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

Yining Hong,[1] **Qing Li,** [1] **Ran Gong,** [1] **Daniel Ciao,** [1] **Siyuan Huang,** [1] **Song-Chun Zhu,** [1]
[1] 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, requiring an explicit understanding of real-world situations and strong mathematical reasoning ability. Previous neural solvers of math word problems lack the interpretation of the world and events, thus failing to generalize to more sophisticated situations. In this paper, we present *SASPAR*, a Situation model for Algebra Story Problems based on Attributed grammar. We propose a parser based on a Markov random field to generate a parse graph, an instance of the grammar. An iterative learning paradigm is applied to update the parser. We create a new dataset for algebra story problems called *Story6.6k*. Experimental results on Story6.6k show that the proposed SASPAR model has better interpretability and outperforms all previous neural solvers by a large margin. To test these models' generalization capability, we design an out-of-distribution (OOD) test where the test set has more complex problems than the train set. Our model exceeds state-of-the-art neural models by 17% in the OOD test, demonstrating its superior generalization ability.

Introduction

Algebra Story Problems, depicted by Hinsley, Hayes, and Simon (1977) as "those twentieth-century fables", remain an important challenge in artificial intelligence. An algebra story problem typically describes a real-world situation and inquires about an unknown attribute in the situation. It goes beyond arithmetic expressions since one has to comprehend the situation and recognize the goal in the problem, then come up with a solution for it, which is a key aspect of human intelligence (Nathan, Kintsch, and Young 1992a). 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 over real-world tasks. Therefore, algebra story problems should be distinguished from other types of math word problems such as number problems and geometry problems.

The *situation model* (van Dijk and Kintsch 1983) is used to represent agents, actions, and events when humans solve algebra story problems (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 situation model instead of the problem text itself. As shown in Figure 1, the situation model often interacts with a math conceptual model, which in turn embeds relations in the situation model.

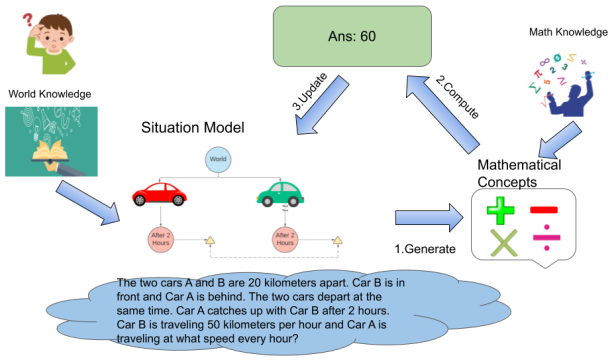


Figure 1: The process of solving algebra story problems: First, we build a situation model from text comprehension. Then we apply math knowledge on the situation model to compute an answer. If we fail to generate a correct solution, we can adjust our situation model until we get one.

Researchers have recently focused on using neural networks to solve math word problems (Wang, Liu, and Shi 2017a; Huang et al. 2018; Wang et al. 2018; Xie and Sun 2019). They usually use end-to-end neural models (*e.g.*, Seq2Seq, Seq2Tree) to directly translate the problem text into an expression, which is then executed to get the final answer. This is a deviation from how humans solve algebra story problems using situation models. Such neural models suffer from several drawbacks:

- Lack of generalization to more sophisticated scenarios. If the number of events in the problem exceeds the neural models' observations before, the solver crashes. However, since the situation model recursively handles the events with grammar, it can tackle the problem well. It also benefits from the fact that "people tend to remember the situation model but not the text representation"(Bransford, Barclay, and Franks 1972; Barclay 1973) — thus, the situation model embodies better memory.
- Lack of actual understanding of the problem. As stated by

van Dijk and Kintsch (1983), if a model fails to perceive a situation where particular objects have the attributes or relations indicated by the text, it fails to understand the text itself.

- Lack of interpretability. The neural network only generates an equation, which can be hard to understand without the intermediate problem-solving process. As shown in Figure 3, the annotated equation can be "(24+60)/[1-(1-(2/5))*(3/10)-(2/5)*(3/4)-(2/5)]", which barely makes sense to a human. The situation model, on the other hand, is interpretable to humans. The representation of events, objects, and relations between them naturally provide the solving process we need for education.

To build a situation model for algebra story problems, we observe that a *parse graph* derived from Attributed Grammar (Knuth 1990; Liu, Zhao, and Zhu 2014; Park and Zhu 2015; Park, Nie, and Zhu 2016b) possesses all the features of a situation model by its nature. Specifically, we propose SASPAR, a Situation Model for solving Algebra Story Problems via Attributed grammar, which represents the world, agents, and events as nodes in a hierarchical manner. We also design the production rules for the decomposition of the graph nodes. To better incorporate mathematical reasoning, attributes including rate, amount, and total quantity are linked to the nodes. The reasoning process is thus formulated as the relation propagation between the node attributes.

While previous datasets of math word problems often fuse different types of problems, we seek to build a dataset of algebra story problems only. Math23K (Wang, Liu, and Shi 2017b) is the most frequently-used math problem dataset in recent years. It contains several problem types, including number problems, geometry problems, story problems *etc*. We build our dataset upon Math23K following precisely the criteria by Mayer (1981) to collect and categorize algebra story problems.

Since the parse graph annotation can be time-consuming, we design an initial graph parser to propose the first source of supervision for learning. Specifically, we extract nodes and attributes of the parse graph according to the Markov random field (MRF). The numerical relations on attributes are represented as first-order logic. Due to the inflexibility of the initial parser, we simply use it to provide pseudo-gold parse graphs (i.e., parse graphs that can be executed to get the ground-truth answer) for the semi-supervised learning of an updated graph parser. The updated parser includes a Named Entity Recognition (NER) system to extract nodes and attributes, together with a Seq2Seq model to update relations. We propose an iterative learning method where pseudo-gold parse graphs at each iteration augment the supervision for the next learning iteration.

We conduct experiments on the constructed Story6.6k dataset. The proposed situation model SASPAR outperforms all neural network baselines by a large margin. Moreover, it shows stronger generalization ability in an out-of-distribution test, where the test problems are more complex than those in the training set. A qualitative study also suggests that our model achieves more interpretability and generalization ability than the neural models.

Related Works

Situation Model A situation model has 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. Kintsch and Greeno (1985) use a situation model to analyze processing requirements and difficulties of algebra word problems. Nathan, Kintsch, and Young (1992b) 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.

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 ranging from scenes(Han and Zhu 2005; Wang et al. 2013), video events(Lin et al. 2009), human poses(Park, Nie, and Zhu 2016a) to indoor layouts(Qi et al. 2018), *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. Besides, attributes in a parse graph can be propagated in a controlled and formal way. In (Lin et al. 2009; Park, Nie, and Zhu 2016a), authors use soft constraints in the form of MRF potentials, which is adopted by this paper. This paper also utilizes hard constraints for mathematical reasoning.

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 prob-

lems using neural networks (Ling et al. 2017; Wang, Liu, and Shi 2017a; Huang et al. 2018; Robaidek, Koncel-Kedziorski, and Hajishirzi 2018; Wang et al. 2018, 2019; Chiang and Chen 2019; Xie and Sun 2019; Zhang et al. 2020). 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 strength of both symbolic reasoning and neural networks, where we use neural modules to update symbolic representations.

Dataset

We collect our dataset (Story6.6k) from the commonly used Math23K dataset. To select and categorize algebra story problems, we first compute term frequency–inverse document frequency (TF-IDF) for the entire corpus. Then each problem is represented as a TF-IDF vector. Next, we use k-means clustering to group problems into different categories. We use the elbow method to find the optimal K. To further remove noise, we use keywords filtering to remove problems that do not belong to the group. Since we want the problem itself to describe a situation or tell a story, we make sure our selection strategy is consistent with the following criterion inspired by (Mayer 1981):

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

Following the collection criterion, we obtain a dataset of 6666 problems spanning from 4 typical types of algebra story problems (Mayer 1981; Nathan, Kintsch, and Young 1992b): motion, price, relation and task completion.

Here, we provide a brief summary of the problem types:
Motion: problems involve traveling and require understanding of per time rate.
Task Completion: 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.
See supplementary for details about dataset statistics, examples of each type and keyword filtering techniques.

Situation Model

In this section, we introduce a situation model for algebra story problems via attributed grammar (SASPAR). We claim a situation model satisfies 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 has all the above properties. Therefore, we implement our situation model based on attributed parse graphs. An attributed parse graph combines i) an instantiation of a context free grammar(CFG), ii) contextual relations among nodes. In algebra story problems, CFG represents the hierarchical decomposition from the world of a problem (top-level) to the events (bottom-level). Contextual relations encode the relations between nodes, as well as the numerical relations between attributes of nodes.

We denote an attributed grammar of SASPAR by a 6-tuple:

$$\mathcal{G} =< S, V, R, A, E, P > \tag{1}$$

where S is the root node, and V is the vertex set. A is the attribute set on V. R is a set of production rules, E denotes the relations, and P is the probability formulation.

A parse graph is a configuration of the attributed grammar.

$$pg = (V_{pg}, A_{pg}, E_{pg}) \tag{2}$$

Figure 2 shows an example of a parse graph in the situation model.

Vertex Set V can be decomposed into a finite set of non-terminal nodes and terminal nodes: $V = V_{NT} \cup V_T$.

- Non-terminal nodes V_{NT} represent the decomposition of a parent node to its child nodes using CFG. In algebra story problems, there are three types of non-terminal nodes: i) **World**: the scope of the world the text is talking about (e.g., the distance between two cities; or a task to be completed). It is also the root note S of the AOG. ii) **Agents** or **Agent**: the agents performing events in the scope of the world. (e.g., a car; a person). iii) **Events**: The events performed by the agents in the world.
- Terminal nodes V_T are **Event**-nodes, which are decomposed from **Events**-nodes.

Production rules R for the grammar is defined as:

$$\begin{aligned} \text{World} &\rightarrow \text{Agents} \\ \text{Agents} &\rightarrow \text{Agents + Agent} \mid \text{Agent} \\ \text{Agent} &\rightarrow \text{Events} \\ \text{Events} &\rightarrow \text{Events + Event} \mid \text{Event} \end{aligned}$$

Attribute set A denotes attributes linked to nodes. Inspired by Nathan, Kintsch, and Young (1992b); Mayer (1981); Roy and Roth (2017), we define three kinds of attributes: i) **Rate**: 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**: measurement of units of rate quantities (*e.g.*, hour). iii) **Total**: the total quantity which equals to rate$\times$ amount (*e.g.*, distance).
Contextual relations E contain two kinds of relations defined on Markov random field (MRF): i) Implicit Relation E_{pgi}: the closeness between a node and its child nodes or the closeness between a node and its attribute; and ii) Explicit Relation E_{pge}: numerical relations between attributes, represented by first-order logic. Note that there is a commonsense relation that applies to each node: Total = Rate $\times$ Amount.

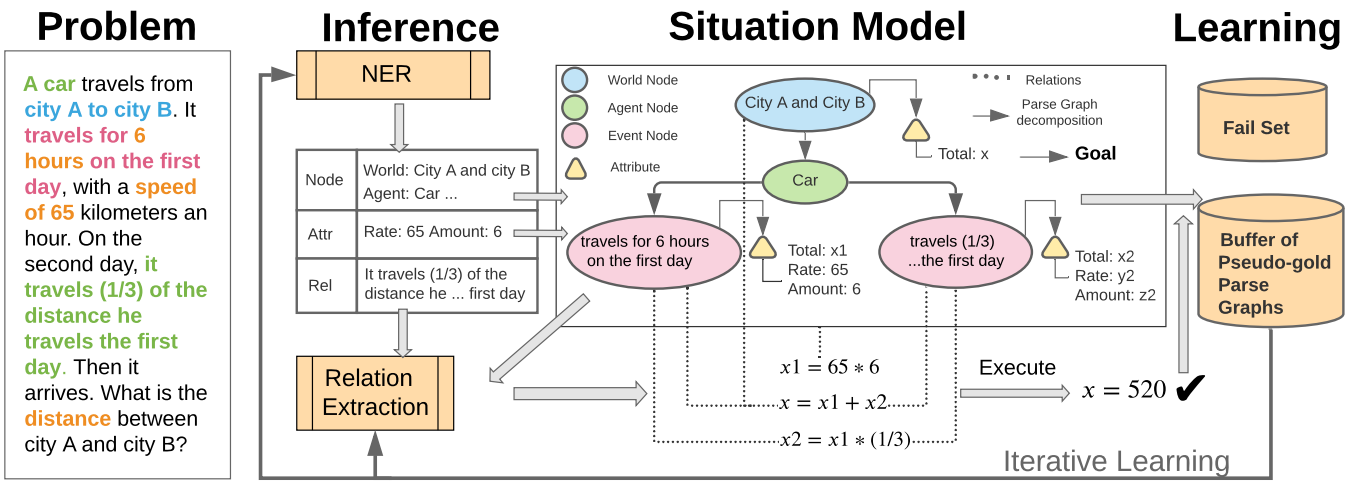


Figure 2: Overview of our SASPAR model. The Named Entity Recognition (NER) module extracts the nodes , attributes, as well as relations spans from the text, and construct a parse graph using Attributed Grammar. The Relation Extraction module uses the relation span and the parse graph already constructed 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 fail set to be updated at the following iterations.

Probability formulation P Given an algebra story problem x, we aim to derive the final answer using an intermediate parse graph pg. The prior probability of pg given a math problem x is formulated as a Gibbs distribution:

$$\begin{aligned} p(pg \mid x) &= \frac{1}{Z}\exp\{-\mathcal{E}(pg \mid x)\} \\ &= \frac{1}{Z}\exp\{-\mathcal{E}(V_{pg} \mid x) - \mathcal{E}(A_{pg} \mid x) \\ &\quad - \mathcal{E}(E_{pgi} \mid x) - \mathcal{E}(E_{pge} \mid x)\} \end{aligned} \tag{3}$$

Where $\mathcal{E}(pg \mid x)$ is the energy function of a parse graph. $\mathcal{E}(V_{pg} \mid x), \mathcal{E}(A_{pg} \mid x), \mathcal{E}(E_{pgi} \mid x), \mathcal{E}(E_{pge} \mid x)$ are energy functions of the vertex nodes, attributes, implicit relations and explicit relations respectively. The energy functions can be defined differently depending on the situation model one designs. In the following sections, we define energy functions separately according to different models.

Parse Graph Proposal

Since we do not have the annotations for ground-truth parse graphs, we first generate parse graph proposals using an initial graph parser. The parser proposes nodes, attributes and relations to construct a parse graph. Note that the initial parser only provide pseudo annotations for learning, thus the energy functions in this section can be straightforward.

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 1 shows the attributes and attribute units of an exemplar problem.

[1]https://spacy.io/

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 1: 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. The following energy functions are for proposing node candidates:

$$\mathcal{E}(V_{pg}|x) = \sum_{v \in V_{pg}} \mathcal{E}(v \mid x) = \sum_{v} -\log(l_{pos}(v) + l_{dep}(v)) \tag{4}$$

l_{pos} denotes the pos tagging constraints and l_{dep} denotes the dependency parsing constraints. Specifically, for an agent node, l_{pos} encodes the score of a word being a noun given pos tagging probabilities, and l_{dep} equals zero. For an event node, l_{pos} encodes the score of a word being a verb, and l_{dep} is the score of a span which is also the sub parse tree with the verb being the root note. The world node represents the scope of a problem, and has a Total attribute denotes the total quantities to be covered in the problem. The extraction is conditioned on the problem type. For motion, it means the total distance within the scope of the problem; for task, it is usually the total amount of tasks to be completed; for price, it is the total money that can be spent. The world node is extracted based on rules using pos tagger, dependency parser and regular expressions. If there's no "scope" information in the problem, we just place a default world node in the graph.

To link nodes to each other, we have the implicit relations E_{pgi} between nodes as MRF:

$$\mathcal{E}(E_{pgi} \mid x) = -\sum_{v_a, v_e} (l_{\text{dis}}(v_a, v_e) + l_{\text{dep}}(v_a, v_e)) - \sum_{v_i \in V, a_i} (l_{\text{dis}}(v_i, a_i) + l_{\text{dep}}(v_i, a_i)) \quad (5)$$

This function measures the closeness between a parent node and its child nodes, as well as the closeness between a node and its attributes. v_a is an agent node and v_e is an event node. a_i represents an attribute value of vertex v_i. The l_{dis} function takes into account two kinds of distance: the word distance between two nodes in the problem text, as well as the distance (number of links) between them in the dependency parse tree. The l_{dep} denotes the dependency parsing constraints (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 material for the list of constraints, as well as more detailed definitions of the energy functions.

Explicit Relation Extraction We see explicit numerical relations in math problems as first-order logic predicates. A numerical relation has two parts: relation span r that expresses relation using words from the problem text; and predicates which translate the relation span into logic forms.

- **Variables** We define a node v to be a variable.
- **Functions** We consider an attribute to be a function $F(v)$ of a node . These include: **Rate(**v**), Amount(**v**), Total(**v**)**. Moreover, we define two extra functions. A sum function which can take in the same attribute of several nodes: e.g., **Sum(Total(**v_i**), Total(**v_j**))**. This is usually used for calculating the sum of quantities of events completed so far. Another function is left function, e.g., **Left(Total(**S**), Total(**v_i**), Total(**v_j**))**. This is used for denoting the quantities haven't been covered by events so far, and is left in the total quantities of the root node (i.e., **Total(**S**)**)
- **Predicates** We view numerical relations of a math problem as predicates. Predicates take in functions F above, and sometimes include 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**)**. See supplementary for complete definitions and examples for functions and predicates.

We use the following energy function for explicit relation extraction:

$$\mathcal{E}(E_{pge} \mid x) = -\sum_{r} \sum_{v_i} (l_{key}(r) + l_{\text{dis}}(v_i, r)) \quad (6)$$

where l_{key} denotes the keyword matching function, measuring how much a span is considered to indicate relations based on keywords (e.g., more than, less than, equals to, times of, left...). v_i is a node associated with relation span r. l_{dis} depends on the word distance and dependency distance. We transform r to a predicate using keywords matching.

Goal Recognition Intuitively, a goal of the problem (*i.e.*, what the problem inquires about) is a function expressed in an interrogative phrase. Sometimes, the goal can be the value n when the problem asks about a relation.

Execution Note that a relation in the form of a predicate can be easily transformed to a system of equations and solved for the unknowns. The goal is one of the unknowns.

Parse Graph Update

The initial graph parser can get fragile and fail to generalize due to the unstableness of dependency parser and pos tagger, as well as the inflexiblity of keyword matching. As learning can be viewed as the modification of situation models(van Dijk and Kintsch 1983), we propose an *updated graph parser* to make modifications. The updated parser includes a Named Entity Recognition (NER) system to extract nodes, attributes and relation spans, as well as a Seq2Seq model which takes in the relation span and outputs the equation. We consider a parse graph that can generate the right answer to be a $pseudo\ gold\ pg$. Such graphs generated by the initial parser serve as the first source of supervision for the updated parser.

NER-Based Parse Graph Update For Named Entity Recognition system, we have 7 categories of entities: World, Agent, Event, Rate, Amount, Total, Relation Span. Specifically, we have one NER model to extract the attributes, and another one for extraction of nodes and relations spans. 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.

Each node and attribute output by NER system is assigned a probability. Therefore, we can have the energy functions of nodes and attributes respectively as:

$$\mathcal{E}(V_{pg}|x) = \sum_{v \in V_{pg}} \mathcal{E}(v \mid x) = \sum_{v} -\log p_{ner}(v|x) \quad (7)$$

$$\mathcal{E}(A_{pg}|x) = \sum_{a \in A_{pg}} \mathcal{E}(a \mid x) = \sum_{a} -\log p_{ner}(a|x) \quad (8)$$

where p_{ner} is the probability of a node/attribute being assigned a specific entity category in the NER output. We choose a span output by NER as a node or attribute candidate, and the category of the highest probability to be the type of the node or attribute. Note that even though a span is output by NER, it is not necessarily a part in our parse graph. While Equation 5 defines how the nodes and attributes are linked to each other, it filters node and attribute candidates.

Seq2Seq-Based Relation Update For explicit numerical relation extraction, we first use the relation extraction in the initial parser to acquire part of the relations. However, the parser is partially based on keywords matching, which lacks generalization. To achieve better generalization ability, we use a Seq2Seq model to map a relation span to an equation.

Different from previous neural solvers which take the full problem text as input, and output a single expression, our input is the text span (output by NER) that indicates a certain relation, concatenated by nodes and attributes extracted

before (in the form of plain text). Our output is one equation that is translated from the relation span. If a problem has several relation spans, they are handled separately. Note that the predicates defined before can be easily converted to equations, and vice versa.

The energy function for the explicit relations E_{pge} becomes:

$$\begin{aligned}\mathcal{E}(E_{pge} \mid x) = &-\sum_{r} \log p_{ner}(r|x) \\ &-\sum_{r}\sum_{v_i} l_{\text{dis}}(v_i, r) \\ &-\sum_{r}\sum_{v_i} \log p_{seq}(r_e|r, v_i) \end{aligned} \quad (9)$$

where $p_{ner}(r|x)$ is the NER probability of a span assigned as a relation span. v_i is a node associated with the relation text r, and r_e is the equation translated from r by Seq2Seq. $p_{seq}(r_e|r)$ is the probability of an equation generated by Seq2Seq given a relation span. l_{dis} measures the word distance between a node and the relation span in the problem text.

Inference & Learning

Inference The inference task is equivalent to finding the most probable parse graph pg from the constructed grammar model for given problem x.

$$pg^* = \arg\max_{pg} p(pg \mid x) \quad (10)$$

in which $p(pg \mid x)$ is defined in Equation 3.

It's unrealistic to enumerate all the possible pgs. Therefore, our parsing process is partially greedy:

- Candidates proposal for attributes and nodes using either the initial parser or NER.
- Nodes and attributes filtering and connection by jointly optimizing Equation 5, Equation 4 (Initial Parser) or Equation 7 together with Equation 8 (Updated Parser).
- Relation extraction based on both the relation span and the nodes and attributes, as in Equation 6 or Equation 9.

Iterative Learning We propose an iterative learning paradigm where $pseudo\ gold\ pg$s are put into a buffer. The parse graphs that can not generate the right answer are placed in a fail set. At each iteration, we use the buffer to train the model, and update the instances in the fail set. The algorithm is illustrated in Algorithm 1. The $pseudo\ gold\ pg$s generated by the initial parser serve as the source of supervision for the first iteration.

Algorithm 1 Iterative Learning

Input: training set $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$
Buffer $\mathcal{B}$, Fail Set $\mathcal{F}$, 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 SASPAR model on the new Story6.6k dataset. The test set contains 1334 randomly sampled instances (20% of the whole dataset).

Evaluation Metric We evaluate the model performance with answer accuracy, where the generated solution is considered correct if it executes to the ground-truth answer. Furthermore, we design an OOD (out-of-distribution) test to examine the models' generalization ability.

Baselines We compare the proposed SASPAR model with several fully-supervised neural baselines and state-of-the-art models for math word problems. MathEN(Wang et al. 2018) and Group-ATT(Wang et al. 2019) are two models based on Seq2Seq. GTS(Xie and Sun 2019) is a novel Seq2Tree model. Graph2Tree (Zhang et al. 2020) is the state-of-the-art neural model for math problems.

Results and Analyses

Test Accuracy Table 2 summarizes the comparison of test answer accuracy with the neural solvers with regard to problem types. The proposed SASPAR model outperforms all the neural models and beats the state-of-the-art model by 3%. The updated parser has 15% higher accuracy than the initial parser, demonstrating the strength of iterative learning.

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
SASPAR-IP	64.3	57.8	72.2	63.6	70.4
SASPAR-UP	**79.5**	**79.8**	**79.0**	**77.9**	81.8

Table 2: Test Accuracy (%). IP denotes Initial Parser, and UP denotes updated parser.

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
SASPAR-IP	50.9	45.2	54.3	50.5	52.6
SASPAR-UP	**63.2**	**65.0**	**64.8**	**62.9**	**60.8**

Table 3: OOD (out-of-distribution) Test Accuracy (%). The test set is the 20% longest problems of each type.

Generalization Ability To measure the generalization ability of the models, we conduct OOD (out-of-distribution) test, where the test set contains more complex data than

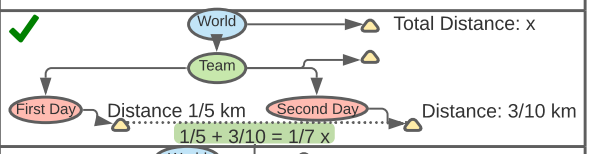
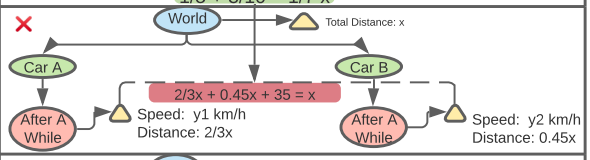
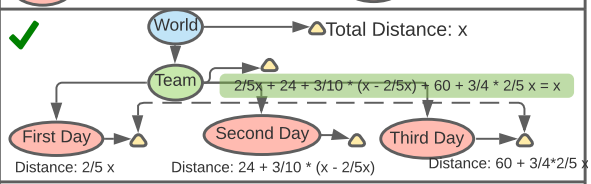
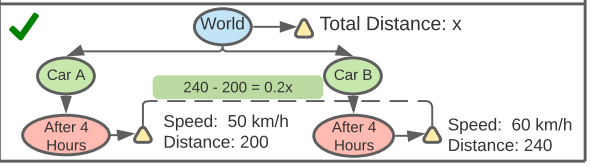
Problem	Neural Network Annotation	Neural Network Output	SASP Output
A road repair team needs to repair a section of road. The first day it repairs (1/5) kilometers, the second day they repairs (3/10) kilometers, and the road they already repairs is (1/7) of the total length. How many kilometers is the 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 SASPAR 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.

the training set. The length of an algebra story problem is a good proxy for its solving complexity. Therefore, we use the longest 20% of problems of each type as the test set and the rest as the training set. Table 3 shows SASPAR has better generalization ability. It outperforms the neural models by 17%, revealing SASPAR's ability to reason about more complicated situations even it's trained on much simpler data.

Iterative Learning Figure 4 shows the test accuracy with iterations. We can see that the model improves during iterative learning. It also begins to converge at the third iteration.

Ablative Study The ablative study in Figure 4 adds the modules sequentially in the inference procedure on test set. We can see that the sole use of NER achieves only $\sim$ 20% accuracy. This indicates that reasoning about relations among 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.

Qualitative Study

To further analyze our model's comprehension and generalization ability, we visualize several examples during testing, 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

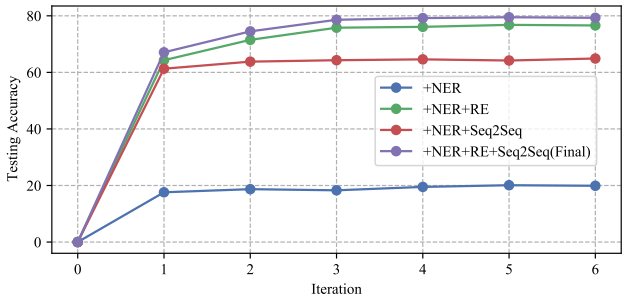


Figure 4: Ablative Study across Iterations. RE denotes the relation extraction in the initial parser. NER denotes the Named Entity Recognition system. Seq2Seq is model used to detect relations not extracted by the initial parser.

the next car and the relation "20% of the total distance" , in the meantime SASPAR is good at both.

However, SASPAR also makes false predictions. In the second instance, the "two vehicles are 35 kilometers apart" is ambiguous since we do not know if the cars have met each other yet. Therefore, SASPAR gives a negative value of 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 and Discussions

In this work we propose a situation model with attributed grammar for algebra story problems. The experiment results on story6.6k indicate our model outperforms state-of-the-art models and shows better interprebility and generalization ability. Future work includes creating 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.

References

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

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

Barclay, J. R. 1973. The role of comprehension in remembering sentences.

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.

Bransford, J. D.; Barclay, J. R.; and Franks, J. J. 1972. Sentence memory: A constructive versus interpretive approach.

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.* .

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. 2018. 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*.

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.

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

Nathan, M.; Kintsch, W.; and Young, E. 1992a. A Theory of Algebra-Word-Problem Comprehension and Its Implications for the Design of Learning Environments.

Nathan, M. J.; Kintsch, W.; and Young, E. 1992b. 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*.

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 Rong-huai, 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.