# ILLINOIS MATH SOLVER: Math Reasoning on the Web

**Subhro Roy** and **Dan Roth**
University of Illinois, Urbana Champaign
{sroy9, danr}@illinois.edu

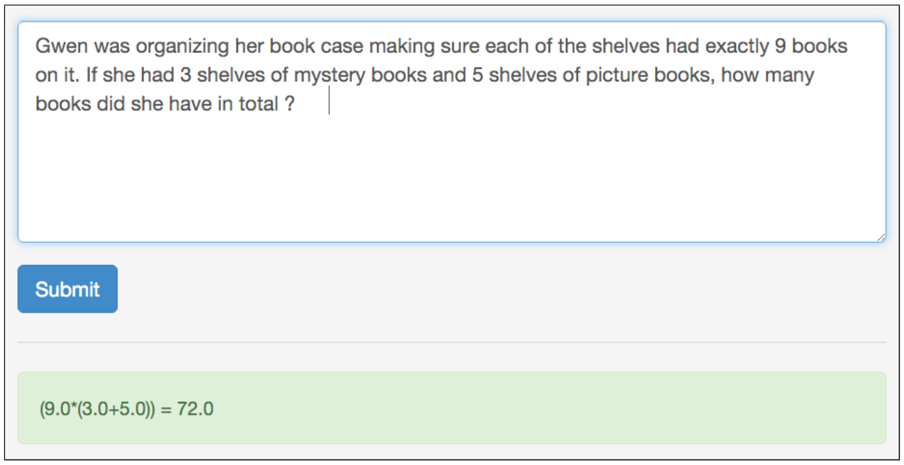
## Abstract

There has been a recent interest in understanding text to perform mathematical reasoning. In particular, most of these efforts have focussed on automatically solving school level math word problems. In order to make advancements in this area accessible to people, as well as to facilitate this line of research, we release the ILLINOIS MATH SOLVER, a web based tool that supports performing mathematical reasoning. ILLINOIS MATH SOLVER can answer a wide range of mathematics questions, ranging from compositional operation questions like *"What is the result when 6 is divided by the sum of 7 and 5 ?"* to elementary school level math word problems, like *"I bought 6 apples. I ate 3 of them. How many do I have left ?"*. The web based demo can be used as a tutoring tool for elementary school students, since it not only outputs the final result, but also the mathematical expression to compute it. The tool will allow researchers to understand the capabilities and limitations of a state of the art arithmetic problem solver, and also enable crowd based data acquisition for mathematical reasoning. The system is currently online at `https://cogcomp.cs.illinois.edu/page/demo_view/Math`.

## 1 Motivation

There has been a lot of interest in understanding text for the purpose of quantitative reasoning. In particular, there has been multiple recent efforts to automatically solve math word problems (Kushman et al., 2014; Hosseini et al., 2014; Roy et al., 2015; Roy and Roth, 2015; Shi et al., 2015; Koncel-Kedziorski et al., 2016). Advancement in this area has great potential to be used as automatic tutoring service for school students. However till date, all the advances in this area are not easily accessible to the general population. ILLINOIS MATH SOLVER addresses this issue by providing a web based platform, where users can type in their math word problem and get the answer. It also outputs the mathematical expression generating the answer, allowing students to understand how to solve the problem. Fig 1 shows a screenshot of the web interface of the ILLINOIS MATH SOLVER.

All systems for math word problem solving are trained and evaluated on datasets created from tutoring websites and textbooks. However the problems from the aforementioned sources tend to have limited variety in problem types and vocabulary. Often these systems are brittle, and make mistakes with slight variation of text. As a result, there is a need for an easy way to analyze the robustness of these systems, as well as extract a wider variety of math word problems not available from textbooks and tutoring websites. ILLINOIS MATH SOLVER solves both these purposes, providing users an easy way to test the robustness of the system, and a tool for crowd based data acquisition. We expect people to query with small edits to math word problem text, to make our system get the wrong answer. This allows for adverserial data acquisition, which can help identify intricacies of mathematical reasoning.


Gwen was organizing her book case making sure each of the shelves had exactly 9 books on it. If she had 3 shelves of mystery books and 5 shelves of picture books, how many books did she have in total ?

(9.0*(3.0+5.0)) = 72.0

**Figure 1:** Screen Shot of ILLINOIS MATH SOLVER

## 2 System Description

The ILLINOIS MATH SOLVER consists of two main modules - a context-free grammar (CFG) based semantic parser, and an arithmetic problem solver. We describe each component below.

### 2.1 CFG Parser

We use a CFG based semantic parser to handle queries asking for operations between numbers. Examples of such queries include *"What is the difference of 22 and 5 ?"* and *"What is the result when 6 is divided by the sum of 7 and 5 ?"*. We refer to such queries as "number queries".

Our parser recognizes the mathematical terms in a number query like "added", "difference", "fraction", etc. It then creates a list of the numbers and math terms mentioned in the query, maintaining the order in which they appear in the query. For the example *"What is the result when 6 is divided by the sum of 7 and 5 ?"*, the parser creates the list $\{6, \text{divided}, \text{sum}, 7, 5\}$.

Next, it tries to parse the list of numbers and the math terms into a mathematical expression, using a list of derivation rules. An example of a derivation rule is as follows:

$$E \rightarrow E_1 \quad \text{divided} \quad E_2$$
$$val(E) \rightarrow val(E_1)/val(E_2)$$

where $E$, $E_1$ and $E_2$ are non-terminals of our CFG representing mathematical expressions. For each such non-terminal, we have an associated function $val(\cdot)$, which computes the numeric value of the mathematical expression represented by that non-terminal. The above rule states that whenever we see the word "divided" between two expressions, we can parse them into a new expression. The value of the new expression will be obtained by dividing the first expression value with the second one. Overall, we have 26 such derivation rules, and we will be augmenting it as we come across more varied number queries. We use CKY algorithm for parsing. The derivation rules naturally capture composition. In the above example, it will first parse $\{$*"sum,* 7, 5*"*$\}$ into an expression $E$, and next parse $\{6,$ *divided,* $E\}$.

### 2.2 General Arithmetic Problem Solver

The second component of our system is the arithmetic word problem solver developed in our previous work (Roy and Roth, 2015). The solver tackles a general class of arithemetic word problems, and achieves state of the art results on several benchmark datasets of arithmetic word problems.

#### 2.2.1 Technical Details

The solver decomposes an input arithmetic problem into several decision problems, and learns predictors for these decision problems. Finally the predictions for the decomposed problems are combined to generate a binary expression tree for the solution mathematical expression. Fig 2 gives an example of an arithmetic word problem coupled with the binary expression tree of the solution.

| Problem |
|---|
| *Gwen was organizing her book case making sure each of the shelves had exactly* 9 *books on it. She has* 2 *types of books - mystery books and picture books. If she had* 3 *shelves of mystery books and* 5 *shelves of picture books, how many books did she have total?* |

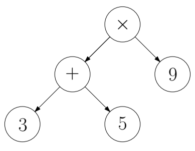
| Solution | Expression Tree of Solution |
|---|---|
| $(3+5) \times 9 = 72$ |  |

**Figure 2:** An arithmetic word problem, solution expression and the corresponding expression tree

The arithmetic solver learns classifiers for the following two prediction problems:

1. For every pair of quantities $q_i, q_j$ in a problem $P$, a classifier is learnt to predict a math operation (one of addition, subtraction, multiplication, division) along with the order of operation (applicable for subtraction and division). This operation is expected to denote the operation at the lowest common ancestor (LCA) node of $q_i$ and $q_j$ in the solution tree. For example, in fig 2, the operation between 3 and 5 is addition, and that between 5 and 9 is multiplication. A multi-class classifier is trained for this prediction task.

   Finally, we define $\textsc{Pair}(q_i, q_j, op)$ to denote the likelihood score of *op* to be the operation at the LCA node of $q_i$ and $q_j$ in the solution expression tree of $P$. The aforementioned classifier is used to obtain these scores.

2. We also train a classifier to predict whether a quantity $q$ mentioned in a problem $P$ is irrelevant for the solution. For example, in fig 2, the number "2" is irrelevant, whereas all other numbers are relevant. A binary classifier is trained to predict this.

   We define $\textsc{Irr}(q)$ to denote the likelihood score of quantity $q$ being an irrelevant quantity in $P$, that is, $q$ is not used in creating the solution. The aforementioned binary classifier is used to obtain these scores.

For an expression $E$, let $\mathcal{I}(E)$ be the set of all quantities in $P$ which are not used in expression $E$. Let $\mathcal{T}$ be an expression tree for $E$. We define $\text{Score}(E)$ of an expression $E$ in terms of the above scoring functions and a scaling parameter $w_{\text{IRR}}$ as follows:

$$\begin{aligned}\text{Score}(E) =& w_{\text{IRR}} \sum_{q \in \mathcal{I}(E)} \textsc{Irr}(q) + \\ & \sum_{q_i, q_j \notin \mathcal{I}(E)} \textsc{Pair}(q_i, q_j, \odot_{LCA}(q_i, q_j, \mathcal{T}))\end{aligned}$$

where $\odot_{LCA}(q_i, q_j, \mathcal{T})$ is the operation at the LCA node of $q_i$ and $q_j$ in the expression tree $\mathcal{T}$.

Our search for solution expression tree is also constrained by legitimacy and background knowledge constraints, detailed below.

1. **Positive Answer**: Most arithmetic problems asking for amounts or number of objects usually have a positive number as an answer. Therefore, while searching for the best scoring expression, we reject expressions generating negative answer.

2. **Integral Answer**: Problems with questions such as 'how many" usually expect integral solutions. We only consider integral solutions as legitimate outputs for such problems.

Let $\mathcal{C}$ be the set of valid expressions that can be formed using the quantities in a problem $P$, and which satisfy the above constraints. The inference algorithm now becomes the following:

$$\arg\max_{E \in \mathcal{C}} \text{Score}(E)$$

### 2.2.2 Evaluation

We evaluated our arithmetic word problem solver on three publicly available datasets – addition subtraction problems from AI2 dataset (AI2) (Hosseini et al., 2014), single operation problems from Illinois dataset (IL)(Roy et al., 2015), and multi-step problems from commoncore dataset (CC)(Roy and Roth, 2015). We compare against systems which had achieved previously known best scores on these datasets, and show that our system achieves state of the art performance on all the above datasets. Table 1 shows the comparison. Finally, the models of the Illinois Math Solver are trained on the union of the aforementioned datasets.

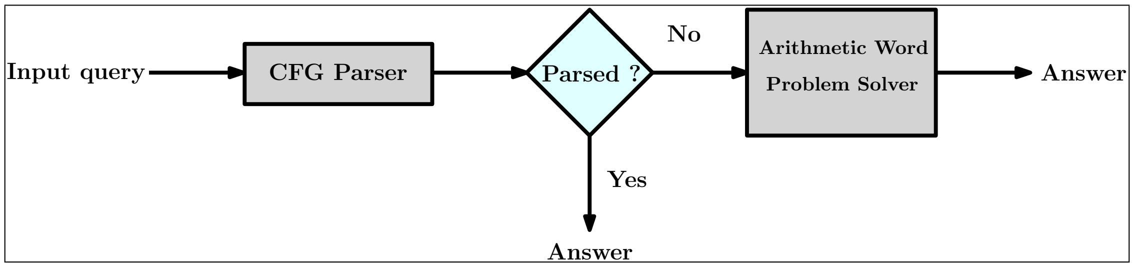


**Figure 3:** Pipeline of ILLINOIS MATH SOLVER

| | AI2 | IL | CC |
|---|---|---|---|
| Our system | **78.0** | **73.9** | **45.2** |
| (Hosseini et al., 2014) | 77.7 | - | - |
| (Roy et al., 2015) | - | 52.7 | - |
| (Kushman et al., 2014) | 64.0 | 73.7 | 2.3 |

**Table 1:** Accuracy in correctly solving arithmetic problems. We achieve state of the art results in all three datasets.

### 2.3 Illinois Math Solver pipeline

The pipeline of the ILLINOIS MATH SOLVER is shown in Fig 3. Given input text, we first run the CFG parser, to check whether it is a number query. If our CFG parser can make sense of the query and can generate a mathematical expression from the query, we immediately output it as the answer. Otherwise, the query is fed to the arithmetic problem solver. The output of the solver is then displayed as the result.

## 3 Related Work

The interface of Wolfram Alpha is probably the closest to ours. However their system is limited to handling mostly number queries, and very simple arithmetic problems. In contrast, our system can solve complicated arithmetic problems described by multiple sentences and requiring multiple operations. There has also been a lot of work in quantitative reasoning. Roy et al. (2015) looks at understanding entailment relations among expressions of quantities in text. There has also been efforts to automatically solve school level math word problems. Hosseini et al. (2014) looks at solving elementary addition subtraction problems, Roy et al. (2015) aims to solve single operation problems and Koncel-Kedziorski et al. (2016) solves single equation problems. The system of Shi et al. (2015) tackles number word problems by semi-automatically generated parsing rules, and is similar to our CFG parsing approach for tackling number queries. Kushman et al. (2014) proposes a template based approach for solving algebra word problems and finally, our system proposed in Roy and Roth (2015) solves a general class of arithmetic word problems, and achieves state of the art results on multiple arithmetic word problem datasets. This is the solver we use for handling arithmetic problems in ILLINOIS MATH SOLVER.

## 4 Conclusion and Future Directions

We release ILLINOIS MATH SOLVER, an online tool to automatically solve number queries and arithemtic word problems. It will help elementary school students to self-tutor. In addition, it will be a source of highly varied math queries, which might reveal difficulties of mathematical reasoning, and assist future advancement in the area.

There are various fronts on which we will be improving the system in future. Currently, the arithemetic solver assumes the final solution can be generated by combining the numbers mentioned in the text, and hence, cannot introduce new numbers for the solution. . For example, *"I eat 1 apple each day. How many apples will I eat in 1 week ?"* is currently not handled since it requires knowing that 1 week has 7 days. This will require leveraging a knowledge base to bring in the additional information. We will also try to handle algebra word problems, which involve generating multiple equations with one or more variables, and then solving these equations to generate the answer.

## References

[Hosseini et al.2014] M. J. Hosseini, H. Hajishirzi, O. Etzioni, and N. 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 2014*.

[Koncel-Kedziorski et al.2016] R. Koncel-Kedziorski, H. Hajishirzi, A. Sabharwal, O. Etzioni, and S. Dumas Ang. 2016. Parsing algebraic word problems into equations. In *TACL*.

[Kushman et al.2014] N. Kushman, L. Zettlemoyer, R. Barzilay, and Y. Artzi. 2014. Learning to automatically solve algebra word problems. In *ACL*.

[Roy and Roth2015] S. Roy and D. Roth. 2015. Solving general arithmetic word problems. In *Proc. of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

[Roy et al.2015] S. Roy, T. Vieira, and D. Roth. 2015. Reasoning about quantities in natural language. *Transactions of the Association for Computational Linguistics*, 3.

[Shi et al.2015] Shuming Shi, Yuehui Wang, Chin-Yew Lin, Xiaojiang Liu, and Yong Rui. 2015. Automatically solving number word problems by semantic parsing and reasoning. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.