Operator Prediction through Sentence Simplification for Solving Arithmetic Word Problems

First Author
Affiliation / Address line 1
email@domain

Second Author
Affiliation / Address line 1
email@domain

Abstract

This paper presents a sentence simplification approach in learning to solve arithmetic word problems. The approach performs a thorough analysis to map each sentence in the word problem to a simplified sentence . The objective of our approach is to use the basic properties of the language to simplify sentences and then classify the simplified sentences to their operators. We simplify the sentence until it represents a single operation. Using the predicted operators for each simplified sentence we build an equation and solve the problem. We train our classifier on MAWPS (A Math Word Problem Repository) (Koncel-Kedziorski et al., 2016) dataset and achieve an accuracy of 82.57%. Experimental results show that our method outperforms existing systems, achieving state of the art performance on benchmark datasets of arithmetic word problems.

1 Introduction

Interpreting a sentence representing a single mathematical operation is relatively easier than interpreting a sentence having multiple mathematical operations. For example it will be difficult to extract quantities from the sentences in the actual question. However, it becomes much simpler to extract information useful for the equation from the simplified sentences as shown in Figure 1 in order to solve the problem.

To simplify the word problem, we execute a set of rules on each sentence so that it possibly has multiple simplified sentences. Our goal here is to have a single operation in each simplified sentence. To solve the problem , we extract an equation using the coherent set of simplified sentences and their predicted operators.

<table>
<tr><th colspan="2">Example Word Problem</th></tr>
<tr><td colspan="2">A spaceship traveled 0.5 light-year from Earth to Planet X and 0.1 light-year from Planet X to Planet Y. How many light-years did the spaceship travel in all ?</td></tr>
<tr><th>Simplified Sentence</th><th>Predicted Operations</th></tr>
<tr><td>A spaceship traveled 0.5 light-year from Earth to Planet X.</td><td>+ 0.5 light-year</td></tr>
<tr><td>A spaceship traveled 0.1 light-year from Planet X to Planet Y.</td><td>+ 0.1 light-year</td></tr>
<tr><td colspan="2">Equation: + 0.5 light-year + 0.1 light-year</td></tr>
</table>

Figure 1: Equation Extraction from Simplified Sentences

2 Related Work

There have been a number of attempts to solve arithmetic word problems through machine learning (ML). All of the approaches that are not template based (e.g., (Hosseini et al., 2014), (Roy et al., 2015) and (Roy and Roth, 2015)) use different methods to extract similar information. Based on different ways the information is represented, an equation is generated for the problem text. The template based method of (Kushman et al., 2014) implicitly assumes that the solution will be generated from a set of predefined equation templates. Some of these methods only solve addition and subtraction problems (e.g., (Hosseini et al., 2014) and (Roy et al., 2015)) while others (e.g., (Roy and Roth, 2015) and (Kushman et al., 2014)) can also solve problems that require multiplication/division operations. Our approach uses the idea of sentence simplification to predict operators and handles addition and subtraction problems.

3 Our Method

In this section we describe how our system maps an arithmetic word problem to an equation. It consists of three main steps:

1. Extract simplified sentences from complex word problems using the simplification rules.

2. Train a model to classify operators for each simplified sentence.

3. Solve the problem by updating the world states with the learned verb categories and forming equations.

3.1 Sentence Simplification and Problem Decomposition

Sentences in an arithmetic word problem are sometimes complex. Hence, it is difficult to extract information from such sentences. Even more challenging is to predict the impact of the sentence on the result. We extract a total of 1218 addition subtraction problems from the MAWPS repository (Koncel-Kedziorski et al., 2016) and execute sentence simplification on all of them. We also release a dataset of simplified sentences for these word problems.[1] We create a mapping for each sentence in the problem text to its simplified sentences by extracting their relational dependencies from the Stanford dependency parser. Currently, our system simplifies sentences based on conjunctions and punctuation characters such as comma. There are certain rules when simplifying the sentence as described in Section 3.1.1.

Notation: Given a problem text $\mathbf{S}$, let the sentences in the $\mathbf{S}$ be $< \mathbf{s_1}, ..., \mathbf{s_n} >$. Each sentence $\mathbf{s_i}$ will be simplified to m simplified sentences. Let the simplified sentences of $\mathbf{s_i}$ be $< k_1, ..., k_m >$.

3.1.1 Rules for Simplifying Sentences

When the conjunction *"and"* or the punctuation character *","* is encountered, our system attempts to create two simplified sentences from the actual sentence. The first sentence is the part before these elements while the second sentence is the part after them. Notably, after the split there may be some words which would be required in the second sentence. Consider the sentence in Figure 2:

[1]URL not provided to maintain anonymity.

$\mathbf{s}$: The school cafeteria ordered 42 red apples and 7 green apples for students lunches.

Figure 2: Example Sentence

In s, the split based on *"and"* will result in two sentences as shown below:

k_1:*The school cafeteria ordered 42 red apples*
k_2:*7 green apples for students lunches*

Here k_1 has a subject and a verb while k_2 does not have them, making it an improper sentence. Hence, there are some rules for adding words to simplified sentences:

3.1.2 Rules for adding words to simplified sentences.

1. If k_1 starts with an existential and has a verb after it and if k_2 does not have either expletive or verb, distribute them to k_2. Consider the example in Figure 3:

s: There were 2 siamese cats and 4 house cats.
k_1: There were 2 siamese cats.
k_2: There were 2 house cats.

Figure 3: Example sentence for Rule 1

.

The expletive and verb were added to k_2 based on the simplification rule mentioned above.

2. If k_1 starts with a noun, and if k_2 starts with a verb, the noun from the former will be distributed to the latter. Refer to an example in Figure 4.

s: Joan ate 2 oranges and threw 3 apples.
k_1: Joan ate 2 oranges.
k_2: Joan threw 3 apples.

Figure 4: Example sentence for Rule 2.

3. If k_1 starts with a noun and k_2 has a *noun verb* pattern, do nothing.

   In the example presented in Figure 5, No words from k_1 were added to k_2 since it had the *noun verb (Sara has)* pattern.

s: Tom has 9 yellow balloons and Sara has 8 yellow balloons.
k_1: Tom has 9 yellow balloons.
k_2: Sara has 8 yellow balloons.

Figure 5: Example sentence for Rule 3.

4. If k_2 contains a preposition at the end and k_1 does not, it will be distributed from k_2 to k_1. Consider the example presented in Figure 6:

s: Joan found 6 seashells and Jessica found 8 seashells on the beach.
k_1: Joan found 6 seashells on the beach.
k_2: Jessica found 8 seashells on the beach.

Figure 6: Example sentence for Rule 4.

After splitting the sentence based on *and*, the preposition and the words after it *on the beach* were added to the first sentence.

5. Based on the output by the dependency parser and our rules, there might be some words which might not have been identified. But we still need those words in the simplified sentences. Therefore, the sentence simplification system identifies all such words. If these words appear before the conjunction, they are added to k_1 at the correct index and if they appear after the conjunction, they are added to k_2.

4 Operation Prediction Classifier

After all the sentences are simplified, we randomly divide the dataset into training and testing in the ratio of 3:1. We train our model using Random Forest classifier that predicts one of the following classes for each simplified sentence in the word problem:

Class Label	Description
+	Addition Operation.
-	Subtraction Operation
?	Fragment asking some question
=	Assignment Operation
i	Irrelevant information for solving the word problem

Figure 7: Class Labels for Operator Prediction Classifier

4.1 Features

4.1.1 Position based

The index of simplified sentence in the question is important to determine the operation that sentence will perform. We take 2 such features into consideration as shown in Figure 8

Feature	Description
IsItAFirstSentence	Most word problems in the training data had a positive operation in the first sentence.
IsItALastSentence	Almost always the last sentence in the word problem is a question sentence.

Figure 8: Position based Features

4.1.2 Relation based

Existence of some important dependency relations is used as a feature. Refer to Figure 9 for the list of relation based features:

Feature	Description
nsubj	The sentence is more likely to perform an operation in the presence of these two relations.
dobj	

Figure 9: Relation based Features

4.1.3 Parts of Speech based

Existence of some Parts of Speech of the words in the sentence is used as a feature. Refer to Figure 10 for the list of POS based features:

Feature	Description
CD: Cardinal	The sentence is more likely to perform an operation in the presence of these Parts of Speech.
WRB: WH-Adverb	
EX: Expletive	
RBR: Comparative Adverb	
RBS: Superlative Adverb	
VBD: Past tense Verb	
VB: Base form Verb	

Figure 10: Position based Features

4.1.4 Verb Similarity based

A *Positive Verb* is a verb which indicates that the subject in the sentence is gaining some quantified object. A *Negative Verb* is a verb which indicates that the subject is losing something. We extract the most frequent verbs in $+$ and $-$ labeled sentences. Based on the frequencies we extract 13 significantly differentiating verbs for each class. The similarity of the lemma of these verbs to the action verb in the sentence is then used as a feature. Therefore, we have a total of 26 such features. The similarity score is calculated based on the WUP word similarity using WordNet (Miller, 1995).

5 Word Problem Solver

5.1 Using Operator Prediction Results

Based on the predicted operators for each simplified sentence, we create a representation for every subject having one or more objects. Refer to Figure 12 for details:

w: Joan found 70 seashells on the beach . she gave Sam some of her seashells . She has 27 seashell . How many seashells did she give to Sam ?
k_1: Joan found 70 seashells on the beach.
k_2: she gave Sam some of her seashells .
k_3: She has 27 seashell .
k_4: How many seashells did she give to Sam ?

Figure 11: Example Word Problem

The representation of the above simplified sentences would be as shown in Figure 13:

Sentence	Predicted Operator	Representation
k_1	$+$	$Joan > 70 seashell$
k_2	$-$	$Joan > 70 - X seashell$ $Sam > +X seashell$
k_3	$=$	$Joan > 70 - X = 27$ *seashell*

Figure 12: Example Word Problem

We use Spacy[2] to extract dependency relations and attempt to extract equation for a word problem based on the subject and object identified in the question sentence. There are 4 scenarios we consider:

1. If the question sentence has a singular subject and an object, we map the subject to one of the entities in our representation and output the result.

2. If the question sentence has a plural subject (For Example: *they*) and an object, we perform all the identified operations for that object.

3. If the question sentence has a comparative adjective and multiple subjects or multiple objects, we output the result by subtracting the smaller quantity from the greater one.

4. If the question sentence does not fall in any one of the above cases, we perform all the identified operations in our representation and output the result.

6 Experimental Results

6.1 Operator Prediction Classifier

Out of 1218 simplified word problems, we use 1015 to train our classifier and the remaining 203 to test.

Class	Training Count	Testing Count	Precision	Recall
$+$	1811	375	96.23	74.93
$-$	528	102	85.57	87.25
?	1015	203	100	100
$=$	113	28	25	53.57
i	317	44	35.48	75
Accuracy: 82.57%				

Figure 13: Operator Prediction Results

6.2 Word Problem Solver

	MA1	IXL	MA2	Total
Hosseini et al. (2014)	83.6	75.0	74.4	77.7
Roy and Roth (2015)	-	-	-	78.0
Kushman et al. (2014)	89.6	51.1	51.2	64.0
	93.0	52.0	**81.0**	75.33

Figure 14: Solver Results for AI2 Dataset

[2] https://spacy.io

	Training	**Testing**
Count	1015	203
Accuracy	90.14	**91.62**

Figure 15: Solver Results for MAWPS Dataset

7 Conclusion

This paper presents a method for understanding and solving addition and subtraction arithmetic word problems. We develop a novel theoritical framework, centered around the notion of sentence simplification for operator predictions. We show this by developing a classifier that yields strong performance on several benchmark collections. Our approach also performs equally well on multistep problems, even when it has never observed a particular problem type before.

References

Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. 2014. Learning to solve arithmetic word problems with verb categorization. In Alessandro Moschitti, Bo Pang, and Walter Daelemans, editors, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 523–533. ACL.

Rik Koncel-Kedziorski, Subhro Roy, Aida Amini, Nate Kushman, and Hannaneh Hajishirzi. 2016. Mawps: A math word problem repository. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1152–1157, San Diego, California, June. Association for Computational Linguistics.

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, Baltimore, Maryland, June. Association for Computational Linguistics.

George A. Miller. 1995. Wordnet: A lexical database for english. *Commun. ACM*, 38(11):39–41, November.

Subhro Roy and Dan Roth. 2015. Solving general arithmetic word problems. In Llus Mrquez, Chris Callison-Burch, Jian Su, Daniele Pighin, and Yuval Marton, editors, *EMNLP*, pages 1743–1752. The Association for Computational Linguistics.

Subhro Roy, Tim Vieira, and Dan Roth. 2015. Reasoning about quantities in natural language. *Transactions of the Association for Computational Linguistics*, 3:1–13.