# Machine-Guided Solution to Mathematical Word Problems

**Bussaba Amnueypornsakul**
University of Illinois,
Urbana-Champaign, USA
amnueyp1@illinois.edu

**Suma Bhat**
University of Illinois
Urbana-Champaign, USA
spbhat2@illinois.edu

## Abstract

Mathematical word problems (MWP) test critical aspects of reading comprehension in conjunction with generating a solution that agrees with the "story" in the problem. In this paper we design and construct an MWP solver in a systematic manner, as a step towards enabling comprehension in mathematics and teaching problem solving for children in the elementary grades. We do this by (a) identifying the discourse structure of MWPs that will enable comprehension in mathematics, and (b) utilizing the information in the discourse structure towards generating the solution in a systematic manner. We build a multistage software prototype that predicts the problem type, identifies the function of sentences in each problem, and extracts the necessary information from the question to generate the corresponding mathematical equation. Our prototype has an accuracy of 86% on a large corpus of MWPs of three problem types from elementary grade mathematics curriculum.

## 1 Introduction

Mathematical word problems (MWP) constitute an integral part of a child's elementary schooling curriculum. Solving an MWP is a complex task involving critical aspects of reading comprehension (understanding the components of the problem), and generating a solution that agrees with the 'story' in the problem. Children are trained through the process of problem solving by the use of various strategies. In this study, we formulate solving an MWP as an NLP task involving text classification, discourse processing and information extraction. Our primary goal is to guide young learners through the important steps of mathematics comprehension and problem solving of arithmetic word problems commonly encountered in the elementary grades. We take a bottom-up approach, identifying the discourse structure of the MWP and then utilizing the semantic information contained in the components of the problem to generate a solution.

In an MWP, significant background information is presented in text format. The ability to solve an MWP critically depends on the ability to detect the problem type and identify the components of the word problem as observed in studies in mathematics education and cognitive psychology (De Corte and Verschaffel, 1987; Cummins, 1991; Verschaffel et al., 2000).

Motivated by these studies, we divide the overall problem solving process into stages: *predicting the problem type*, *identification of the function of sentences* (or sentence type) in each problem, and extracting the necessary information from the question to *generate the corresponding mathematical equation*. Since classification of the problem and sentence types involves a decision based on the textual representation, the classification tasks can be viewed as automatic text categorization problems (Yang and Liu, 1999) with domain-specific feature engineering. More broadly, a knowledge of the discourse structure of an MWP provides the human solver with a critical first step for information extraction and text summarization needed for mathematics problem comprehension and solving.

A text classification perspective to MWP solu-


28th Pacific Asia Conference on Language, Information and Computation pages 111–119

tion calls for an approach different from routine text classification methods. Surface word statistics and a keyword spotting approach, that convey topicality, for instance, are insufficient to derive necessary information about problem type or document structure owing to the short document lengths of MWP. Stop word removal and stemming, two common preprocessing steps in text classification by topic, have been observed to negatively impact classification of problem types (Cetintas et al., 2009). Thus, feature engineering that *leverages* the natural language properties of word problems not only at a sentence level but also at a problem level is an important novelty in this study as we explore the usefulness of a text classification approach to solving MWPs. In addition, our study is novel in adopting the multistage approach to solving word problems automatically.

Specifically, this paper makes the following contributions.

1. Taking a text classification approach towards automatically identifying the information structure of MWPs, we show empirically that an ensemble classifier yields the best performance for identifying the problem type and for identifying the discourse structure of MWP. Not only are the performance gains over the baseline vastly substantial, but the performance gains of the solver when compared with state-of-the-art MWP solvers such as WolframAlpha (Barendse, 2012) are also substantial.
2. We demonstrate the efficacy of our software prototype to solving MWPs automatically. The multistage approach can be construed as a careful combination of inductive inference (statistical methods) and deductive inference (rule-based approach) to reflect the key aspects of mathematics comprehension in arithmetic problem solving as pointed out in psychology studies: The use of natural language to identify the discourse structure and a set of rules to derive the corresponding mathematical form (De Corte and Verschaffel, 1987; Cummins, 1991; Verschaffel et al., 2000).

## 2 Related Work

Prior studies attempting to solve mathematical word problems in an automatic manner fall into two primary categories: those intended to understand the cognitive aspects of problem solving in children and those intended for intelligent tutoring systems. Prototypical systems such as WORDPRO (Fletcher, 1985), SOLUTION (Dellarosa, 1985), ARITHPRO (Dellarosa, 1986) and (LeBlanc and Weber-Russell, 1996) are representations of cognitive models of human processes of mathematical word problem solving. With the exception of (LeBlanc and Weber-Russell, 1996), these operate on propositional representations of the problem text later solved in a rule-based manner.

In the realm of intelligent tutoring systems automatic MWP solvers were based on either using specific sentence structures and keywords (Bobrow, 1964), or using templates (schema) limited in scope by variety and problem types - (Supap et al., 2013) for grade-level problems in Thai and (Liguda and Pfeiffer, 2011; Liguda and Pfeiffer, 2012) for grade-level problems in German.

An early approach to automatic classification of MWP using natural language processing methods was (Cetintas et al., 2009). The study pointed out that certain problem types (such as the multiplicative compare and equal group) were characterized by their lexical content and that a blind text categorization approach via stop word removal and stemming failed to help the classification task for those problem types. Another related study (Cetintas et al., 2010), addresses sentence-level classification of sentences in MWP into relevant and irrelevant sentences to identify the information-bearing components of the problem.

A more recent study in a related area is (Matsuzaki et al., 2013), which aims at understanding the complexity of MWPs encountered by students appearing for a Japanese university entrance examination. It includes and end-to-end method of problem solving by transforming the question sentences into their logic representation to be eventually solved by an automatic solver. The problems considered are significantly more complex than grade-level arithmetic problems. A semantic parser used on the related topic of learning to solve algebra word problems is the material of (Kushman et al., 2014). In all these studies the goal was to arrive at a solution automatically without paying attention to the step-by-step approach to assisted problem solving which

is what we address in this work.

Taking a view different from that of prior studies, our focus here is two-fold: first, inspired by the approach to identify the structure of scientific abstracts in (Guo et al., 2010), we would like to gain a fundamental understanding of the discourse structure of an MWP which serves as its information-bearing component; second, knowing the structure of an MWP we would like to discover the interrelation between available units of information and eventually solve the problem.

Our approach in this study is closely related to that in (Supap et al., 2013) in spirit, but instead of a top-down approach via having a static template for each problem type, we resort to constructing dynamic templates in a bottom-up fashion using information on problem types and associated discourse structure. The classification algorithm leverages natural language properties at the sentence level as well as across sentence boundaries.

For the classifiers we use a combination of a deductive learner driven by inductive learners which has been very successful in other domains such as electronic design automation tools (Chaganty et al., 2013; Liu et al., 2012). The cognitive modeling perspective to solving MWP in children renders the inductive-deductive learner combination a natural choice for our study.

## 3 Method

Our approach to solving an MWP is grounded in harnessing the information available in the discourse structure of the word problem. We hypothesize that classification of the problem type is a crucial first step. After knowing the problem type, we focus on the solution by identifying the components of the problem and their interrelation.

### 3.1 Data

MWPs have the information to solve them embedded in text rather than in an equation. While recognizing that there are several categories of word problems, we consider for our study the set of word problems considered in a cognitively guided instruction scheme (CGI).

The CGI framework aims at developing a child's mathematical thinking via intuitive strategies for problem solving (Carpenter et al., 2000). Focusing on the curriculum of the cognitively guided instruction scheme, this study aims to solve all three problem types at the elementary grade level: problems of the type *join and separate*, *compare* and *part-part-whole* involving only one mathematical operation - that of addition or subtraction.

The choice of these problem types is motivated by early developmental theories in children's arithmetic competencies that focus on word problems classified into natural classes based on their semantic structures, the relation between the sets in the problem statement.(LeBlanc and Weber-Russell, 1996).

The word problems considered here constitute the major types proposed by the CGI curriculum. The problem types are general in that they do not call for a specific arithmetic operation but we have restricted our approach to only those involving addition and subtraction. Although details of the exact proportion of these word problem types in the respective grade levels is not available, we expect word problems of the types considered here to be prevalent in grades Kindergarten to fourth grade (as evidenced from the collected corpus of sample practice problems).

**Join and separate** (J-S) problems have three main functional types of sentences in a question: given, change and result. A *Given sentence* is a narrative sentence where a quantity is given; a *Change sentence* indicates that there are some changes to the quantity in the *Given sentence* and the *Result sentence* is the result of the change applied to the given quantity. A sentence that is not of the above functional types is an *Unknown sentence*. When the change applied to the given quantity results in a decrease, the problem is of the *separate* kind (subtraction) and when the result is an increase in the given quantity, the problem is of the *join* kind (addition). Problems of this type are characterized by significant action language that describe changes in the possession or condition of objects. As an example consider a problem of the type *separate*:

Henry is walking dogs for money. There are 7 dogs to walk on Henry's street. Henry walked 4 of them. How many dogs does Henry have left to walk?

Note : The yellow highlight is the *given sentence*. The blue highlight is the *change sentence* and the pink highlight is the *result sentence* of the example problem. The remaining sentences are of the type *unknown sentence.*

Equation: 7 - x = 4

**Part-part-whole** (PPW) is the second problem type which contains two main functional types of sentences: part and whole. The *part sentence* indicates the quantity of a set, while the *whole sentence* indicates the total amount in a category that subsumes the set. Problems of this type involve static descriptions of the counts of two or more disjoint subsets and the union of those sets and do not contain significant actions. For example,

Some kids are playing in a playground. 3 boys are playing on the slide. 4 girls are playing on the merry-go-round. How many kids are there in the playground?

Note : The yellow highlight is the *part sentence*. The blue highlight is the *whole sentence*. The rest of the question is the *unknown sentence*.

Equation: 3 + 4 = x

The simplest of the three types, **compare problems** (C) involve a comparison of the counts of two sets. For example, Angela has 6 mittens. Jordan has 4 more mittens than Angela. How many mittens does Jordan have?

It is important to note that in a given problem, the missing quantity could be in the *Given*, *Change* or *Result* sentence (likewise in the *part* or the *whole* sentence). It is also crucial to remember that although the equations corresponding to the problem types are similar, our focus is not just the solution but also the steps leading to the solution. The dataset used in our study is a set of sample problems from the South Dakota Counts (Olson et al., 2008) and teacherweb.com (Ebner, 2011). A brief description of the problems of each type and their characteristics in the corpus is summarized in Table 1.

| **Problem type** | **J-S** | **PPW** | C |
|---|---|---|---|
| No. of problems | 330 | 164 | 257 |
| No. of words/problem (mean) | 25.54 | 22.47 | 21.13 |
| No. of sentences/problem | 3.42 | 2.72 | 3.06 |
| No . of verb types (total) | 99 | 36 | 46 |

Table 1: Corpus description of the set of problems studied.

The problems were grouped by problem type at the source. However, their sentence type annotations were not available. The problems in the dataset were manually annotated for sentence functional type (*Given*, *Change*, *Result*, *Part* and *Whole*) and sign (join or separate) by the researchers. The annotators agreed on 99.4% of the sentence function types.

Notice from Table 1 that the J-S problems constitute a majority of the problem types and that these problems are also the longest in terms of average number of words per problem. Another significant feature is the number of sentences per problem. We notice that it is 3.42 for J-S problems suggesting that there are more than 3 sentences which would be the case when just the *Given*, *Change* and *Result* sentences are present. Again, in the case of PPW sentences, we notice that the sentences are not necessarily *Part*, *Part* and *Whole*, but the 'parts' may even be relegated to the same sentence.

## 3.2 Models

The first stage is problem type classification. Problem type classification takes as input the entire problem divided into sentences and assigns it to one of Join-Separate, Part-Part-Whole or Compare type. Depending on the problem type, the necessary classifiers are cascaded. We divide the problem solution into a maximum of three stages depending on the problem type with a classifier for each stage, described as follows. A schematic representation of the solver is given in Figure 1.

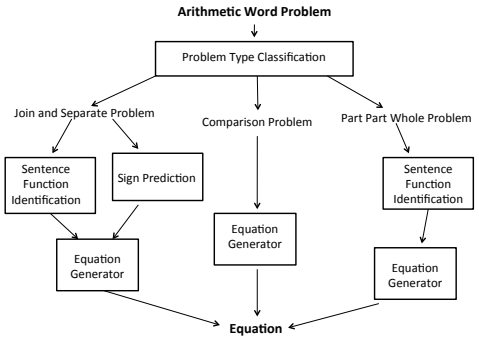


Figure 1: Flow chart for the system.

### 3.2.1 Join and separate problems (JS)

Join and separate problems are the most versatile of problems because the problem's discourse structure affords phrasing of its constituent sentences in many ways. The constituent sentences can either be separate, joined using a conjunction or could be formed as a complex sentence with the use of conditionals.

Figure 2 shows a step-by-step approach to solving problems of this type. First, we **classify the sentence functional type** for each sentence (whether it is Given or Change or Result sentence). Then, we perform a *sign prediction* (whether the problem calls for addition or subtraction). The pivot sentence for this task is the *Change* sentence because it indicates the direction of change of the quantity in the *Given* sentence in terms of an effective increase or decrease.The last task is to combine the results of the first two stages and generate the corresponding equation.

This problem focuses on the relationship between nouns in each sentence of the question. There are two steps to solve this problem. The first step is to identify

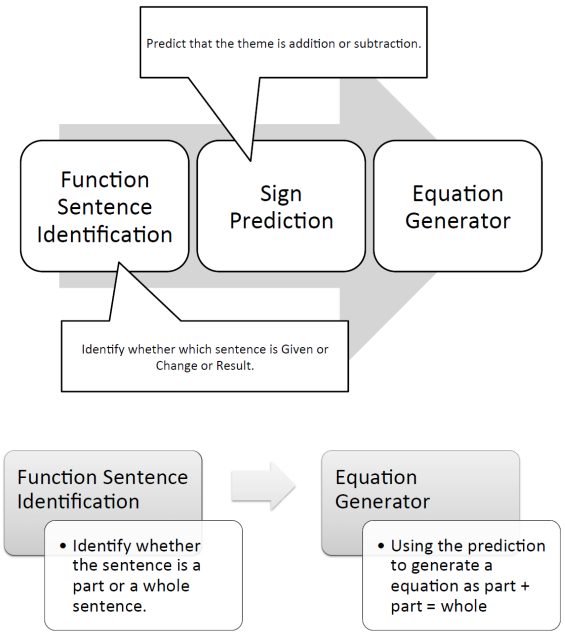


Figure 2: Top: Flow chart for Join and Separate Problem. Bottom: part part whole Problem.

whether the sentence is a *part sentence* or a *whole sentence*. We then use the information from this classification to generate the equation. The flowchart of the problem is displayed in figure 2.

#### 3.2.2 Compare problems

Comparison problems focus on similarities or differences between sets. By nature of its type, the problem's discourse structure is limited. This means we can generate a set of rules to convert a question to its corresponding equation. Once a problem is classified as belonging to this type in the problem type identification stage, the problem is then processed by a rule-based classifier leading to its equation.

#### 3.2.3 Equation generation

Once the component sentence types comprising the discourse structure of the problem are identified the information in each sentence is extracted. We note that the sentence type (and hence discourse structure) plays a crucial role in this stage of information extraction. We use the NLTK toolkit (Loper and Bird, 2002) to extract the numerical quantity from each sentence.

In the J-S equation generator, we construct an equation of the form (quantity in *Given*) + (quantity in *Change*) = *Result*. The quantity in the *Change* sentence bears the sign of the question (depending on whether it is addition or subtraction). If a sentence with no numerical information is classified as *Given, Change* or *Result*, we assign an **X** to that sentence and the information is excluded from the equation (a potential source of error).

The analog holds for the PPW equation generator. With its sentences classified as *Part* or *Whole* we proceed to the equation generation as follows. When the *Part* sentence has more than one numerical quantity, we assign the first number as *Part1* and the other numbers as *Part2* (or into more buckets as the case may be). Then, we arrange them into the corresponding equation as: Part1 + Part2 = Whole.

In both these equation generators, when the equation has insufficient information owing to errors from previous stages (we will defer discussing some scenarios to Section 6), a solution is not generated. The generated equation is solved using Numpy (Oliphant, 2006).

### 3.3 Implementation

For the tasks of problem type classification, sentence type classification and sign prediction, we use the ensemble method of inductive classifier - Random Forest. The equation generation stage is a rule-based deductive learner that combines the result of sentence type classification (and sign prediction for the J-S problems) to derive the numerical quantities needed for the equation. We use the scikit implementation of Random Forest (Pedregosa et al., 2011).

### 3.4 Evaluation

We evaluate the performance of the classifiers on problem type classification, sentence type, sign prediction and overall solution generation by the level of accuracy (how exact the classification is) calculated using 5-fold cross validation. In addition to evaluating a classifier's performance on each task, we also evaluate the contribution of each feature class to the classification by noting the accuracy of the classifier when that feature class is excluded.

## 4 Experiment

We first consider the preprocessing steps and the features considered before delving into the models by type of mathematical word problem being solved.

### 4.1 Preprocessing

We employed Python NLTK (Loper and Bird, 2002) to segment the problems into sentences, perform tokenization, convert words into lower case, tag the words with their Penn treebank part-of-speech tags and lemmatize all the verbs and nouns. We also obtain the dependency parse of the sentences using the Stanford parser (De Marneffe et al., 2006).

### 4.2 Features

We use four classes of features that we describe below.

**Problem-level features:**

- The features in this class are length-related and document-related. The length of the problem in number of sentences is a feature that we consider at the problem level, noticing that on an average, J-S problems tend to have more sentences per problem than those of the C type, which in turn have more sentences than those of the PPW type (refer Table 1).
- Structure that is specific for problem of type C which is the binary valued feature indicating the presence of *comparative adjective and "than"*.
- Keywords (with binary values) extracted using tf-idf constitute another type of problem-level features. To avoid overfitting, we consider only those keywords that occur at least five times in the corpus of problems. We exclude verbs and prepositions from this list. The intuition here is that keywords such as *altogether* characterize PPW problems.

**Sentence-level features:** Mainly used for sentence-level classification into types, the features in this class are positional, structural or semantic.

- Sentence position in the problem tends to be an indicator of the sentence type for PPW and JS problems. For instance, a majority of the JS sentences have the first sentence of the type *Given*, as a manner of discourse structure.
- Structural features essentially capture shared relationships between entities in a sentence, such as that between the subject and object in a sentence obtained in the form of dependency relations. Other structural features are verb phrase (binary valued) such as *to start with*, comparative structure such as *more than* (binary valued) and prepositions such as *on* (binary valued).

**Action-related features:** We observe that problems of the J-S type are characterized by significant action language that describe changes in the possession or condition of objects. Thus, we posit that the count of unique verb lemmas will serve as a discriminating feature. Consider for instance a J-S problem, *Grandma had 5 strawberries. Grandpa gave her 8 more strawberries. How many strawberries does Grandma have now?* The verb from the *Given* sentence *Grandma had 5 strawberries* has changed in the *Change* sentence *Grandpa gave her 8 more strawberries* and thus the problem has 2 verb lemmas (*have* and *give*).

**Entity-related features:** An example of this feature is the number of unique noun phrases. Since problems of type PPW involve static descriptions of two or more disjoint subsets in the *Part* sentence and the union of those sets (or the super category of the entities in the *Part* sentence) in the *While* sentence, a characteristic of problems of this type is the variety of noun phrases. For instance, *Jarron has 5 red triangles and 10 blue squares. How many shapes does he have altogether?* The first sentence which corresponds to *Part* sentence contains two noun phrases: *red triangles* and *blue squares*. The other sentences is *whole sentence.* It has only one noun which is *shapes*. Here red triangles and blue squares are subcategories of shapes and so the number of unique noun phrases is 3.

### 4.3 Parameter tuning

The hyperparameters of the Random Forest classifier were tuned as follows. The corpus of problem types and sentence types were split into a training and test set via a random 80-20 split. The parameters of the random forest classifiers at the problem type, sentence type and sign prediction stages were independently tuned by 5-fold cross validation on the training data set choosing the set that achieves the highest cross-validation accuracy.

As a result, with $n$ as the number of total available features the problem type prediction classifier was set to have a maximum of $\sqrt{n}$ features and allowed to reach a maximum depth of 15 nodes. The sentence type classifier for J-S was set to have a maximum of $n$ features and allowed to reach a depth of 25 nodes, whereas that for PPW had the parameters set to $n$ and 10 respectively. The corresponding parameters for sign prediction module were $log_2 n$ and 50.

## 5 Experimental Results

We report results of using the inductive classification in the first few stages followed by the results of the deductive classification in the equation generation stage.

### 5.1 Problem type classification

The majority baseline is the proportion of the largest problem class in the corpus which is about 44% We observe that problem type classification using Random Forest yielded an accuracy of 93.47% The performance of Random Forest is justified considering that many of our features are correlated. Additionally, our data falls in the realm of the 'small n, large p' scenario where Random Forest is known to perform best. We thus use only Random Forest for classification in the following stages.

### 5.2 Sentence-type classification

For sentence type classification, the baseline is the majority class among sentence types since the sentences are classified independently. Thus, the baseline for J-S problems is 36.12% (majority class is *Change* sentence) and for PPW is 62.47% (majority class is *Part* sentence).

| JS | | PPW | |
|---|---|---|---|
| Baseline | Classifier | Baseline | Classifier |
| 36.12% | 91.55% | 62.47% | 92.32% |

Table 2: Performance of the Random Forest classifier for sentence type classification. The improvement over the baseline is significant.

From Table 2 we notice that the ensemble classifier outperforms the baseline by a wide margin in both J-S and PPW solvers. The performance of the classifier on sentence type prediction for both types seems comparable even though one involves a 3-way classification (for J-S) and the other only two-way (for PPW).

For sign prediction, we note that the module is used only to solve problems of type J-S. Hence, the baseline is the majority class which in our case is 50% owing to the equal number of addition and subtraction problems. The accuracy of the classifier that performs sign prediction is 84.33%. This renders the sign-prediction stage a bottleneck for solving J-S problems.

| **JS** | **PPW** | **C** | Overall |
|---|---|---|---|
| 78.67% | 87.33% | 94.92% | 85.64% |

Table 3: Comparison of the accuracy of the solvers for each problem type.

### 5.3 Overall Solution

The overall solution is obtained by combining the result of the individual stages as per problem type to generate the corresponding equation. The accuracies of the solvers for each problem type are compared in Table 3.

We prepare a simple rule-based baseline with which we compare the results of the equation generation. First, if there is more than one numerical quantity in a sentence, they are all summed up. Any sentence without a numerical quantity is ignored and the question sentence is mapped to the variable. Second, if the number in the first sentence is larger than the number in the second sentence, the first number will be subtracted by the second number; otherwise the two numbers are added. With these two rules, we disregard the type of MWP and generate the equation. The baseline accuracy becomes 59.58% (J-S accuracy is 48%, C accuracy is 55.69%, and PPW accuracy is 87.5%). We would like to point out that a plausible reason that the baseline for PPW is higher than that of the stage-wise approach is because PPW problems' structure coincides with our rules.

This baseline is to be interpreted with some caution, however. Recalling that the purpose of the study is to guide the learner through the stages leading to generating the equation, a comparison of the results of the equation generation stage with the baseline alone is misguided. The final accuracy for solving problems of type Join-Separate is 78.67%. For problems of the PPW type, the accuracy of problem solution after the equation generation stage is 87.33% and that for the class of Compare problems is 94.92%. Based on this we remark that for the automatic solver, problems of the J-S type are the hardest to solve, and those of the Compare type are the easiest. This is justified here by noting that the sign-prediction module is a bottleneck for the J-S solver, as well as an additional classification stage compared to the other problem types.

Pooling the results of each problem type together, we arrive at the overall accuracy of our solver to be 85.64%.

### 5.4 Comparison with the state-of-the-art

A general purpose MWP solver is available via the publicly available WolframAlpha engine. The details of its implementation were unavailable, but we believe it to be operational from its associated blog post that elaborates its functionality and the diagrammatic solution feature of this solver) (Barendse, 2012). We compare the accuracy of our solver with that of the solver provided by WolframAlpha[1] in the absence of other published MWP solvers for arithmetic problems that we study. Since the details of the solution process employed by WolframAlpha are not available we are only able to compare the respective performances at the level of equation generation.

For the purpose of this comparison, we choose the test set (20% of our corpus) compare the accuracy of solutions produced by the solvers. While our MWP solver had an accuracy of 86% on the sample, the performance of Wolfram Alpha is remarkably poor. In particular, barely 9% of the problems were answered correctly, of which about 4% had an incorrect diagram associated with the solution. The vast majority of the MWPs are not solved and the results come back with the error "WolframAlpha doesn't understand your query". Surprisingly, the Wolfram Alpha system performed quite poorly on our dataset. Without the details of the WolframAlpha approach, we are unable to point to the advantages of our approach over that of the state-of-the-art.

### 5.5 Ablation Analysis

| Task | Accuracy | Problem level | Sentence level | Action related | Entity related |
|---|---|---|---|---|---|
| Prob. type | 93.47 | 77.29 | 92.17 | 75.10 | 81.26 |
| Sign | 84.33 | 61.33 | 60.33 | 61.67 | 65.33 |
| JS sent | 91.55 | 89.48 | 54.93 | 87.02 | 90.63 |
| PPW sent | 92.32 | 81.82 | 68.05 | 90.68 | 92.02 |

Table 4: Comparison of the accuracy results (in %) with different feature classes ablated for each classification task with the accuracy where no features were excluded.

Table 4 summarizes the results of the ablation study

[1] www.wolframalpha.com visited on June 01, 2014.

conducted for each task by removing each class of features. For problem type prediction, the action-related features constitute the most important set of features (most likely influenced by the predominance of J-S problems) followed by the problem-level features. The sentence level features seem to have little impact on the overall accuracy. Even though the entity-related features do not have an effect on PPW sentence type classification, it contributes substantially to question type classification (most likely by way of characterizing PPW). Sign prediction depends primarily on the sentence-level features but is about equally dependent on the other sets of features.

## 6 Discussion

The higher the accuracy of classification is, the better the outcome in generating equations will be. In this section, we consider some of the issues that negatively impact the classification process. The first issue involves the preprocessing steps that a MWP has to go through before passing through our analysis. This happens when the problem relates to time, money, and distance and needs quantity conversions before the arithmetic calculations (e.g. *Josie has 7 pennies and 5 nickels. How much money does she have?*). Another obvious class is when the problem requires world knowledge for its solution (e.g. *Today is October 25th. How many days are there until Halloween?*). The other case where our program fails is when a question has a complex sentence structure. e.g *How many Yodas flew away from the planet in the space shuttle if 23 Yodas stayed on the planet of 30 Yodas in all?*

Focusing on the errors of the J-S problem solver, the majority of errors result from incorrect sign prediction, explained by the fact that this module is the bottleneck in our J-S automatic solver. The overall accuracy is slightly higher than we expect because the error from sign prediction and sentence type classification overlap. It is also the case that even though the classifier misclassifies *Change* and *Given* sentences, if the sign is correctly assigned as '+', the final equation is still correct i.e. $3 + x = 4$ is the same as $x + 3 = 4$. Finally, the main source of error for problems of PPW type is that the problem type classifier misclassifies PPW to be JS, which leads to an incorrect solution. JS and PPW are very similar but they focus on different aspects. JS focuses on the dynamic action, while PPW captures the relationship between nouns in each sentence.

For problems of the Compare type, there are two sources of error. First, the rule-based classifier itself provides 94.92% because some questions need quantity conversion before being processed. For example, *Joel started the paper route at 7:05. He worked for 25 minutes. When did he finish?* The other is that the comparison problem is misclassified as J-S or PPW at the problem type classification stage. Accounting for these errors would entail working with better classifiers that handle inter-sentence semantics.

To get a feel for the model's generalizability we tested on a set of problems not of the CGI type from Dadsworksheets.com[2]. On this set of 400 addition and subtraction word problems our model yielded an overall accuracy of 87%, suggesting that our method is not restricted to solving problems of the CGI type alone. Looking ahead, we are working to solve more complicated MWPs of upper elementary grades.

It is conceivable that a multi-stage approach such as the one considered here can constitute one of the key design factors in applications involving intelligent tutoring systems for elementary mathematics education. The goal of guiding the learner to understand the steps involved in solving the problem can be met via our approach of identifying the problem types, highlighting the discourse elements (sentence types) while simultaneously helping arrive at the answer.

## 7 Conclusion

We present a multi-stage text-classification approach to solve arithmetic problems of elementary level automatically. Our approach recognizes the problem type, identifies the discourse structure and generates the corresponding equation to eventually solve the problem. This is in line with results from cognitive psychology studies in children learning to solve MWPs. With accuracies substantially higher than the baseline, we also observe that the performance gains of our solver compared with the state-of-the-art MWP solvers such as WolframAlpha are also substantial.

## References

Peter Barendse. 2012. Solving word problems with wolfram—alpha@ONLINE, October. http://blog.wolframalpha.com/2012/10/04/solving-word-problems-with-wolframalpha/.

Daniel G. Bobrow. 1964. A question-answering system for high school algebra word problems. In *Proceedings of the October 27-29, 1964, Fall Joint Computer Conference, Part I*, AFIPS '64 (Fall, part I), pages 591–614, New York, NY, USA. ACM.

Thomas P Carpenter, Elizabeth Fennema, Megan Loef Franke, Linda Levi, and Susan B Empson. 2000. Cognitively guided instruction: A research-based teacher professional development program for elementary school mathematics. research report.

[2]http://www.dadsworksheets.com/ accessed on March 20th, 2013

Suleyman Cetintas, Luo Si, Yan Ping Xin, Dake Zhang, and Joo Young Park. 2009. Automatic text categorization of mathematical word problems. In *FLAIRS Conference*.

Suleyman Cetintas, Luo Si, Yan Ping Xin, Dake Zhang, Joo Young Park, and Ron Tzur. 2010. A joint probabilistic classification model of relevant and irrelevant sentences in mathematical word problems. *JEDM-Journal of Educational Data Mining*, 2(1):83–101.

Arun Chaganty, Akash Lal, Aditya V Nori, and Sriram K Rajamani. 2013. Combining relational learning with smt solvers using cegar. In *Computer Aided Verification*, pages 447–462. Springer.

Denise Dellarosa Cummins. 1991. Children's interpretations of arithmetic word problems. *Cognition and Instruction*, 8(3):pp. 261–289.

Erik De Corte and Lieven Verschaffel. 1987. The effect of semantic structure on first graders' strategies for solving addition and subtraction word problems. *Journal for Research in Mathematics Education*, pages 363–381.

Marie-Catherine De Marneffe, Bill MacCartney, Christopher D Manning, et al. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of LREC*, volume 6, pages 449–454.

Denise Dellarosa. 1985. Solution: A computer simulation of childrens recall of arithmetic word problem solving. *University of Colorado Technical Report*, pages 85–148.

Denise Dellarosa. 1986. A computer simulation of children's arithmetic word-problem solving. *Behavior Research Methods*, 18:147–154, March.

Kerianne Ebner. 2011. Cognitively guided instruction (cgi) problem types @ONLINE, July.

Charles R. Fletcher. 1985. Understanding and solving arithmetic word problems: A computer simulation. *Behavior Research Methods*, 17(5):565–571.

Yufan Guo, Anna Korhonen, Maria Liakata, Ilona Silins Karolinska, Lin Sun, and Ulla Stenius. 2010. Identifying the information structure of scientific abstracts: An investigation of three different schemes. In *Proceedings of the 2010 Workshop on Biomedical Natural Language Processing*, BioNLP '10, pages 99–107, Stroudsburg, PA, USA. 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.

Mark D LeBlanc and Sylvia Weber-Russell. 1996. Text integration and mathematical connections: a computer model of arithmetic word problem solving. *Cognitive Science*, 20(3):357–407.

Christian Liguda and Thies Pfeiffer. 2011. A question answer system for math word problems. First International Workshop on Algorithmic Intelligence.

Christian Liguda and Thies Pfeiffer. 2012. Modeling math word problems with augmented semantic networks. In Gosse Bouma, Ashwin Ittoo, Elisabeth Mtais, and Hans Wortmann, editors, *Natural Language Processing and Information Systems*, volume 7337 of *Lecture Notes in Computer Science*, pages 247–252. Springer Berlin Heidelberg.

Lingyi Liu, Chen-Hsuan Lin, and Shobha Vasudevan. 2012. Word level feature discovery to enhance quality of assertion mining. In *ICCAD*, pages 210–217.

Edward Loper and Steven Bird. 2002. Nltk: The natural language toolkit. In *Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics - Volume 1*, ETMTNLP '02, pages 63–70, Stroudsburg, PA, USA. Association for Computational Linguistics.

Takuya Matsuzaki, Hidenao Iwane, Hirokazu Anai, and Noriko Arai. 2013. The complexity of math problems – linguistic, or computational? In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 73–81, Nagoya, Japan, October. Asian Federation of Natural Language Processing.

Travis E. Oliphant, 2006. *Guide to NumPy*. Provo, UT, March.

Shawn Olson, Natalie Musser, Sue McAdaragh, Roxane Dyk, Jonath Weber, Tracy Mittleider, Lucy Atwood, and Marcia Torgrude. 2008. South dakota counts: Cgi problems created by south dakota math teacher leaders @ONLINE, January.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Wanintorn Supap, Kanlaya Naruedomkul, and Nick Cercone. 2013. Mathmaster: an alternative math word problems translation. *Computational Approaches to Assistive Technologies for People with Disabilities*, 253:109.

Lieven Verschaffel, Brian Greer, and Erik De Corte. 2000. *Making sense of word problems*. Lisse.

Yiming Yang and Xin Liu. 1999. A re-examination of text categorization methods. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 42–49. ACM.