The 33rd Conference on Computational Linguistics and Speech Processing (ROCLING 2021)
Taoyuan, Taiwan, October 15-16, 2021. The Association for Computational Linguistics and Chinese Language Processing

Mining Commonsense and Domain Knowledge from Math Word Problems

Shih-Hung Tsai, Chao-Chun Liang, Hsin-Min Wang, Keh-Yih Su
Institute of Information Science, Academia Sinica, Taiwan
{doublebite,ccliang,whm,kysu}@iis.sinica.edu.tw

Abstract

Current neural math solvers learn to incorporate commonsense or domain knowledge by utilizing pre-specified constants or formulas. However, as these constants and formulas are mainly human-specified, the generalizability of the solvers is limited. In this paper, we propose to explicitly retrieve the required knowledge from math problem datasets. In this way, we can determinedly characterize the required knowledge and improve the explainability of solvers. Our two algorithms take the problem text and the solution equations as input. Then, they try to deduce the required commonsense and domain knowledge by integrating information from both the problem text and equation. To show the effectiveness of our algorithms, we construct two math datasets and prove by experiments that our algorithms can retrieve the required knowledge for problem-solving.

Keywords: Math word problem solving, knowledge retrieval

1 Introduction

Math word problem (MWP) solving is a special subtask of question answering in which machine solvers need natural language understanding and numerical reasoning capability to solve a given problem. Traditionally, feature-based solvers (Kushman et al., 2014; Hosseini et al., 2014) learn to apply the corresponding operations with the help of salient features or indicators (e.g., "*buy A, B and C in total*" may indicates a series of addition).

Benefiting from the availability of large scale datasets, neural solvers have emerged. They utilize encoder-decoder architectures to encode the problem text into hidden representations and learn to decode them into equation strings or operation trees (Wang et al., 2017; Amini et al., 2019; Xie and Sun, 2019; Zhang et al., 2020). During the decoding stage, pre-specified constants and formulas are either added to the vocabulary or introduced by some special mechanisms so that the solver can generate equations that carry mathematical knowledge. In most cases, the constants or formulas are limited and human-specified, impeding the generalizability of the solvers to different types of problems (e.g., commonsense problems, geometry problems, etc).

In this paper, we propose to alleviate this issue by automatically retrieving the required knowledge from MWP datasets. To do so, our algorithms try to identify **numbers** and their **associated concepts** or **units** in the text (e.g., in "*the length is 5 m*", *length* is the concept and *m* is the unit) and then deduce the required knowledge by aggregating information from solution equations. For example, if there are two different units in the problem (e.g., "*the length is 5 m and width is 50 cm*"), then our algorithms will try to find the ratio that possibly **bridges** these two units. In this way, our algorithms may be able to retrieve the unit conversion knowledge that there is a conversion ratio *100* between "cm" and "m". Technically, this task differs from standard problem solving tasks in which we aim to characterize all the required knowledge in a dataset rather than predicting the required knowledge for a single problem.

To verify our algorithms, we construct two middle-sized MWP datasets and annotate each problem with the associated knowledge. Experimental results show the effectiveness of our algorithms that they can retrieve 69.8% and 62.5% of the required knowledge for these two datasets, respectively.

Type	Example	Example Problem
Object property (commonsense)	A chicken has two feet and a rabbit has four.	There are 15 chickens and 10 rabbits in the cage. How many animal feet are in there? ($animal\_feet = 15 \times 2 + 10 \times 4$)
Hyper/hyponym (commonsense)	A daisy or rose is a kind of flower; a flower pot is not a flower.	Mary bought 3 daisies, 2 roses, and 5 flower pots from a flower store. How many flowers does she have?
Unit conversion (commonsense)	One kilometer equals 1000 meters.	Sam just ran a race of 3400 meters long. How many kilometers was the race?
Geometry (domain)	Formulas like "area = length×breadth".	The length of a rectangular plot is thrice its breadth. If the area is 972 sq. m, then what is the perimeter of the plot?

Table 1: Commonly used commonsense and domain knowledge in MWP solving

2 External Knowledge in MWP Solving

As with other QA tasks, solving MWPs usually requires external knowledge that is beyond the given information in the problem. Table 1 lists the common types of commonsense and domain knowledge used in MWP solving with prototypical examples.

Commonsense Knowledge is the set of prior knowledge that solvers are presumed to hold when dealing with problems concerning some real world scenarios. For example, as shown in Table 1, object-property or hyper/hyponym knowledge is critically needed to perform arithmetic operations between different objects. To calculate the number of flowers, a solver needs to know "daisy and rose are hyponyms of flower". As another example, the knowledge "a chicken has two feet; a rabbit has four" is required to calculate the number of animal feet.

Domain Knowledge On the other hand, domain knowledge also plays an essential role in MWP solving. Ranging from geometry and probability to combinations and permutations, a solver needs to apply some particular domain knowledge to solve the MWPs. In most cases, the domain knowledge is in the form of formulas. For example, a solver applies "the area formula for rectangle" to solve the geometry problem in Table 1. As another example, a solver may apply the conditional probability formula to solve a probability problem. Therefore, in this work we target on the domain knowledge that can be represented as formulas.

3 Retrieving Commonsense Knowledge

Our first step is to retrieve from MWP datasets the commonsense knowledge for problem solving.

3.1 Commonsense Knowledge as Mapping Ratios

Typically, commonsense knowledge concerns the introduction of extra numerical information, most of which can be regarded as *specific ratios* between concepts. For example, in the first MWP in Table 1, a solver introduces the object-property knowledge to calculate the total number of animal feet, where "a chicken has two feet" and "a rabbit has four feet". In fact, "2" and "4" serve as the associated mapping ratios that convert the concepts of "chicken" and "rabbit" to "animal feet".

Likewise, the knowledge of hypernym and hyponym can be considered as a 1-to-1 ratio that maps a hyponym to its hypernym or vice versa. On the other hand, the unit-conversion knowledge, obviously, can be represented as a mapping ratio between two units.

3.2 Identifying Mapping Ratios in Equations

To identify the ratios, our algorithm first extracts numbers in the text, and then creates

mappings that map numbers to their corresponding concepts or units, as shown in the first step in Figure 1. For example, the noun "pennies" is captured as the unit for the number "9". Specifically, we use StanfordNLP toolkit (Manning et al., 2014) for dependency parsing in order to locate the numbers and their head nouns (concepts/units).

On the other hand, for variables, our algorithm uses five simple semantic patterns to capture the problem target as the corresponding concept or unit, as shown in Table 2. For example, we capture "cent" from "how many cents ..." as the unit for variable "x". In our pilot study, this heuristic handles about 90% of the cases.

Pattern	Rule
how many A (and B) ...	The goal object is A (and B)
(what is / find) the (length / distance) ...	We take the first length unit in the problem as the goal unit.
(what is the / find the / how much) time ...	We take the first time unit as the goal unit.
how much weight ...	We take the first mass unit as the goal unit.
how much ...	The default unit is dollar.

Table 2: Patterns for capturing the goal concept/unit of the variables

Next, the algorithm deduces the mapping ratios using these number-to-concept mappings. Here we use basic arithmetic principles for ratio deduction. For an equation to make sense, every term must correspond to the same concept/unit. As in the equation "$9 + 4 \times 5 + 10 \times 10 = x$" in Figure 1, "9", "4×5", "10×10", and "x" should share the same concept. Based on the fact that "9" and "4" corresponds to "penny" and "nickel", respectively, we can thus infer that "5" serves as the mapping ratio that maps "nickel" to "penny". Figure 1 illustrates the overall deduction flow for a single MWP.

Finally, the algorithm collects the ratio candidates for the whole dataset. It calculates the occurrence frequency for each candidate and then removes the ones whose counts are less than a pre-specified threshold λ, as a way to filter wrongly generated ratios (like the one in red in Figure 1).

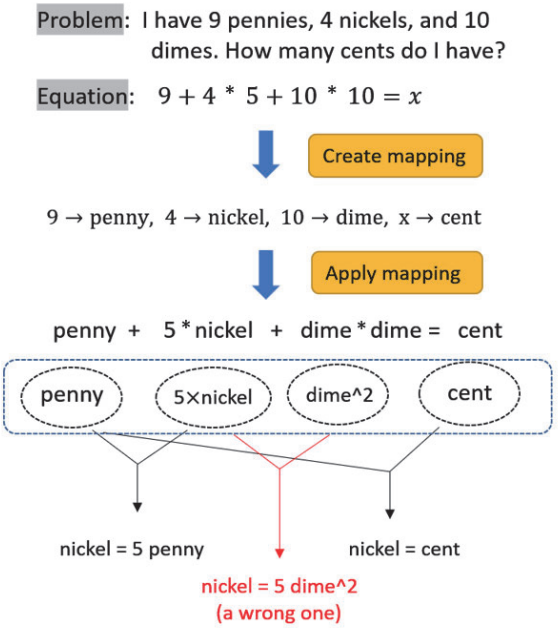

Figure 1: Flow of commonsense knowledge retrieval

4 Retrieving Domain Knowledge

Our next target is to retrieve the domain knowledge involved in MWP solving. We consider the types of domain knowledge used in the form of formulas, and assume they (at least the common ones) appear in more than one problem in a dataset so that our algorithm can discover them by finding common patterns.

4.1 Formulas and Concept Mappings

Generally, a solver uses a formula by substituting values (numbers) into it and then generating corresponding equations. As a result, the generated equations more or less keep the skeleton of the source formula, as shown in Figure 2. Thus, our goal is to retrieve the underlying formula from equations by considering the mapping between numbers and domain concepts.

To find the mapping, our algorithm performs entity recognition and relation prediction to identify domain concepts, numbers, and their mapping relationships, respectively.

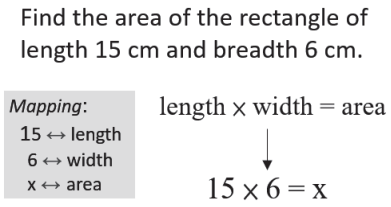

Figure 2: Simple example showing the idea that we have a mapping in mind when using formulas

As different domains may come with different domain language in their problem description, here we use neural models (which are more generalizable than semantic rules) for this task. Our pilot study showed that little labeled data is enough to train the models. Specifically, we employ two intuitive Bert-based models for entity recognition (Devlin et al., 2018) and relation prediction (Shi and Lin, 2019), and label a small amount of data to finetune both models.

Here we describe the entity and relation types as well as the model architectures we use in details. In this work, we consider geometry as the sample domain knowledge, and we identify four important entity types that are related to geometry domain: *object*, *attribute*, *value*, and *target*. Table 3 gives a detailed description for each entity. We use the architecture in Figure 3 to discern these entities in the problem text. Specifically, the model is based on Bert (Devlin et al., 2018) and fine-tuned on our MWP entity data using IO tagging.

Next, we seek to predict relationships between these entities. The relation types that we use are: *attribute-of*, *value-of*, and *none*, as described in Table 4. We adopt the framework of (Shi and Lin, 2019) for our model, as shown in Figure 4. In this framework, a special format is used for the input, in which entity mentions are replaced with entity-type masks in their original position and then moved to the end of the input. Such arrangement helps

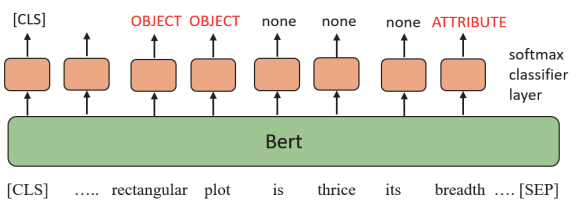

Figure 3: Architecture of entity recognition model

Entity types	Description
object	A geometric shape or real-world object, such as "circle" or "cylindrical container".
attribute	Attribute of the objects, such as "length", "width".
value	Number or value of a quantity, such as "two" triangles and "5" cm.
target	The goal of the problem, such as "volume" in "*what is the volume of X*".

Table 3: Entity types and their descriptions

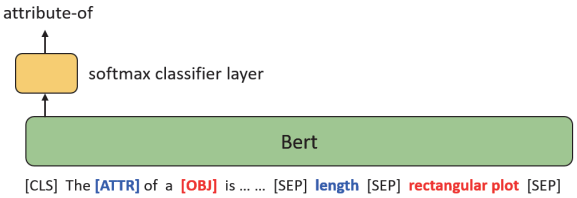

Figure 4: Architecture of relation extraction model. The original sentence "The length of a rectangular plot ..." becomes "The [ATTR] of a [OBJ] ... " and both entities are moved to the end of the input.

inform the model the two entities to focus on. Specifically, the model takes the problem text and two entities as input and is fine-tuned to predict their corresponding relation. Finally, we construct a concept mapping between an attribute and a number if there is a "value-of" relationship between them.

Types	Description
attribute-of	Relation between *object* and *attribute*, such as "circle" and "radius" in the description "the radius of the circle".
value-of	Relation between *value* and *attribute*, such as "radius" and "4" in "the radius is 4 cm".
none	None of the relations above.

Table 4: Relation types and their descriptions

4.2 Formula Candidate Generation

As shown in Section 3, our algorithm uses the mappings generated by entity recognition and

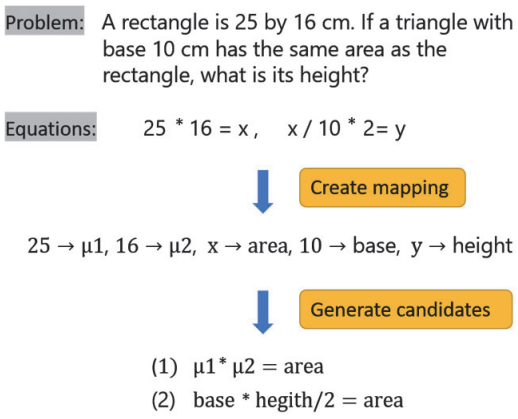

Figure 5: Flow of domain knowledge retrieval. We use μ_n to indicate that the concept of the number is unknown.

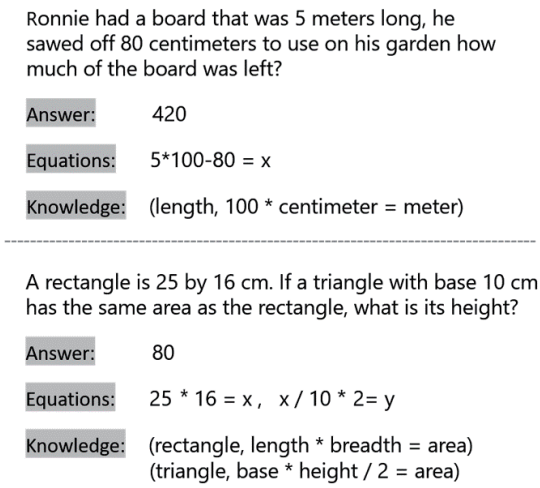

Figure 6: Sample MWPs from UnitQA (above) and GeometryQA (below). The annotation is in the form of "(*type*, *knowledge*)".

relation extraction models to generate formula candidates from the equations. Then, it splits long candidates into shorter ones by addition and subtraction operators, and normalize the resulted candidates in order to reduce the degree of freedom. After that, the algorithm gathers all formula candidates for each MWP in the dataset and calculates the occurrence frequency of these candidates. Finally, it removes the candidates whose counts are less than a pre-specified threshold λ.

5 UnitQA & GeometryQA

To check the effectiveness of our methodology, we construct two middle-sized MWP datasets: *UnitQA* and *GeometryQA*. The first dataset contains 1128 MWPs that require the commonsense unit-conversion knowledge, while the second dataset contains 675 MWPs that require geometric domain knowledge.

Problems of both datasets are collected from two large-scale datasets: *Dolphin18K* (Huang et al., 2016) and *MathQA* (Amini et al., 2019). We collect these MWPs using some domain-relevant keywords. Then, we manually annotate the required external knowledge (if any) for each problem, as shown in Figure 6. We select only a subset of problems from the large datasets because we would like to focus on basic problems first. The more advanced ones are left for future work.

Table 5 shows the statistics of UnitQA. It contains a total of 1128 MWPs, 305 out of which require unit-conversion knowledge across 43 different types, including the conversion knowledge between units of money, time, length and so on. To test the capability for retrieving other types of commonsense knowledge, we also heuristically select 25 problems that require **object-property** or **hypernym/hyponym** knowledge. Due to data sparsity, these problems are not large enough to form a dataset, yet they should help demonstrate the effectiveness of our algorithm (details described in Section 6.2).

Table 6 shows the statistics of GeometryQA. It contains 675 MWPs, 570 out of which require 40 different formulas for 18 different geometric objects, including circle, rectangle, and so on. In addition, we annotate an extra 193/46 geometric MWPs with corresponding entities/relationships to train the two different BERT models. We found that a small amount of annotated MWPs are enough to make accurate entity and relation predictions.

UnitQA	
Total problems	1128
Knowledge required	305
Total knowledge types	43
Types of unit conversion: money(34.9%), length(32.6%), time(14%), mass(11.6%), volume(4.7%), and area(2.3%).	

Table 5: Dataset statistics of UnitQA

GeometryQA	
Total problems	675
Knowledge required	570 (84% of 675)
Total formula types	40
There are 40 types of geometric formulas for 18 objects, including the area, perimeter, and volume formulas for square, circle, cubic, sphere, and so on.	

Table 6: Dataset statistics of GeometryQA

6 Experimental Results

6.1 Experimental Settings

We use exact the same setting for both models and implement them based on HuggingFace[1]. The dropout rate is set to 0.1. The parameters are optimized by Adam (Kingma and Ba, 2014), with learning rate 1e-4, batch size 50, and a max sequence length for the input 68.

6.2 Retrieving Commonsense Knowledge

We first conduct experiment on *UnitQA* to retrieve the commonsense unit-conversion knowledge. We test with threshold $\lambda = 0, 2, 5$ (for larger dataset, the λ should be adjusted accordingly.) Table 7 shows the overall performance of the experiment. When $\lambda = 0$, where no candidate is eliminated for insufficient frequency, it shows an upper bound of the recall (79%). As expected, when the threshold increases, it causes a decrease in recall and an increase in precision. We found that for $\lambda = 2$, about 67% of the unretrieved cases are due to concept identification errors, where the concept for numbers are wrongly identified. This suggests the limit of our rule-based deduction strategy.

Due to data sparsity, we have not collected enough problems that require object-property or hyper/hyponym knowledge. Yet, in our small-scale experiment (about 25 problems), our algorithm can identify about 68% of the required knowledge and retrieve something like "chicken ↔ 2 feet" , "rabbit ↔ 4 feet" , and " bicycle ↔ 2 wheels" for the first and second types of knowledge in Table 1.

[1]https://github.com/huggingface/transformers

	Precision	Recall
$\lambda = 0$	47.8% (33/69)	79.1% (34/43)
$\lambda = 2$	**90.9%** (30/33)	**69.8%** (30/43)
$\lambda = 5$	100% (9/9)	20.9% (9/43)

Table 7: Precision and recall for retrieving the unit-conversion knowledge in UnitQA with $\lambda = 0, 2, 5$

6.3 Retrieving Domain Knowledge

In this experiment, we test the effectiveness of our algorithm for retrieving domain knowledge on *GeaometryQA*. We test with $\lambda = 0, 2, 5$. Table 8 presents the overall knowledge retrieval result. When $\lambda = 0$, where no candidate is eliminated for insufficient frequency, it shows an upper bound of recall (70%). As expected, when the threshold increases, it causes a decrease in recall and an increase in precision. We conduct error analysis on $\lambda = 2$ and find that about 75% unretrieved formulas are also caused by concept identification errors. That is, if an equation contains several variables, our algorithm cannot always find the corresponding concept for each variable and thus unable to retrieve the correct formula.

	Precision	Recall
$\lambda = 0$	71.8% (28/39)	70% (28/40)
$\lambda = 2$	**92.6%** (25/27)	**62.5%** (25/40)
$\lambda = 5$	80% (8/10)	20% (8/40)

Table 8: Precision and recall for knowledge retrieval on GeometryQA with $\lambda = 0, 2, 5$

7 Conclusion

In this paper, we introduced a task of retrieving the required knowledge for math word problem datasets. By explicitly identifying the required knowledge, we can characterize the datasets and assist current neural solvers. We then proposed two algorithms which retrieves the commonsense and domain knowledge, respectively, and constructed two datasets with each MWP annotated with the required knowledge. Experimental results demonstrated the effectiveness of our algorithms.

References

Aida Amini, Saadia Gabriel, Peter Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. 2019. Mathqa: Towards interpretable math word problem solving with operation-based formalisms. *arXiv preprint arXiv:1905.13319*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. 2014. Learning to solve arithmetic word problems with verb categorization. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 523–533.

Danqing Huang, Shuming Shi, Chin-Yew Lin, Jian Yin, and Wei-Ying Ma. 2016. How well do computers solve math word problems? large-scale dataset construction and evaluation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 887–896.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Nate Kushman, Yoav Artzi, Luke Zettlemoyer, and Regina Barzilay. 2014. Learning to automatically solve algebra word problems. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 271–281.

Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, pages 55–60.

Peng Shi and Jimmy Lin. 2019. Simple bert models for relation extraction and semantic role labeling. *arXiv preprint arXiv:1904.05255*.

Yan Wang, Xiaojiang Liu, and Shuming Shi. 2017. Deep neural solver for math word problems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 845–854.

Zhipeng Xie and Shichao Sun. 2019. A goal-driven tree-structured neural model for math word problems. In *IJCAI*, pages 5299–5305.

Jipeng Zhang, Lei Wang, Roy Ka-Wei Lee, Yi Bin, Yan Wang, Jie Shao, and Ee-Peng Lim. 2020. Graph-to-tree learning for solving math word problems. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3928–3937.