and dependency parsing.

Relation Extraction We represent the relations explicitly stated in the problem text via the first-order logic:

- **Variables:** We define a node v to be a variable.
- **Functions:** We consider an attribute to be a function of a node, *i.e.*, Rate(v), Amount(v), Total(v). Moreover, we define two extra functions: a sum function which takes in the same attribute of several nodes and returns their sum, e.g., Sum(Total(v_i), Total(v_j)); a left function, which computes the quantities that haven't been covered by events so far, e.g., Left(Total(S), Total(v_i), Total(v_j)).
- **Predicates:** We view relations as predicates. Predicates take in functions F, and sometimes a value n representing numerical relation (if n is detected by relation extraction, we exclude it from the attribute set). These include: Equal($F(v_i)$, $F(v_j)$); More_than($F(v_i)$, $F(v_j)$, n); Less_than($F(v_i)$, $F(v_j)$, n); Times_of($F(v_i)$, $F(v_j)$, n).

Please refer to the supplementary materials for the complete definitions for functions and predicates.

[1] https://spacy.io/

To mine relations from the text, we first use keyword matching to measure how much a text span is considered to indicate a relation based on keywords (e.g., more than, less than, equals to, times of, left) and then represent the relation as a first-order logic predicate. The predicates are transformed into equations.

Iterative Learning

To further improve the performance of SMART, we propose an iterative learning method: we keep a success buffer, which stores the pseudo gold parse graphs generating correct answers, and a failure buffer, which keeps track of the problems not being solved yet. The pseudo gold parse graphs provided by initial supervision served as the initialization of the success buffer for the first iteration. At each iteration, we first use the success buffer to update the model, and then apply the updated model to the instances in the failure buffer to check if the updated model can solve new problems. The details of the iterative learning method are illustrated in Algorithm 1.

Algorithm 1 Iterative Learning

Input: training set $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$
Success buffer $\mathcal{B}$, Failure buffer $\mathcal{F}$, updated parser θ
▷*Parse Graph Proposal*
for $x_i, y_i \in \mathcal{D}$ **do**
  pg_i = initial_parser (x_i)
  if execute(pg_i) = y_i **then**
    $\mathcal{B} \leftarrow \mathcal{B} \cup \{x_i, pg_i\}$
  else
    $\mathcal{F} \leftarrow \mathcal{F} \cup \{x_i, y_i\}$
▷*Iterative Learning*
while not converge **do**
  for $x_i, pg_i \in \mathcal{B}$ **do**
    $\theta = \theta - \nabla_\theta J(x_i, pg_i)$
  for $x_i, y_i \in \mathcal{F}$ **do**
    pg_i = updated_parser (x_i)
    if execute(pg_i) = y_i **then**
      $\mathcal{B} \leftarrow \mathcal{B} \cup \{x_i, pg_i\}$
      remove $\{x_i, y_i\}$ from $\mathcal{F}$

Experiments

Experimental Setup

Dataset We evaluate our SMART model on the newly curated ASP6.6k dataset. The final dataset is randomly divided into training and test sets of 5,332 and 1,334 problems (i.e, approximately a 80/20 split).

Evaluation Metric We report the answer accuracy of the models: the generated solution is considered correct if it executes to the ground-truth answer. Furthermore, we design an out-of-distribution (OOD) evaluation to examine the models' generalization ability.

Baselines We compare the proposed SMART model with several state-of-the-art neural models for math word problems: MathEN (Wang et al. 2018), Group-ATT (Wang et al. 2019), GTS(Xie and Sun 2019), and Graph2Tree (Zhang et al. 2020).

Results and Analyses

Comparison with State-of-the-art Models Table 5 summarizes the comparison of the answer accuracy on the test set with regard to problem types. The proposed SMART model significantly outperforms all the neural models and beats the state-of-the-art model by nearly 3% in terms of the overall accuracy. More specifically, SMART outperforms the neural models by 3% and 4% on the Motion and Relation problems, while it achieve comparable performance on the Task problems and lower performance on the Price problems.

Model	Overall	Motion	Task	Relation	Price
MathEN	67.8	68.3	70.2	63.3	70.5
GroupATT	67.4	65.2	70.7	63.6	71.5
GTS	76.8	73.2	72.1	76.0	**83.6**
Graph2Tree	76.8	76.9	**79.0**	73.8	78.7
SMART	**79.5**	**79.8**	**79.0**	**77.9**	81.8

Table 5: The answer accuracy on the test set (%).

Model	Overall	Motion	Task	Relation	Price
MathEN	31.7	22.6	28.9	39.9	33.2
GroupATT	35.0	24.0	42.2	42.6	32.7
GTS	45.8	44.5	41.9	49.9	45.3
Graph2Tree	45.1	34.1	47.4	55.1	41.9
SMART	**63.2**	**65.0**	**64.8**	**62.9**	**60.8**

Table 6: The answer accuracy in the OOD evaluation (%). The test set is the 20% longest problems of each type.

Out-of-distribution Evaluation To measure the generalization ability of the models, we conduct an out-of-distribution (OOD) evaluation, where the test set contains more complex problems than the training set. The length of an algebra story problem is a good proxy for its solving complexity. Therefore, we select the longest 20% of problems for each type as the test set and the rest as the training set. Table 6 summarize the answer accuracy of all models in the OOD evaluation. The results show that SMART has better generalization ability. It outperforms the neural models by 17%, revealing SMART's strong ability to reason about more complicated situations even if it is trained on much simpler problems.

Iterative Learning Figure 4 shows the test accuracy versus iterations. Here an iteration is an update on the SMART model based on collected successful samples prior to that iteration, as illustrated in Algorithm 1. We can see that the model improves during iterative learning and begins to converge at the third iteration.

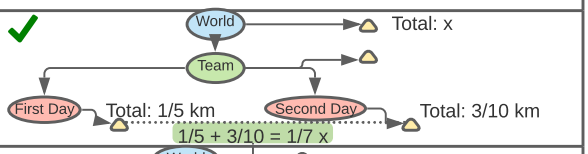
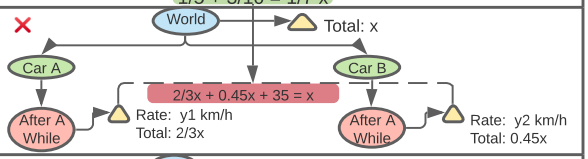
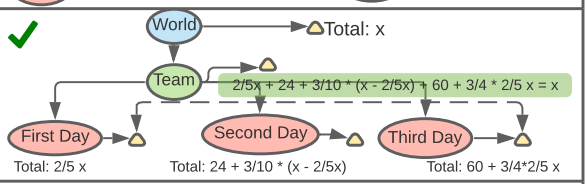
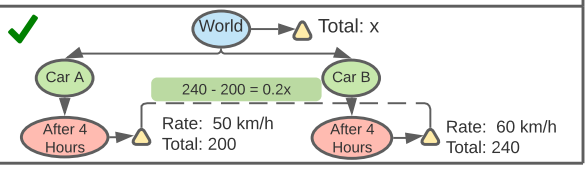
Problem	Expression	GTS	SMART
A road repair team needs to repair a section of road. The first day it repairs (1/5) kilometers, the second day they repairs (3/10) kilometers, and the road they already repairs is (1/7) of the total length. How many kilometers is the total length of the road?	((1/5)+(3/10))/(1/7)	((1/5)+(3/10))/(1/7) ✓	
Car A and car B set off from city A and city B at the same time, and head toward each other. After a while Car A travels (2/3) of the entire journey, and Car B travels 45% of the entire journey. At this time, the two vehicles were 35 kilometers apart. How many kilometers are between two city A and city B?	35/((2/3)+45%-1)	35/(1-(2/3)) ✗	
An engineering team is building a the road. On the first day, it builds (2/5) of the total length. On the next day, it builds 24 meters more than (3/10) of the remaining length. On the third day, it builds 60 meters (3/ 4) the length of the first day, and then they finish. How many meters is the length of the road?	(24+60)/[1-(1 -(2/5))*(3/10) -(2/5)*(3/4)-(2/5)]	(60-24)/ ((1-(2/5)) -(3/4))-24 ✗	
Two cars drives out from city A and city B at the same time. Car A travels 50 kilometers per hour, and car B travels 60 kilometers per hour. After 4 hours, the distance between the two cars was 20% of the total distrance. How many kilometers are the total distance between city A and city B?	(50+60) *4/(1-20%)	50*4 ✗	

Figure 3: Qualitative study of the GTS model and our SMART model. For the first two instances, we visualize the result in the original test set. The last two instances are from the OOD dataset, where the test set has greater problem length in general compared to the training set.

Ablative Study The ablative study in Figure 4 adds the modules sequentially in the inference procedure on the test set. We can see that the sole use of NER only achieves around 20% accuracy. This indicates that reasoning about relations between attributes is needed in most problems. The final result with both the original relation extraction (RE) module and Seq2Seq outperforms the model with just the RE module. This suggests Seq2Seq does detect relations not extracted by RE. However, Seq2seq alone is inferior to RE. It suggests explicit mapping of relations works better than implicit learning of a neural module.

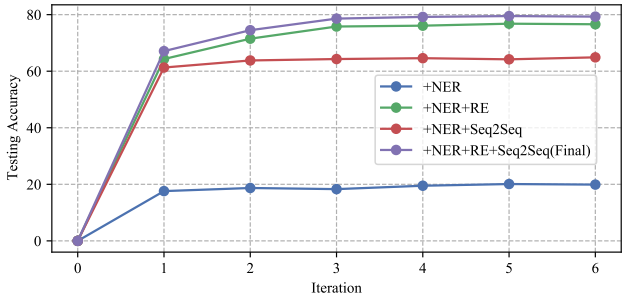


Figure 4: The answer accuracy on the test set across various iterations during the iterative learning. RE denotes the relation extraction in the initial parser. NER denotes the Named Entity Recognition system. Seq2Seq is the model used to detect relations not extracted by the initial parser.

Qualitative Study To further analyze our model's interpretability and generalization ability, we visualize several examples from the test set, as shown in Figure 3. The first two instances are from the original set, while the last two are from the OOD dataset. Both models work well on the first instance, since there's similar data in the training set. When it comes to the third instance, the neural network crashes because it has never observed a problem with more than two events, and does not comprehend the relations. However, the situation model handles well due to event independence. In the fourth instance, the neural network fails to determine the speed of the next car and the relation "20% of the total distance", while SMART is good at both.

However, SMART also makes false predictions. In the second instance, "the two vehicles were 35 kilometers apart" is ambiguous since we do not know if the cars have met each other yet. Therefore, SMART gives a negative value for the total length. In the future, error analysis is needed in the situation model so that when an implausible answer is given, the situation model seeks for an alternative solution.

Conclusions

In this work we propose a situation model with attributed grammar for algebra story problems. The experiment results on ASP6.6k indicate our model outperforms state-of-the-art models and shows better interpretability and generalization ability. Future work includes creating an intelligent tutor that help students develop better problem solving skills.

Ethical Impact

This paper implements the situation model applied by humans to solve algebra story problems on the setting of artificial intelligence. Based on the model, educators can design intelligent tutors to guide students in mathematical learning.

Acknowledgement

This work reported herein is supported by ARO W911NF1810296, DARPA XAI N66001-17-2-4029, and ONR MURI N00014-16-1-2007.

References

Abedi, J.; and Lord, C. 2001. The language factor in mathematics tests. *Applied measurement in education* 14(3): 219–234.

Amini, A.; Gabriel, S.; Lin, S.; Koncel-Kedziorski, R.; Choi, Y.; and Hajishirzi, H. 2019. MathQA: Towards Interpretable Math Word Problem Solving with Operation-Based Formalisms. In *NAACL-HLT*.

Bakman, Y. 2007. Robust Understanding of Word Problems with Extraneous Information.

Bjork, I. M.; and Bowyer-Crane, C. 2013. Cognitive skills used to solve mathematical word problems and numerical operations: A study of 6-to 7-year-old children. *European journal of psychology of education* 28(4): 1345–1360.

Chen, Y.; Huang, S.; Yuan, T.; Qi, S.; Zhu, Y.; and Zhu, S.-C. 2019. Holistic++ scene understanding: Single-view 3D holistic scene parsing and human pose estimation with human-object interaction and physical commonsense. In *Proceedings of the IEEE International Conference on Computer Vision*.

Chiang, T.-R.; and Chen, Y.-N. 2019. Semantically-Aligned Equation Generation for Solving and Reasoning Math Word Problems. *ArXiv* abs/1811.00720.

Coquin-Viennot, D.; and Moreau, S. 2007. Arithmetic problems at school: when there is an apparent contradiction between the situation model and the problem model. *The British journal of educational psychology* 77 Pt 1: 69–80.

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

Gernsbacher, M. A. 2013. *Language comprehension as structure building*. Psychology Press.

Greeno, J. G. 1989. Situation models, mental models, and generative knowledge.

Han, F.; and Zhu, S.-C. 2005. Bottom-up/top-down image parsing by attribute graph grammar. In *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, volume 2, 1778–1785. IEEE.

Hegarty, M.; Mayer, R. E.; and Monk, C. A. 1995. Comprehension of arithmetic word problems: A comparison of successful and unsuccessful problem solvers. *Journal of educational psychology* 87(1): 18.

Hinsley, D.; Hayes, J. R.; and Simon, H. A. 1977. From words to equations. *P. Carpenter and M. Just, eds., Cognitive Processes in Comprehension. Hillsdale, N J : Erlbaum.* .

Hong, Y.; Li, Q.; Ciao, D.; Huang, S.; and Zhu, S.-C. 2021. Learning by Fixing: Solving Math Word Problems with Weak Supervision. In *AAAI Conference on Artificial Intelligence*.

Hosseini, M. J.; Hajishirzi, H.; Etzioni, O.; and Kushman, N. 2014. Learning to Solve Arithmetic Word Problems with Verb Categorization. In *EMNLP*.

Huang, D.; Liu, J.; Lin, C.-Y.; and Yin, J. 2018a. Neural Math Word Problem Solver with Reinforcement Learning. In *COLING*.

Huang, D.; Shi, S.; Lin, C.-Y.; and Yin, J. 2017. Learning Fine-Grained Expressions to Solve Math Word Problems. In *EMNLP*.

Huang, D.; Shi, S.; Lin, C.-Y.; Yin, J.; and Ma, W.-Y. 2016. How well do Computers Solve Math Word Problems? Large-Scale Dataset Construction and Evaluation. In *ACL*.

Huang, S.; Qi, S.; Zhu, Y.; Xiao, Y.; Xu, Y.; and Zhu, S.-C. 2018b. Holistic 3d scene parsing and reconstruction from a single rgb image. In *Proceedings of the European Conference on Computer Vision (ECCV)*.

Jiang, C.; Qi, S.; Zhu, Y.; Huang, S.; Lin, J.; Yu, L.-F.; Terzopoulos, D.; and Zhu, S.-C. 2018. Configurable 3d scene synthesis and 2d image rendering with per-pixel ground truth using stochastic grammars. *International Journal of Computer Vision* .

Kintsch, W.; and Greeno, J. G. 1985. Understanding and solving word arithmetic problems. *Psychological review* 92(1): 109.

Kintsch, W.; and Walter Kintsch, C. 1998. *Comprehension: A paradigm for cognition*. Cambridge university press.

Knuth, D. E. 1990. The genesis of attribute grammars. In *Attribute Grammars and Their Applications*, 1–12. Springer.

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.

Kushman, N.; Zettlemoyer, L.; Barzilay, R.; and Artzi, Y. 2014. Learning to Automatically Solve Algebra Word Problems. In *ACL*.

Leiss, D.; Schukajlow, S.; Blum, W.; Messner, R.; and Pekrun, R. 2010. The Role of the Situation Model in Mathematical Modelling—Task Analyses, Student Competencies, and Teacher Interventions. *Journal für Mathematik-Didaktik* 31: 119–141.

Lin, L.; Gong, H.; Li, L.; and Wang, L. 2009. Semantic event representation and recognition using syntactic attribute graph grammar. *Pattern Recognition Letters* 30(2): 180–186.

Ling, W.; Yogatama, D.; Dyer, C.; and Blunsom, P. 2017. Program Induction by Rationale Generation: Learning to

Solve and Explain Algebraic Word Problems. *ArXiv* abs/1705.04146.

Liu, X.; Zhao, Y.; and Zhu, S.-C. 2014. Single-View 3D Scene Parsing by Attributed Grammar. *2014 IEEE Conference on Computer Vision and Pattern Recognition* 684–691.

Mayer, R. E. 1981. Frequency norms and structural analysis of algebra story problems into families, categories, and templates. *Instructional Science* 10(2): 135–175.

Meurer, A.; Smith, C. P.; Paprocki, M.; Čertík, O.; Kirpichev, S. B.; Rocklin, M.; Kumar, A.; Ivanov, S.; Moore, J. K.; Singh, S.; Rathnayake, T.; Vig, S.; Granger, B. E.; Muller, R. P.; Bonazzi, F.; Gupta, H.; Vats, S.; Johansson, F.; Pedregosa, F.; Curry, M. J.; Terrel, A. R.; Roučka, v.; Saboo, A.; Fernando, I.; Kulal, S.; Cimrman, R.; and Scopatz, A. 2017. SymPy: symbolic computing in Python. *PeerJ Computer Science* 3: e103. ISSN 2376-5992. doi:10.7717/peerj-cs.103. URL https://doi.org/10.7717/peerj-cs.103.

Mitra, A.; and Baral, C. 2016. Learning To Use Formulas To Solve Simple Arithmetic Problems. In *ACL*.

Nathan, M. J.; Kintsch, W.; and Young, E. 1992. A theory of algebra-word-problem comprehension and its implications for the design of learning environments. *Cognition and instruction* 9(4): 329–389.

Nathan, M. J.; and Young, E. 1990. Thinking situationally: Results with an unintelligent tutor for word algebra problems.

Nesher, P.; Hershkovitz, S.; and Novotna, J. 2003. Situation model, text base and what else? Factors affecting problem solving. *Educational Studies in Mathematics* 52(2): 151–176.

Park, S.; Nie, B. X.; and Zhu, S.-C. 2016a. Attribute and-or grammar for joint parsing of human attributes, part and pose. *arXiv preprint arXiv:1605.02112* .

Park, S.; Nie, X.; and Zhu, S.-C. 2016b. Attribute And-Or Grammar for Joint Parsing of Human Attributes, Part and Pose. *ArXiv* abs/1605.02112.

Park, S.; and Zhu, S.-C. 2015. Attributed Grammars for Joint Estimation of Human Attributes, Part and Pose. *2015 IEEE International Conference on Computer Vision (ICCV)* 2372–2380.

Qi, S.; Zhu, Y.; Huang, S.; Jiang, C.; and Zhu, S.-C. 2018. Human-centric indoor scene synthesis using stochastic grammar. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 5899–5908.

Raudszus, H.; Segers, E.; and Verhoeven, L. 2019. Situation model building ability uniquely predicts first and second language reading comprehension. *Journal of Neurolinguistics* 50: 106–119.

Reusser, K. 1990. From text to situation to equation: cognitive simulation of understanding and solving mathematical word problems.

Robaidek, B.; Koncel-Kedziorski, R.; and Hajishirzi, H. 2018. Data-Driven Methods for Solving Algebra Word Problems. *ArXiv* abs/1804.10718.

Roy, S.; and Roth, D. 2017. Unit Dependency Graph and Its Application to Arithmetic Word Problem Solving. *ArXiv* abs/1612.00969.

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*.

Straková, J.; Straka, M.; and Hajic, J. 2019. Neural Architectures for Nested NER through Linearization. In *ACL*.

Thorndike, R. L. 1953. Who belongs in the family? *Psychometrika* 18: 267–276.

van Dijk, T. A.; and Kintsch, W. 1983. Strategies of discourse comprehension.

Wang, L.; Wang, Y.; Cai, D.; Zhang, D.; and Liu, X. 2018. Translating Math Word Problem to Expression Tree. In *EMNLP*.

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. *Proceedings of the AAAI Conference on Artificial Intelligence* 33(01): 7144–7151.

Wang, S.; Joo, J.; Wang, Y.; and Zhu, S.-C. 2013. Weakly supervised learning for attribute localization in outdoor scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 3111–3118.

Wang, Y.; Liu, X.; and Shi, S. 2017a. Deep Neural Solver for Math Word Problems. 845–854. Copenhagen, Denmark: Association for Computational Linguistics.

Wang, Y.; Liu, X.; and Shi, S. 2017b. Deep Neural Solver for Math Word Problems. In *EMNLP*.

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

Yu-hui, M.; Ying, Z.; Guang-zuo, C.; Yun, R.; and Ronghuai, H. 2010. Frame-Based Calculus of Solving Arithmetic Multi-Step Addition and Subtraction Word Problems. *2010 Second International Workshop on Education Technology and Computer Science* 2: 476–479.

Zhang, J.; Wang, L.; Lee, R. K.-W.; Bin, Y.; Shao, J.; and Lim, E.-P. 2020. Graph-to-Tree Learning for Solving Math Word Problems. *ACL 2020* .

Zhou, L.; Dai, S.; and Chen, L. 2015. Learn to Solve Algebra Word Problems Using Quadratic Programming. In *EMNLP*.

Zwaan, R. A.; Magliano, J. P.; and Graesser, A. C. 1995. Dimensions of situation model construction in narrative comprehension. *Journal of experimental psychology: Learning, memory, and cognition* 21(2): 386.