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 that needed to determine a satisfactory answer. 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 corresponding quantifiers. 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. An empirical study is conducted on 3 datasets and we achieve results comparable with existing state-of-the-art systems.

1 Introduction

Answering arithmetic word problems has gained significant interest in recent years. The problem is attractive to natural language processing (NLP) methodologies since the text is concise and relatively straightforward with identifiable semantic constraints. As these problems are directed towards el-

ementary 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 that corresponds to the problem text.

There have been a number of attempts to solve arithmetic word problems through machine learning (ML). All the approaches which 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 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 Figure 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?

Figure 1: Example Arithmetic Word Problem.

To derive the solution to this problem, the system needs to understand that *they* refers to *Debby and her sister* (i.e. anaphora/pronominal resolution). 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 (Section 4). 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. Section 2 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 5.2 presents the overall computational approach, including the way classifier learns to classify simplified sentences to operators. Finally, experimental study is discussed followed with future work and 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 5.1 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. The parts of speech we consider in our representation are as follows:

Adjective: A problem text might have multiple types of same entities. Consider the example presented in Figure 2:

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

Figure 2: Entities with Adjective.

Though the 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 Figure 3 for an example:

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

Figure 3: Entity with Cardinal Numbers.

5 more in the sentence 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. Refer to Figure 8 for an example.

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

Figure 4: Sentence with Determiner.

In the example presented in Figure 4, *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 Section 4 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.

Figure 5: Simplified sentences based on Existential.

In the example in Figure 5, 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 Section 4 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 5.1.

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 5.1.

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:
s: 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?
s_1: Henry had 11 dollars.
s_2: For his birthday he got 18 more dollars but spent 10 on a new game.
s_3: How much money does he have now?

Figure 6: Example Arithmetic Word Problem.

Consider the example presented in Figure 6. Sentence s_2 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. The simplified sentences for s_2 are listed in Figure 7.

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.

Figure 7: Simplified Sentences.

We create a mapping for each sentence in the problem text to its simpli-

fied sentences by extracting their relational dependencies from the Stanford dependency parser[1]. Currenly, our system simplifies sentences based on conjunctions. There are certain rules when simplifying the sentence.

4.1 Rules for simplifying sentences based on conjunctions.

When a conjunction is encountered, our 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.

Notably, after the split there may be some words which would be required in the second sentence. Consider the example in Figure 8.

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

Figure 8: Example Sentence

Given a sentence s from problem text p, after simplification process it will have n simplified sentences i.e. $< s_1, ..., s_n >$. In the sentence s, the split based on conjunction *and* will result in two sentences.

s_1:*The school cafeteria ordered 42 red apples*

s_2:*7 green apples for students lunches*

s_1 will have a subject while s_2 will not have a subject and a verb making it an improper sentence. Hence, there are some rules for adding words to simplified sentences:

4.1.1 Rules for adding words to simplified sentences.

1. If s_1 starts with an existential and has a verb after it and if s_2 does not have either expletive or verb, distribute them to s_2. Consider the example in Figure 9:

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

s: There were 2 siamese cats and 4 house cats.
s_1: There were 2 siamese cats.
s_2: There were 2 house cats.

Figure 9: Example sentence for Rule 1

.

The expletive and verb were added to s_2 based on the simplification rule mentioned above.

2. If s_1 starts with a noun, and if s_2 starts with a verb, the noun from the former will be distributed to the latter. Refer to an example in Figure 10.

s: Joan ate 2 oranges and threw 3 apples.
s_1: Joan ate 2 oranges.
s_2: Joan threw 3 apples.

Figure 10: Example sentence for Rule 2.

3. If s_1 starts with a noun and s_2 has a *noun verb* pattern, do nothing.

s: Tom has 9 yellow balloons and Sara has 8 yellow balloons.
s_1: Tom has 9 yellow balloons.
s_2: Sara has 8 yellow balloons.

Figure 11: Example sentence Rule 3.

In the example presented in Figure 11, No words from s_1 were added to s_2 since it had the *noun verb (Sara has)* pattern.

4. If s_2 contains a preposition at the end and s_1 does not, it will be distributed from s_2 to s_1. Consider the example presented in Figure 12:

s: **Joan found 6 seashells and Jessica found 8 seashells on the beach.**
s_1: Joan found 6 seashells on the beach.
s_2: Jessica found 8 seashells on the beach.

Figure 12: 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 s_1 at the correct index and if they appear after the conjunction, they are added to s_2.

4.2 Syntactic Pattern

In this thesis, a 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 picture in Figure 13:

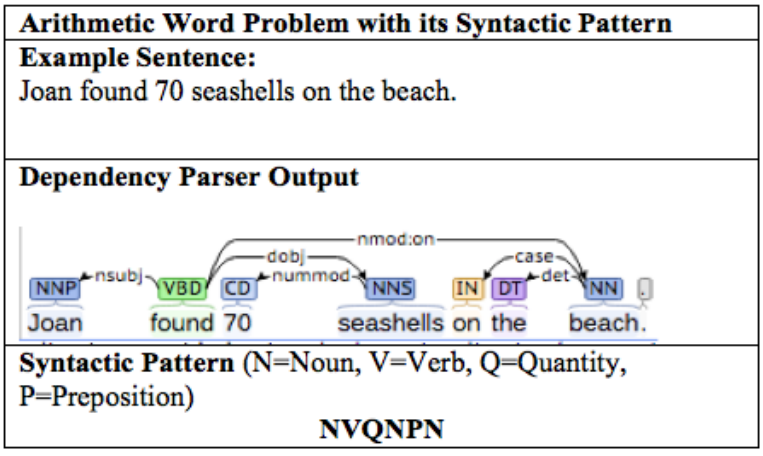

Figure 13: Example sentence of a word problem and its syntactic pattern.

As in Figure 13, 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 example in Figure 14:

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

Figure 14: 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 5.1 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.

Example Sentence	**Syntactic Pattern**
Joan found 70 seashells on the beach.	**NVQNPDN**
Parts of Speech	**Count**
Adjective	0
Cardinal (Quantity)	1
Conjunction	0
Determiner	1
Existential/Expletive	0
Noun	3
Preposition	1
Verb	1
WHAdverb	0

Figure 15: Decomposing the Syntactic Pattern.

To be able to extract this information in a simple way is helpful to our system. Refer to Section 5.1 for more details.

5 Operator Prediction Classifier

We train 4 Logistic Regression binary classifiers each for different operators *(+, -, =, ?)*. We then use one versus all technique to combine the results from the binary classifiers. Given a fragment f from a problem text p, we determine which one of the operations, fragment f is most likely to perform. We have around 94 questions from which 301 fragments were extracted. We label these fragments and perform a 5-fold cross validation to train our classifier. Refer to Figure 16 for distribution of fragments among class labels.

Class Label	**Count**
+	137
-	57
=	13
?	94
Total	301

Figure 16: Count of traning fragments for each class label.

5.1 Features

5.1.1 Syntactic Patterns

From 301 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. For a given fragment, the pattern to which it 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 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. The examples are listed in Figure 17:

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)

Figure 17: Example relations in the fragment.

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 specific 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 example in Figure 18:

Sentence	**Syntactic Pattern**
s_1: George received 30 dollars.	**NVQN**
s_2: 18 cupcakes were sold.	**QNVV**

Figure 18: Example occurence of a sub pattern.

In s_1, both *NVQN* and *QN* appear. While, in s_2 *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. Figure 19 has the list of some words and description of how they are helpful:

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

Figure 19: 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 Figure 20 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 the movement of object.
Number of Verbs.	If there are no verbs in the fragment, there should ideally be no actions performed.
Number of Quantities.	If there is no quantity in the fragment, it is more likely that math operations have not been performed.

Figure 20: 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)* (Miller, 1995) and Word2Vec(*Computes high-quality word vector representations of words from huge datasets with billions of words*) (Mikolov et al., 2013).

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 as listed in Figure 21 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

Figure 21: 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).

Step 1: Pass the identitified verb in the sentence as an input to WordNet. Refer to Figure 22 for an example:

Example Sentence: Joan baked 10 cupcakes.
Verb: baked
Sense 1 <verb.change>bake =><verb.change>cook
Sense 2 <verb.creation>bake =><verb.creation>create from raw material, create from raw stuff
Sense 3 <verb.change>broil1, bake2 =><verb.change>heat1, heat up

Figure 22: Using WordNet on the Verb.

Step 2: Get the top 3 closest words to the verb using Word2Vec. Figure 23 consists an example:

Word2Vec for verb: *baked*
["unbaked", 0.602738618850708]
["cooked", 0.6022148728370667]
["bakes", 0.5771548748016357]

Figure 23: Using WordNet on the Verb.

Step 3: Pass the closest words as an input to WordNet to identify their verb categories. Similar to Figure 22, each word will have their own

distribution of categories. Hence, we normalize results between 8 categories as in Figure 24.

Verb Category	**Probability**
verb.possession	0
verb.change	0.6
verb.communication	0
verb.consumption	0
verb.contact	0
verb.creation	0.4
verb.motion	0
verb.weather	0

Figure 24: Normalized results for verb categories.

Alongside other features, these values are also used as features for our classifier.

5.2 Logistic Regression Classifier

Logistic Regression (LR) 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 discriminative classifier. LR corresponds to following binary classification model (Murphy, 2012):

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

Each fragment will to be classified to one of the operators/class mentioned in Figure 25:

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

Figure 25: 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. Refer to the algorithm presented in 26:

Algorithm 1 Stochastic Gradient Descent (SGD)

Initialize θ, η, 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**. Figure 26 shows the sigmoid function.

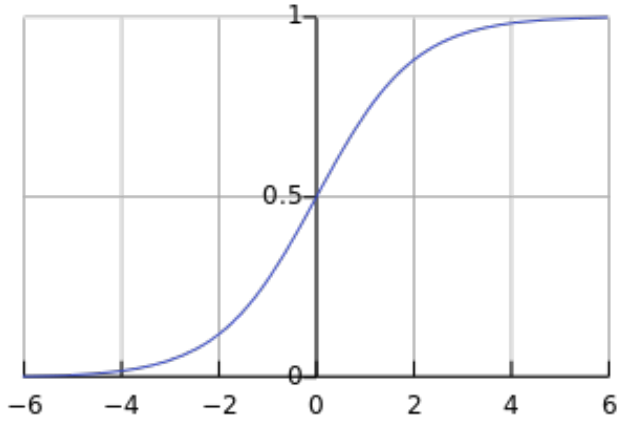

Figure 26: Sigmoid Curve.

The function is defined by the below formula:

$$S(t) = \frac{1}{1 + e^{-t}}$$
$$t = \theta_T X$$
$$\theta \rightarrow \textit{Weight vector}$$
$$X \rightarrow \textit{Feature vector for an example}$$

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 301 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. Figure **??** consists of the distribution of fragments among all the operations and their precision and recall values:

The higher precision and recall of operation **?** 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 operations **-** and = its not really correct to judge since there were less number of testing fragments.

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

Figure 27: Precision and Recall

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. Tthe values we use for all the four classifiers are mentioned in Figure **??**:

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

Figure 28: Parameter Values

These values were decided based on performance on trial of certain values. Section 5.3.3 describes the results achieved on the datasets.

5.3.3 Results

The results in Figure 29 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	-

Figure 29: Accuracy in correctly solving arithmetic problems.

* *These datasets contain problems for all operations. From these datasets, we have evaluated our system for addition and subctraction problems only. Hence, the results for these datasets are for informational purposes.*

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 if not be the correct comparison.

6 Discussion

Our system is generalized as it gives importance to every fragment initially and then based on its structure ignores irrelevant information. Not only verbs(Hosseini et al., 2014), but other parts of speech play an equal role in solving arithmetic word problems.

6.1 Future Work

6.1.1 Multiplication and Division Problems

We aim to solve multiplication and division problems using a similar approach. Based on our analysis, we believe that by adding additional features and training data, multiplication and division problems could be solved.

6.1.2 Additional Complex Parsing Rules

Currently, our system employs some complex but mostly simpl rules to solve arithmetic problems. We aim to add more complex parsing rules to extract quantified nouns and their relations with verbs. This would help the classifier to predict operations more accurately.

6.1.3 Improving the Classifier

We also plan to focus on improving the classifier by using some advanced machine learning methods. At this point, we have a LR classifier which performs reassonably well. This is an area to predict the operator for a fragment more accurately. We plan to consider ensembling methods in future.

6.2 Error Analysis

Error Type	**Example**
Classification Errors (75%)	He now has 56 books in his library. **Classified to "+" but should be "=".**
Parsing Issues (15%)	Sara has 31 **red** and 15 green balloons.
Coreference Resolution (10%)	When they cleaned them , they discovered that **29** were cracked.

Figure 30: Examples of different error categories and relative frequencies.

As per Figure 30, we can see the need to improve the classifier based on the number of classification errors. There is a possibility of improvement in parsing issues by adding more rules.

7 Conclusion

This thesis presents a method for understanding and solving addition and subtraction arithmetic word problems. We develop a novel theoritical framework, centered around the notion of syntactic patterns and implement it to produce simplified sentences. This theory naturally leads to a solution to uniquely solve word problems - *determine the quantified nouns and operation in the problem text*. The theory underlies our algorithmic solution. 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 the particular problem type before.

Acknowledgements
I thank Kevin Small and other reviewers for guidance, motivation, helpful reviews and feedback on the work.

References

Hastie, T., Tibshirani, R., and Friedman, J. (2001). *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA.

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.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781.

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

Murphy, K. P. i. (2012). *Machine learning : a probabilistic perspective*. Adaptive computation and machine learning series. MIT Press, Cambridge (Mass.).

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.

Wasserman, L. (2010). *All of Statistics: A Concise Course in Statistical Inference*. Springer Publishing Company, Incorporated.