Learning to Solve Arithmetic Word Problems Using Sentence Simplification

Vishal Rajpal
Northeastern University
rajpal.vi@husky.neu.edu

Abstract

In order to respond to an arithmetic word problem correctly one needs to understand the question to an extent which allows determining the constraints. These are mostly semantic and are imposed by the question on its answer. Constraints from individual sentences suggest a mathematical operation and when the operators from these sentences are used collectively the answer can be derived. To extract the constraints efficiently, a concept of Syntactic Pattern is introduced which is generated by parsing the sentence using a dependency parser. It encapsulates all the relevant information in a sentence including the subject, verb and object with its quantity. Another important method for this thesis which relies on syntactic patterns is Sentence Simplification. The idea is to have a subject, verb, an object and other necessary parts of speech in the sentence so that it suggests a single operation. Based on the identified patterns a sentence may be simplified to multiple sentences. This would make the classification process easier since the sentences are less complex. To this end, it would be the classifier's job to classify the sentence to a mathematical operator. The identified operators for individual sentences are used to build a mathematical equation for the entire word problem. The results based on 3 datasets are reported and seem promising as compared to the existing systems.

1 Introduction

Answering arithmetic word problems has gained a lot of interest in recent years. The problem is attractive to NLP since the text is concise and relatively straightforward with identifiable semantic constraints. As these problems are directed towards elementary school students, they begin by describing a partial world state, followed by simple quantified updates or elaborations and end with a quantitative question. This information can be mapped to basic operations(addition, subtraction, multiplication and division) and an equation can be created corresponding to the problem text.

There have been a number of attempts to solve arithmetic word problems through Machine Learning. All the approaches which are not template based (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 built 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 (Hosseini et al., 2014), (Roy et al., 2015) while (Roy and Roth, 2015) and (Kushman et al., 2014) can solve problems for all operations.

The approach presented in this thesis can solve a general class of addition and subtraction arithmetic word problems without any predefined equation templates. In particular, it can handle an arithmetic problem as shown in Table 1.

Example 1:
For Halloween, Debby and her sister combined the candy they received. Debby had 32 pieces of candy while her sister had 42. If they ate 35 pieces the first night, how many pieces do they have left?

Table 1: Example Arithmetic Word Problem.

To derive the solution to this problem, the approach needs to understand that *they* refers to *Debby and her sister*. Hence, the number of candies need to be summed up and then *35 candies* need to be subtracted from the total number of candies.

While a solution to these problems requires extracting information and composing numeric expressions, if the sentence is too complex it is hard to extract information accurately.

At the heart of the technical approach, the novel notion of *Sentence Simplification* is involved. Once the sentences in the problem text are simplified, extracting information becomes easier. Each sentence in the problem is simplified to a level where it consists information which is able to be mapped to a single operator. This allows us to decompose the entire problem to a collection of simplified sentences as operator prediction problems. Each sentence represents the quantitative information with the mapped operator. These predictions(operators with the quantitative information) can be combined together to form a mathematical equation.

The approach focuses on addition and subtraction problems currently, but learning to classify operators will allow us to generalize the approach to multiplication and division problems as well. In particular, the system was able to solve Example 1 although it had never seen the problem before and required both addition and subtraction operations.

The approach is evaluated on 3 datasets, achieving competitive performance on all of them. The next section describes the related work in the area of automated arithmetic word problem solving. The theory of sentence simplification is then presented. Later the information decomposition strategy that is based on it is discussed. Section 4 presents the overall computational approach, including the way classifier learns to classify simplified sentences to operators. Finally, experimental study is discussed followed with a conclusion.

2 Related Work

Most of the previous work in automated arithmetic problem solvers has focused on a restricted subset of problems.The approach described in (Roy et al., 2015) handles problems with all basic operations but makes assumptions about the number of quantities in the problem text and the number of steps required to solve the problem. In contrast, our approach does not make any assumptions about the data in the problem text. Kushman's approach (Kushman et al., 2014) tries to map numbers from the problem text to predefined equation templates and implicitly assume that similar equation forms have been seen in the training data. On the contrary, our system does not rely on pre-defined templates and hence is able to solve questions of type which have never been seen before. The approach described in (Hosseini et al., 2014) might be the most related to ours. It handles addition and subtraction problems and tries to predict an operator for the verb in the problem text. Hence, it requires additional annotated data for verb categories. Our approach uses the information of the verb present in the sentence and handles addition and subtraction problems for now, but there is no requirement of additional annotated data. Refer to Section 5 for how the information from verbs is used as a feature.

Most of the methods mentioned above are able to solve problems with all basic operations but our approach is easily generalizable to all the operators and also it performs competitevely compared to all other approaches.

3 Arithmetic Problem Representation

Our approach addresses word problems that include addition and subtraction operations. Given a problem text, multiple fragments are extracted from it where each fragment is a simplified version of the information presented in the sentence. We refer to these fragments as simplified sentences. Each fragment is represented based on the Parts of Speech it contains. Below are the parts of speech we consider in our representation:

Adjective: A problem text might have multiple types of same entities. Consider the below example:

A pet store had 13 siamese cats and 5 house cats.
siamese cats
house cats.

Table 2: Entities with Adjective.

Though the example sentence has cats as the noun, but based on the adjectives*(siamese and house)* there were siamese cats and house cats.

Cardinal: Cardinals are the numbers present in the problem text. We associate them to nouns based on the index at which they occur. Mostly, they occur before the noun and hence associating it is not difficult. Sometimes, the cardinal is a reference to an already occured noun. Consider the example below:

He bought 2 games from a friend and bought 5 more at a garage sale.
2 games
5 games

Table 3: Entity with Cardinal Numbers.

5 more in the above example refers to games. We associate this to the last quantified noun encountered in the sentence.

Conjunction: Conjunctions are stored mainly to simplify the sentence. Most importantly, the index at which it occurs in the sentence plays a crucial role in creating simplified sentences.

Determiner: Determiner by definition is *a modifying word that determines the kind of reference a noun or a noun group has.*

There are 28 students and every student has their own lunchbox.
Every is a determiner

Table 4: Sentence with Determiner.

In the above example, *Every* is a determiner and when considering it, our system will be able to extract the information that there are *28 lunchboxes*.

Existential/Expletive: Since, existentials indicate the existence or presence of something, they are an important part of speech for our system. Also, when the sentences are simplified, the existentials are added to the fragments which don't have them. Refer to the next section for more details.

There are 11 rulers and 34 crayons in the drawer.
There are 11 rulers in the drawer.
There are 34 crayons in the drawer.

Table 5: Simplified sentences based on Existential.

After the split based on conjunction *and* the expletive *There* is added to the fragment after the conjunction.

Noun: Nouns are important for answering the arithmetic word problems. Each problem text has some nouns in form of a subject *(acting entity)* or an object *(Entity that is acted upon by the subject)*. All the nouns are extracted from the simplified sentence and stored as a list.

Preposition: Prepositions are helpful to keep track of location or time. Also, when simplifying sentences based on conjunctions we add prepositions to the fragments which are missing them. Refer to the next section for more details and example.

Verb: Verbs are important for out system to predict the type of action taken by the subject. Determining the kind of action helps to predict an operation. More details on this in Section Features.

WHAdverb: Existence of a WHAdverb mostly indicates the beginning of a question. Hence, having the WHAdverb in our representation helps us in the classification process. Refer to the Section Features.

4 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.

Example 2:
Henry had 11 dollars. For his birthday he got 18 more dollars but spent 10 on a new game. How much money does he have now?

Table 6: Example Arithmetic Word Problem.

The second sentence in the above example is complex for a machine learning algorithm. It has addition and subtraction operation in a single sentence. Our idea is to simplify the sentence to multiple simple sentences so that each simplified sentence has a single operation. Below are the simplified sentences for the second sentence in the above example:

For his birthday he got 18 more dollars but spent 10 on a new game.
For his birthday he got 18 more dollars.
He spent 10 dollars on a new game.

Table 7: Simplified Sentences.

We create a mapping for each sentence in the problem text to its simplified sentences by extracting their relational dependencies for each sentence from the stanford dependency parser[1]. Currenly, our simplification system simplifies sentences based on conjunctions and commas. There are certain rules when simplifying the sentence.

4.1 Rules for simplifying sentences based on conjunctions.

When a conjunction is encountered, our simplification system attempts to create two simplified sentences from the actual sentence. The first sentence is the part before the conjunction while the second sentence is the part after the conjunction.

[1] http://stanfordnlp.github.io/CoreNLP/

But, after the split there may be some words which would be required in the second sentence. Consider the example in Table 4:

Example Sentence.
The school cafeteria ordered 42 red apples and 7 green apples for students lunches.

In the sentence above, the split based on conjunction *and* will result in the first sentence with a subject while the second will not have a subject and a verb. Hence, there are some rules for adding words to the second simplified sentence.

4.1.1 Rules for adding words to second simplified sentence.

1. If the first sentence starts with an existential and the second does not, add the existential to the second sentence as well.

2. If the first sentence starts with an existential and has a verb after it, If the second sentence does not have either expletive or verb, add them to the second sentence. Consider the below example:

There were 2 siamese cats and 4 house cats.
There were 2 siamese cats.
There were 2 house cats.

Table 8: Example sentence for this rule.

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

3. If the first sentence starts with a noun, and if the second sentence starts with a verb, the noun from the first sentence will be added to the second.

Joan ate 2 oranges and threw 3 apples.
Joan ate 2 oranges.
Joan threw 3 apples.

Table 9: Example sentence for this rule.

4. If the first sentence starts with a noun and the second sentence has a *noun verb* pattern, do nothing.

Tom has 9 yellow balloons and Sara has 8 yellow balloons.
Tom has 9 yellow balloons.
Sara has 8 yellow balloons.

Table 10: Example sentence for this rule.

No words from the first sentence were added to the second since the second sentence had the *noun verb (Sara has)* pattern.

5. If the second sentence contains a preposition at the end and the first sentence does not have a preposition, the preposition from the second sentence will be added to the first. Consider the below example:

Joan found 6 seashells and Jessica found 8 seashells on the beach.
Joan found 6 seashells on the beach.
Jessica found 8 seashells on the beach .

Table 11: Example sentence for this rule.

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

6. 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.

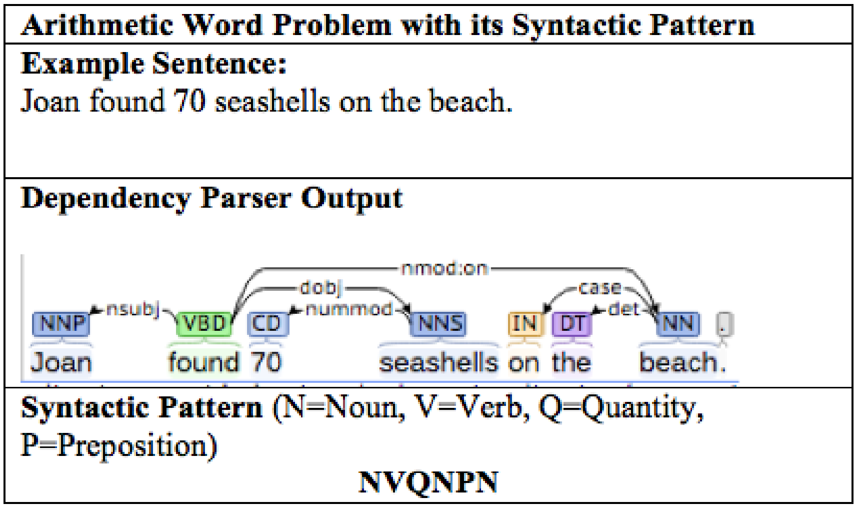
7. Therefore, the sentence simplification system identifies all the words which were not identified by our process. The words which appear before the conjunction are added to the first sentence at the right index. If the words appear after the conjunction, they are added to the second sentence.

He went to the orchard and picked peaches to stock up .
Tom has 9 yellow balloons.
Sara has 8 yellow balloons.

Table 12: Example sentence for this rule.

4.2 Syntactic Pattern

In this thesis, a new concept known as Syntactic Pattern is introduced which is basically a summarized information about a fragment. It is a pattern consisting of the initials of the parts of speech which appear in the fragment. The initials are arranged in the order in which the words appear in the sentence. Consider the below figure:



Arithmetic Word Problem with its Syntactic Pattern

Example Sentence:
Joan found 70 seashells on the beach.

Dependency Parser Output

Syntactic Pattern (N=Noun, V=Verb, Q=Quantity, P=Preposition)

NVQNPN

Figure 1: Example sentence of a word problem and its syntactic pattern.

As in the above example, the pattern *NVQNPN* consists of the initials of the parts of speech in the sentence in their appearance order. This pattern is helpful in many ways:

Grouping Multiple Fragments: Many fragments will have a similar syntactic pattern. Our findings suggest that the fragments having a similar pattern perform similar type of operation. Refer to the below example:

Example Sentence	**Syntactic Pattern**	**Operation**
Joan found 70 seashells on the beach.	**NVQNPN**	Addition
Mary took 10 rulers from the drawer.	**NVQNPN**	Subtraction
Dave gave some rulers to Mary.	**NVDNPN**	Assignment

Table 13: Example sentences and their Syntactic Pattern.

The first two examples have the same pattern and hence perform a similar operation *Addition and Subtraction.* The last sentence has a different pattern and performs an assignment operation which is different than the basic math operations. Refer to Section Features for more information.

Metadata: The pattern acts as a metadata for our fragments. Based on our representation, it is easy to determine if the fragment consists of any conjunction, quantities or any other parts of speech. This information is used in Section Features.

5 Operator Prediction Classifier

We train 4 Logistic Regression binary classifiers each for different operators *(+, -, =, ?).* We then use one versus all technique tocombine the results from the binary classifiers and determine, given a fragment f from a problem text p, which one of the operations f is most likely to perform. Our training data consists of 350 labeled fragments.

5.1 Features

5.1.1 Syntactic Patterns

From 350 training fragments, we were able to extract 125 syntactic patterns. This proves that multiple fragments had same patterns. We use these patterns as individual features. The pattern to which the fragment belongs is set*(on)* and the other patterns are not set*(off)*. We use one-hot encoding approach for pattern features.

5.1.2 Occurence based Features

Relation based occurence features: Dependency relations in a fragment depict its structure. Presence of some dependency relations is useful for the classifier to predict operations. Particularly, *nmod:poss* relation if exists in the fragment, can determine information about a subject possessing an object. *nmod:of* helps to differentiate between two nouns of same kind. Below are the examples:

Sam had 20 pieces of cake and 10 pieces of butter in his plate.
nmod:of(pieces-4, cake-6)
nmod:of(pieces-9, butter-11)
nmod:poss(plate-14, his-13)

Table 14: Example relations in the fragment.

As in the above example, it is easy to see that presence of these relations is mostly in the fragments in which some operation takes place.

Pattern based occurence features: Occurence of some sprecific sub pattern in the syntactic pattern of the fragment can be helpful. Specifically, **NVQN** and **QN** are used as features for the classifier. Their presence again indicates some operation. Consider the below example:

Sentence	**Syntacttic Pattern**
George had 30 dollars.	**NVQN**
18 cupcakes were sold.	**QNVV**

Table 15: Example occurence of a sub pattern.

In the first exmaple, both *NVQN* and *QN* appear. While, in the second *QN* appears. Presence of quantity in a fragment is more likely to suggest an operation.

Word based occurence features: Occurence of some particular words in the fragment help the classifier. Below is list of some words and description of how they are helpful:

Word	**Description**	**Example**
now	Suggests a resulting output after some operations.	28 students wanted to have fruit. There are now 12 fruits remaining.
most, several, some	Suggest an unknown quantity in the fragment.	Joan gave some of her seashells to Sam.

Table 16: Examples for occurence of specific words.

5.1.3 Count based Features

Based on our analysis we found that having a count of some parts of speech occuring the the syntactic pattern proves to be helpful to the classifier in learning accurately. Refer to Table 14 for more details:

Feature	**Description**
Number of Nouns.	Generally a fragment having 2 nouns*subject and object* is more likely to be predicted to addition or subtraction operation.
Number of Prepositions.	A single preposition(such as *from* or *to* is suggesting some basic math operation.
Number of Verbs.	This is basically to indicate that of a verb exists, the fragment suggests an operation.
Number of Quantities.	Similar to verbs, existence of quantities in the fragment is more likely to predict an operation.

Table 17: Count based Features.

5.1.4 Verb similarity based features

Verbs are important to determine the kind of action the fragment is trying to perform. The verbs present in the sentence are given as an input to WordNet*(a large lexical database of English. Nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms)* and Word2Vec*Computes vector representations of words.*

For an input verb v, WordNet outputs a set of categories *(c_1, c_2,...,c_n)* to which the verb is most likely to belong. We shortlisted 8 verb categories which can prove helpful to our system.

Verb Categories.
verb.possession
verb.change
verb.communication
verb.consumption
verb.contact
verb.creation
verb.motion
verb.weather

Table 18: Shortlisted Verb Categories.

We use Word2Vec to get 3 nearest words to the verb and send those words as an input to WordNet to get the verb categories they belong. We normalize the results for these 3 words and the results for the verb itself. Hence, In total we have 8 normalized similarity based features(verb categories)

5.2 Logistic Regression Classifier

Logistic Regression tries to fit a model of the form $p(y|X)$ which directly models the mapping from input X to output y. Since, it discriminates between class labels it is known as a discriminaive classifier. LR corresponds to following binary classification model:

$$p(y|X, W) = Ber(y|sigm(W^T X)) \tag{1}$$

Each fragment will to be classified to one of the operators/class mentioned in the below table:

Operator	**Description**
+	Addition operation.
-	Subtraction operation.
=	Assignment operation.
?	The fragment is asking some question.

Table 19: Logistic Regression output classes.

Each class has its own binary LR classifier. We use an algorithm known as **Stochastic Gradient Descent (SGD)** to train all the four LR classifiers. Below is the algorithm:

Algorithm 1 Stochastic Gradient Descent (SGD)

```
Initialize $\theta$, $\eta$, $b$;
  repeat
    Randomly permute data;
    for i = 1:N do
      s = sigm($X_i$);
      updateWeightVector(s, $X_i$);
      g = updateGradient(s, $y_i$);
      Update b;
    end for
  until
```

We have 3 parameters as shown above: θ(weight vector), η(learning rate) and b(bias). In SGD, we repeatedly run through the training set, and each time we encounter a training example, we update the parameters according to the gradient of the error with respect to that single example only. We calculate the error based on sigmoid function value. A sigmoid function is a mathematical function having an S shaped curve **sigmoid curve**. Below is the sigmoid function:

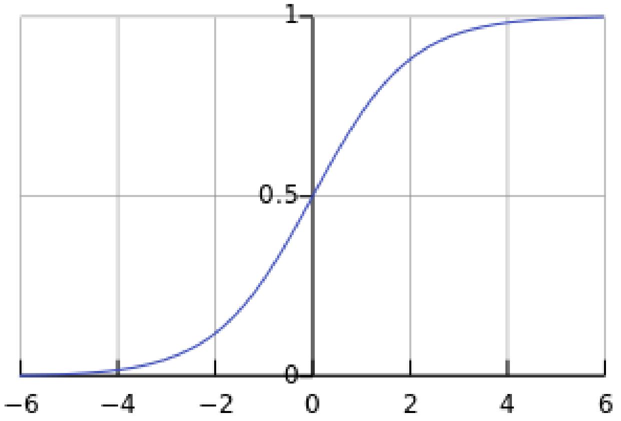


Figure 2: Sigmoid Curve.

The function is defined by the below formula:

$$S(t) = \frac{1}{1+e^{-t}}$$
$$t = \theta_T X$$
$$\theta \rightarrow Weightvector$$
$$X \rightarrow Featurevectorforanexample$$

The function has finite limits and approaches negative infinity from the left side and to infinity from the right. When the algorithm encounters an example, it updates the weight for a particular feature in the weight vector if the feature is set in the current example's feature vector. Based on the sigmoid function value, the gradient and all the parameters are updated.

5.2.1 Training the classifier

We train our four LR binary classifiers on 300 operator labeled fragments and then use one versus all*(OvA)* technique to combine the results from all four of them. The OvA strategy involves training a single classifier per class(4 operators in our case). The samples of that class are treated as positive and all other samples are treated as negative. For example, when training a classifier for + operator, the training samples having + label are positive and samples having other labels i.e. **-**, **=**, **?** are considered negative. This strategy requires the base classifiers to produce a real-valued confidence score for its decision, rather than just a label. Once we have a real valued score from all the base classifiers, the technique predicts a single class using the below mechanism:

$$y' = \underset{k=1,\ldots,K}{\arg\max} f_k(x) \tag{2}$$

5.2.2 Precision and Recall

Our testing data consists of 55 fragments. Below is the distribution of fragments among all the operations and their precision and recall values:

Operator	**No. of instances**	**Precision**	**Recall**
+	32	95.83%	71.87%
-	9	40%	66.66%
=	2	50%	50%
?	15	82.35%	93.33%

Table 20: Precision and Recall

The higher precision and recall of **?** is well understood as there were features such as the presence of an WHAdverb which made it easy for the classifier to predict them accurately. Whereas, for + the verb based similarity features played an important role. For **-** and = its not reallt correct to judge since there were less number of testing fragments.

5.3 Experimental Results

In this section, we evaluate the proposed method on publicly available datasets of arithmetic word problems. We show the parameter setup of our LR classifiers. Lastly, we evaluate the performance of the full system.

5.3.1 Datasets

We evaluate our system on three datasets, each of which comprise a different category of arithmetic word problems.

1. **A12 Dataset:** This dataset is a collection of 395 addition and subtraction problems, released by (Hosseini et al., 2014). They perform a 3-fold cross validation, with every fold containing problems from different sources. This helps to evaluate robustness to domain diversity. To evaluate our system, we follow the same evaluation setting.

2. **IL Dataset:** This is a collection of arithmetic problems released by (Roy et al., 2015). Each of these problems can be solved by performaing one operation. This dataset consists of problems having all basic operations but we only consider addition and subtraction problems. Though, we cannot compare our results to them, we still achieve a competitive accuracy on addition and subtraction problems. We use a 5-fold cross validation to evaluate on this dataset.

3. **Commoncore Dataset:** This dataset is released by (Roy and Roth, 2015) and consists of 600 word problems. These problems similar to the IL dataset have all basic operations, but we only consider addition and subtraction problems. 200 problems have addition and subtraction operations in them. We perform 6-fold cross validation to evaluate on these problems.

5.3.2 Experimental Setup

As explained in the Section 5.2, the algorithm needs 2 parameters η and b. Below are the values we use for all the four classifiers:

Operator	η	b
+	0.0001	-1.5
-	0.0005	-1.0
=	0.01	0.0
?	0.001	0.0

Table 21: Parameter Values

These values were decided based on performance on trial of certain values. The next section describes the results achieved on the datasets.

5.3.3 Results

The below results are achieved in evaluating our system on datasets mentioned above.

Approach/Datasets	**A12**	**IL**	**CC**
	60%	72.88%*	50%*
(Roy and Roth, 2015)	78%	73.9%	45.2%
(Kushman et al., 2014)	64%	73.7%	2.3%
(Hosseini et al., 2014)	77.7%	-	-
(Roy et al., 2015)	-	52.7	-

Table 22: Accuracy in correctly solving arithmetic problems.

The first row of the above table are the results achieved by our system. We achieve competitive results compared to other approaches. For IL and CC datasets we only evvaluate on addition and subtraction problems and hence the results might not be the exact comparison.

References

Hosseini, M. J., Hajishirzi, H., Etzioni, O., and Kushman, N. (2014). Learning to solve arithmetic word problems with verb categorization. In Moschitti, A., Pang, B., and Daelemans, W., 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.

Kushman, N., Artzi, Y., Zettlemoyer, L., and Barzilay, R. (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. Association for Computational Linguistics.

Roy, S. and Roth, D. (2015). Solving general arithmetic word problems. In Mrquez, L., Callison-Burch, C., Su, J., Pighin, D., and Marton, Y., editors, *EMNLP*, pages 1743–1752. The Association for Computational Linguistics.

Roy, S., Vieira, T., and Roth, D. (2015). Reasoning about quantities in natural language. *Transactions of the Association for Computational Linguistics*, 3:1–13.