Automatic Arithmetic Word Problem Solving

Yucen Li
Carnegie Mellon University
Pittsburgh, PA 15213
yucenl@andrew.cmu.edu

Xiaorong (Sean) Zhang
Carnegie Mellon University
Pittsburgh, PA 15213
xiaoronz@andrew.cmu.edu

Abstract

Solving arithmetic word problems may be easy for humans, but it can be challenging for computers. Two of the current approaches are verb categorization and template matching, and each one has its advantages on certain subsets of all word problems. We identified two of these subsets where the differences in performance are significant: the problems with only addition and subtraction, and the problems with irrelevant information. We also designed methods to identify these subsets, using Naive Bayes and Natural Language Processing. Equipped with the subset information, we are able to choose the most relevant algorithm. This approach leads to an increase in overall accuracy.

1 Introduction

Computers are able to solve complex quantitative problems, and automatic computation has great potential to influence our everyday life. However, solving the same problems embedded in text often requires a deep understanding of the real world, as accurately translating words to equations can be difficult. Advancing the ability to solve such word problems can benefit natural language processing: arithmetic word problems require an understanding of relationships between words and entities, and solving these problems allows machines to better understand semantics of sentences and make mathematical inferences.

Arithmetic problems are structured up by first describing a specific state of the world. These states then undergo transitions in quantity, as described by the rest of the problem. The problem finally ends with a quantitative question regarding the states or state transitions of the world. This setup is illustrated in the the following example.

Example 1
There were 28 bales of hay in the barn. Tim stacked bales in the barn today. There are now 54 bales of hay in the barn. How many bales did he store in the barn?

The state of the world starts with the 28 bales of hay in the barn. Next, an unknown state transition occurs, and the state of the worlds ends with 54 bales of hay in the barn. Finally, the problem includes an inquiry about the middle state transition between the beginning and final states.

While any computational system can easily solve the corresponding mathematical equations, these arithmetic word problems are challenging because they require semantic understanding of the described relationships, many of which may be implicit. In our model, we approach this problem by combining the transformation of world states with equation templates. This approach combines the domain robustness of verb categorization with the higher computational ability of template equations.

2 Related Work

A few different methods have been used to solve mathematical word problems.

2.1 Verb Categorization

Hosseini et al. [2] developed a system ARIS which solves word problems by mapping the sentences in the problem to a representation of the states of the world. The transitions between these states are then determined by the categorization of the verbs in the problem. Through these state transitions, ARIS turns this information into a representative equation, which is solved using an equation solver.

In ARIS, the problem text is first split into fragments, with each fragment representing a world state. Each world state is represented using entities, containers, and quantities, as seen in Figure 1. An entity (E) is the object whose quantity is observed or changing. Entities can also have attributes (A), which are properties of the object. The container (C) represents the set of entities as specified by the problem, and often corresponds to locations containing entities or a person with entities. Lastly, the quantities in the problem correspond to the known numbers, any unknown variables, or other numerical expressions which may exist. The transitions between these world states are called state transitions. Figure 1 shows examples of the states and state transitions for two different word problems.

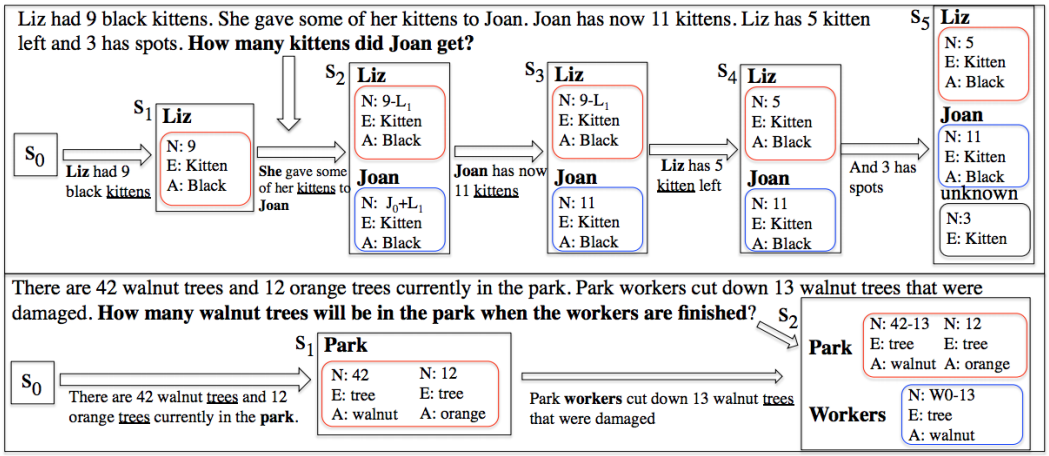

Figure 1: Sequence of states from Hosseini et al. [2]

The paper classifies verbs into different categories depending on how they typically affect entities and containers. For example, in "Liz gave some of her kittens to Joan" from Figure 1, the verb "give" is used to decrease the number of entities in one container (Liz) and increase the number of entities in the second container (Joan). The model is then trained to identify categories for the verbs in the sentences. Each verb is categorized using similarity-based features, WordNet-based features, as well as structural features. ARIS then uses this information to train an SVM. This SVM is used to assign the verb categories for the test data. Different verb categories will affect the quantities of the world states different differently (by adding, transferring, etc).

ARIS has a 76.5% accuracy for addition and subtraction problems, and the categorization of verbs allows the model to be robust across training and testing datasets from different domains. However, the classification of the verbs as either increasing or decreasing prevents the model from working well with more complex problems involving different operations such as multiplication and division, which may involve different operations such as multiplication and subtraction. This is a significant drawback, as ARIS is optimized for a very specific type of problem within the entire domain of arithmetic word problems.

2.2 Template matching

Kushman et al. [1] implemented a different system for this problem based on equation templates. The decision process has two steps. First, we decide an equation template for the question, which looks like

$$\begin{cases} x^{(1)} + y - n_1 = 0 \\ x^{(2)} - n_2 z + n_3 = 0 \end{cases}$$

Here n_1 to n_3 will be filled in with numbers from the question, x to z with word variables, and $x^{(1)}$, $x^{(2)}$ indicates that the same variable occurs twice. For the second step, starting from the template, we match the words and numbers in the question to variables in the equations. We then solve the equation to obtain the final answer.

For the training step, we look at each question string and each template, and we compute a feature vector based on these two parameters. It includes 20 types of entries such as the bigram of the sentence, whether a slot is an object of a question, whether two slots are in the same sentence, etc. After the feature is computed, we simply find a parameter than maximizes the log-likelihood under the feature, and it is found using L-BFGS.

The authors considered different supervision scenarios, including both supervised learning and semi-supervised learning. They obtain an accuracy of 46% in the semi-supervised case, and 69% in the supervised case.

For convenience, from now on we will refer to the verb categorization algorithm as ARIS, and the template matching algorithm as KAZB.

3 Methods

We improve on the current results by:

1. Determining the problem type using Naive Bayes,
2. Determining the information relevance using NLP.

3.1 Determining the problem type

KAZB is designed to solve general word problems: whatever operation the problem contains, as long as it can be expressed in one of the equation templates, it can potentially be solved. On the other hand, ARIS is designed for addition and subtraction only: the verbs are only given in these two categories. Therefore it is not surprising to see that ARIS can handle problems with only addition/subtraction with a higher accuracy, while KAZB is good at problems with also multiplication/division.

Due to the fact that a difference in problem type has a significant impact on overall accuracy, it makes sense to consider classifying the problems first. Our goal is that, whenever we are given a new problem, we first find out if it contains addition/subtraction only, or if it also contains multiplication/division.

A second observation is that, in order to determine whether a question is addition-type or multiplication-type, it is very useful to look at the verbs and nouns in the sentences. For example, the verb "divide" highly suggests that the problem is multiplication-type, and the noun "ratio" suggests the same idea. Of course, a lot of verbs and nouns can potentially appear in both types of problems, but if we take the average over all the verbs and over all the nouns, we should have a good estimate of the actual category.

These ideas motivate us to implement the Naive Bayes algorithm for this step. It does the following: during training, we count the occurrence of each verb and noun, and we compute the probability of that word given addition-type or multiplication-type. So we obtain, for each word w,

$$P(w \mid \text{add}), \quad P(w \mid \text{mult}).$$

We also compute the priors $P(\text{add})$ and $P(\text{mult})$. Now assume that we are given a new sentence with verbs $v_1, \ldots, v_k$. Since the number of verbs can be different across sentence, we first compute the average probabilities by

$$P(v \mid \text{add}) = \frac{1}{k} \left(\sum_{i=1}^{k} P(v_i \mid \text{add}) \right),$$

and similarly we calculate

$$P(v \mid \text{mult}), \quad P(n \mid \text{add}), \quad P(n \mid \text{mult}).$$

By Naive Bayes' assumption, we can multiply the probabilities to obtain

$$P(n, v \mid \text{add}), \quad P(n, v \mid \text{mult}).$$

Using Bayes' rule, we have

$$P(\text{add} \mid n, v) \propto P(\text{add})P(n, v \mid \text{add}), \quad P(\text{mult} \mid n, v) \propto P(\text{mult})P(n, v \mid \text{mult}),$$

and these probabilities allow us to classify the problem into either addition-type or multiplication-type.

For a concrete example, consider the following word problem.

> 24 is divided into two parts such that 7 times the first part added to 5 times the second part makes 146. Find each part.

The verbs are $v_1 =$ "divide," $v_2 =$ "add," and $v_3 =$ "make," and the nouns are $n_1 =$ "part" and $n_2 =$ "time." For each verb, we calculate from the training data

$$P(v_1 \mid \text{add}) = 0, \quad P(v_1 \mid \text{mult}) = 0.00727, \text{ etc.}$$

$$P(n_1 \mid \text{add}) = 0.00506, \quad P(n_1 \mid \text{mult}) = 0.0324, \text{ etc.}$$

After we do this for every verb and noun, we compute the average probability by

$$P(v \mid \text{add}) = 0.0219, \quad P(v \mid \text{mult}) = 0.0394,$$

$$P(n \mid \text{add}) = 0.00759, \quad P(n \mid \text{mult}) = 0.144.$$

So the probabilities for v, n are

$$P(n, v \mid \text{add}) = 1.67 \times 10^{-4}, \quad P(n, v \mid \text{mult}) = 5.67 \times 10^{-3}.$$

In addition, we have the prior probabilities

$$P(\text{add}) = 0.207, \quad P(\text{mult}) = 0.793.$$

Therefore using Bayes rule, we have

$$P(\text{add} \mid n, v) = 3.45 \times 10^{-5}, \quad P(\text{mult} \mid n, v) = 4.50 \times 10^{-3},$$

and we conclude that we should classify the question as multiplication-type. This classification is correct, since if we look at the original problem, we notice that we have to apply division in order to solve it.

One thing to note about our algorithm is that, since the probabilities never get smaller than 10^{-7}, we did not use log likelihood in our computation.

3.2 Determining information relevance

While KAZB performed well on many addition and subtraction problems, it often matched all of the numbers in the problem to a slot in the template. However, there are many instances within word problems where not every single number needs to be used, as can be seen in the following problem.

Example 2
Jason found 49 seashells and 48 starfish on the beach . He gave 13 of the seashells to Tim. How many seashells does Jason now have?

In this question, starfish is mentioned in the beginning of the problem, but the quantity is never further modified or queried in the remainder of the problem. However, the KAZB method often tried to use all of the numbers and would lead to incorrect results.

In contrast, while ARIS does not work as well as KAZB on typical addition and subtraction problems where every number is mapped to part of the template, it has higher accuracy than KAZB on word problems containing irrelevant information. Because the model is designed to generate equations

using verb categorization between relevant states, the model does not incentivize including all the numbers from the word problem. Therefore, if an addition or subtraction question does contain irrelevant information, it seems logical to use the ARIS model rather than the KAZB.

To determine if a question contains unnecessary information, each word problem was parsed into states which were representative of the problem. For each problem, we first used coreference resolution to find all references to an entity throughout the problem. Next, for each sentence, we found the dependency tree as well as the word categories using the Stanford Core NLP parser. Using this grammar structure, we could then better identify the entities, quantities, and attributes associated with each aspect of the word problem. We used the original verb categorization from ARIS in order to identify the changing quantity of entities throughout the problem. An example can be seen in Figure 2

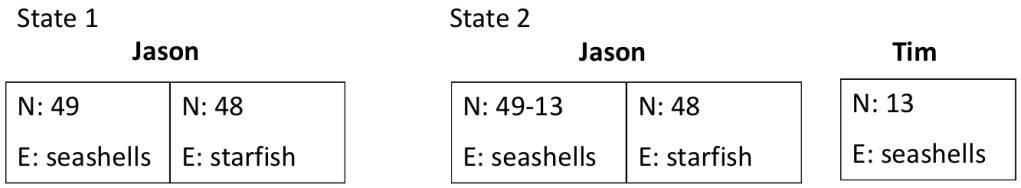

Figure 2: States of Example 2

Once the states of the problem were identified, the problem of recognizing irrelevant information was greatly simplified. We simply searched traversed through the states of the problem: if a state is initialized but never modified or queried throughout the problem, then the problem is likely to have contained an irrelevant number.

4 Dataset

We have used the following three datasets.

1. The first dataset is the one used in [2], and it is available on the author's website at `https://www.cs.washington.edu/nlp/arithmetic`. It contains a json file of 395 word problems, along with their corresponding equations and solutions. Each problem involves only addition and subtraction. This is the most basic dataset out of the three.

2. The second dataset is the one in [1], and it is available on the author's website at `http://groups.csail.mit.edu/rbg/code/wordprobs/`. It contains a json file of 514 algebra problems, as well as the equations (with one or more variables) and solutions. The problems involve all four arithmetic operations.

3. The last dataset is defined in [5], and it is similar to the second dataset. It is available at `https://www.microsoft.com/en-us/download/details.aspx?id=52628`. It contains 1000 algebra problems with their equation templates, equations, alignments of variables, and solutions. They involve all four arithmetic operations.

For convenience, we call the first dataset "Add," the combination of the second and the third datasets "Multiply," and the combination of all three datasets "All."

5 Experiments and Results

We tested the Naive Bayes algorithm on the dataset All. We randomly chose 90% of all the questions to be training data, and the remaining 10% to be test data. The results are as follows.

Table 1: Naive Bayes Results

Dataset	Accuracy
Add	69.6%
Mult	99.8%
All	93.6%

By "Add" and "Mult," we refer to the subset of test data that are in Add or Mult. As we can see, the algorithm performs really well for multiplication problems, and the overall accuracy is also high.

The method for determining irrelevant information was able to correctly identify 65% of the problems in the addition and subtraction dataset which contained irrelevant data, while only misclassifying 1% of problems from the dataset which did not contain irrelevant data.

For the overall results, we have the following table.

Table 2: Overall Results

Datasets	ARIS	KAZB	Our algorithm
Add	77.7	64.0	**79.1**
Mult	0.002	**68.7**	**68.7**
All	16.1	66.2	**70.3**

We see that for addition problems, our algorithm performs even better than ARIS, since we categorized based on irrelevant information. For multiplication, our performance is almost the same as KAZB, which is what we expected, since only KAZB can handle multiplication, so we cannot possibly do better. But most importantly, for overall accuracy, our algorithm has outperformed both ARIS and KAZB.

6 Conclusions and Future Work

After careful analyses on the two existing approach, we conclude that ARIS outperforms KAZB when the problem has irrelevant information, while KAZB does much better on problems with multiplication. Given a new problem, we first classify it based on the operations and relevancy, and then we apply the most appropriate algorithm. In this way we achieve a better overall accuracy.

For further work, we can consider other algorithms for this problem, for example, the method of using expression trees. We can analyze when this approach outperforms the others, and we change our decision process accordingly. Additionally, we can continue identifying cases where one model outperforms another and add additional parameters to determine the classification, such as the complexity of the question. It may also be beneficial to look into different methods of classifying problems, and to use SVM or a neural net instead of our current Naive Bayes classification.

However, the main result of our algorithm is that certain models appear to be better suited for certain types of questions. Because the accuracy for our algorithm was higher than that of either one of the original models, this indicates that the diverse design of models may also lead to diverse performance on questions. For example, KAZB should not be used in cases involving irrelevant numbers due to its number matching behavior. Therefore, in future research, we can consider developing models which heavily focus on a narrow subset of arithmetic word problems, rather than working on a generalized model which may not be as strong.

References

[1] Kushman, Nate, et al. (2014) Learning to automatically solve algebra word problems. *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers).*

[2] Hosseini, Mohammad Javad, et al. (2014) Learning to solve arithmetic word problems with verb categorization. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP).*

[3] Roy, Subhro, and Dan Roth. (2016) Solving general arithmetic word problems. *arXiv preprint, arXiv:1608.01413*.

[4] Ling, Wang, et al. (2017) Program induction by rationale generation: Learning to solve and explain algebraic word problems. *arXiv preprint, arXiv:1705.04146.*

[5] Shyam Upadhyay and Ming-Wei Chang. (2015) Draw: A challenging and diverse algebra word problem set. *Microsoft Research, MSR-TR-2015-78*