AUTOMATIC SOLVER FOR MATHEMATICAL WORD PROBLEMS

BY

BUSSABA AMNUEYPORNSAKUL

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2014

Urbana, Illinois

Adviser:

Professor Pramod Viswanath

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 thesis we design and construct an MWP solver in a systematic manner, as a step toward enabling comprehension in mathematics. 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 toward 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 the elementary grade mathematics curriculum.

To my parents, for their love and support

ACKNOWLEDGMENTS

I am grateful to my adviser, Professor Pramod Viswanath, and Dr. Suma Bhat for all their guidance and support. I also thank my parents for providing me an invaluable opportunity to study at the University of Illinois.

TABLE OF CONTENTS

CHAPTER 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. Unlike a top-down approach of having a template for various problem types and generating a solution for each problem template, 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.

Personalized learning systems today aim at making learning more enjoyable for learners of all ages. While there are many systems for language learning (e.g. [1] for learning to read), few systems available today are geared toward enabling comprehension in mathematics and solving MWPs [2]. The design of personalized learning systems for solving mathematical word problems will be more effective by having a module that identifies the semantic structure of the word problem. This will enable the task of sentence re-ordering thereby aiding the identification of the givens and unknowns in the problem. Few prior studies are available that address the task of solving word problems automatically [3, 4, 5, 6, 7]. There is no study, however, that identifies the semantic structure of a word problem. In this thesis we address this need by: (a) identifying the semantic structure of MWPs that will enable comprehension in mathematics, and (b) utilizing the semantic structure as a source of information toward developing automatic solvers of MWP in a systematic manner.

Table 1.1: Sample problems of each type considered in this thesis.

J-S	Grandma had 5 strawberries. Grandpa gave her 8 more strawberries. How many strawberries does Grandma have now? Equation: $5 + 8 = x$
PPW	Nicole has 6 red buttons and 6 green buttons. How many buttons does Nicole have? Equation: $6 + 6 = x$
C	Angela has 6 mittens. Jordan has 4 more mittens than Angela. How many mittens does Jordan have? Equation: $4 + 6 = x$

In an MWP, significant background information is presented in text format. Table 1.1 shows examples of the MWPs considered in this thesis: J-S stands for Join-Separate, PPW represents Part-Part-Whole, and C is from Compare. The task is to generate the equation corresponding to the problem and solve it. 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 [8, 9, 10], which is not the same as reading comprehension on passages.

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 [11] 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. It is conceivable that such a multi-stage approach can constitute one of the key design factors in applications involving intelligent tutoring systems for mathematics education.

A text classification perspective to MWP solution 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 [12]. 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 work as we explore the usefulness of a text classification approach to solving MWPs. In addition, our thesis is novel in adopting the multistage approach to solving word problems automatically.

Specifically, this thesis makes the following contributions.

1. Taking a text classification approach toward 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 [13] are also substantial.
2. We demonstrate the efficacy of our multistage approach implemented as a deductive learner driven by an inductive engine by building a 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. Supported by psychology studies, we see that this could be turned around for learning technology development.

CHAPTER 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 [14], SOLUTION [15], and ARITHPRO [16] and [3] are representations of cognitive models of human mathematical word problem solving processes. With the exception of [3], these operate on propositional representations of the problem text later solved in a rule-based manner. Notable among these was the approach in [3] that solves arithmetic word problems by parsing the meaning on a sentence by sentence basis and constructing corresponding representations for eventual problem solving. Evidently, the natural language processing capabilities in these studies were limited to mostly manual input mechanisms.

In the realm of intelligent tutoring systems automatic MWP solvers were based on either using specific sentence structures and keywords [5], or using templates (schema) limited in scope by variety and problem types as in [4] for grade-level problems in Thai and [6, 7] for grade-level problems in German.

An early approach to automatic classification of MWP using natural language processing methods was [12]. The study pointed out that certain problem types (such as the multiplicative compare and equal group) were characterized by their lexical content. Their empirical results showed that a blind text categorization approach via stop word removal and stemming failed to help the classification task for those problem types. In addition, they identify the role of including discriminative part of speech tags for problem type classification. Another related study [17], 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 [18], which aims at understanding the complexity of MWPs encountered by students appearing for a Japanese university entrance examination. It includes an 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, and the goal was to arrive at a solution automatically without paying attention to the intermediate stages of problem solving.

While preparing this thesis, we noticed a paper in the conference ACL 2014 on the topic of learning to solve algebra word problems [19]. It is the most recent study, which attempts to solve word problems with a system of linear equations automatically. They used a template to fit in the information that they harnessed from a given problem. The problems that they considered are different from our scope. While they studied the way to solve the system of linear equations containing only addition, we consider the problem of determining an equation from a one-variable MWP with addition and subtraction.

Taking a view different from that of prior studies, our focus here is twofold: first, inspired by the approach to identify the structure of scientific abstracts in [20], we would like to gain a fundamental understanding of the discourse structure of an MWP which serves as the information-bearing component of the problem; 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 [4] 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 [21, 22]. The cognitive modeling perspective to solving MWP in children renders the inductive-deductive learner combination a natural choice for our study.

CHAPTER 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 [23]. 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 (referring to the relation between sets in the problem statement)[3]. (The problem types we consider correspond to the change, compare and combine classes respectively, studied in prior works on cognitive psychology and mathematics education.)

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*.

Figure 3.1: An example of J-S question structure.

3.1.1 Join and Separate (J-S)

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* is as Figure 3.1

3.1.2 Part-Part-Whole (PPW)

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. The example represents in Figure 3.2

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*.

Figure 3.2: An example of PPW question structure.

3.1.3 Comparison

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

3.1.4 Corpus Statistics

In a given problem, the missing quantity could be in the *Given*, *Change* or *Result* sentence (likewise in the *part* or the *whole* sentence). The dataset used in our study is a set of sample problems from the South Dakota Counts [24] and teacherweb.com [25]. A brief description of the problems of each type and their characteristics in the corpus is summarized in Table 3.1, Figure 3.3 and Figure 3.4.[1]

Table 3.1: Corpus description of the set of problems studied.

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 (mean)	3.42	2.72	3.06
No. of verb types (total)	99	36	46

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

[1]The set of word problems studied is available for research purposes at http://anonymized.willbemade.available

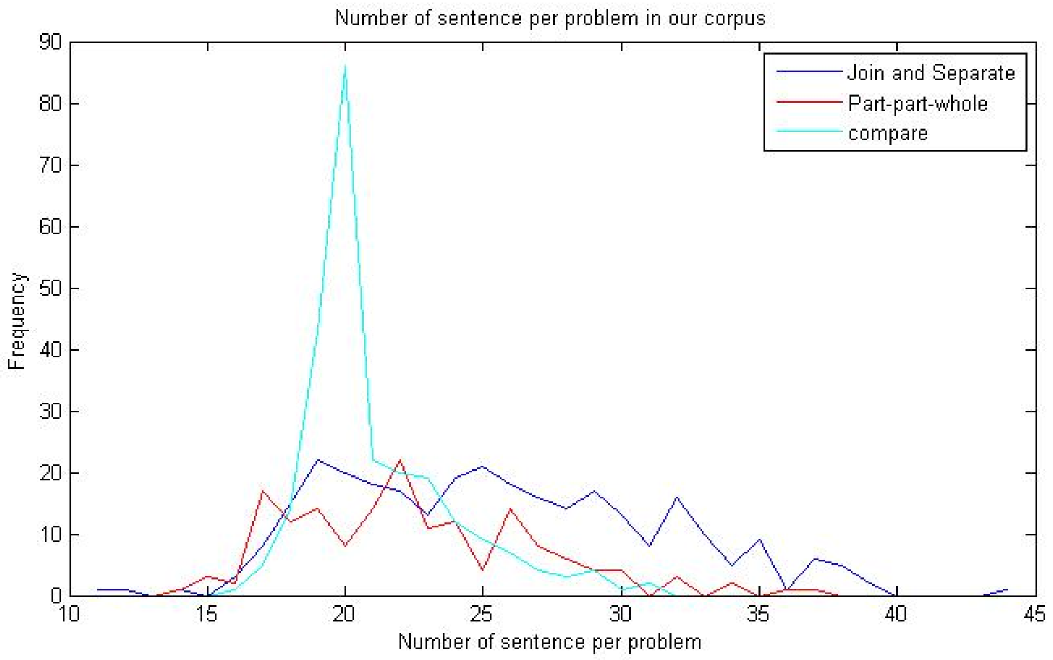


Figure 3.3: Number of word per problem.

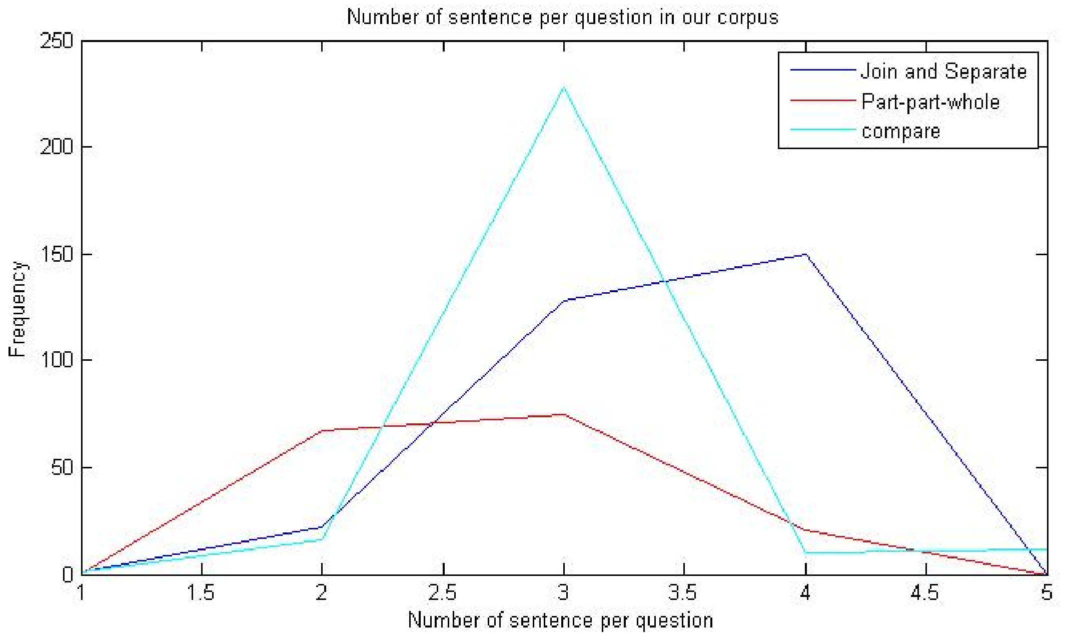


Figure 3.4: Number of sentence per problem.

separate) by the researchers. The annotators agreed on 99.4% of the sentence function types.

Notice from Table 3.1 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 three 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 Model

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 3.5.

3.2.1 Join and Separate

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. In its simplest form, the *Given*, *Change* and *Result* sentences are separate as in *Grandma had 5 strawberries. Grandpa gave her 8 more strawberries. How many strawberries does Grandma have now?* or joined as in *Grandma had 5 strawberries and Grandpa gave her 8 more strawberries. How many strawberries does Grandma have now?* The same problem can be rephrased as a complex sentence in *How many strawberries would Grandma have now if Grandma had 5*

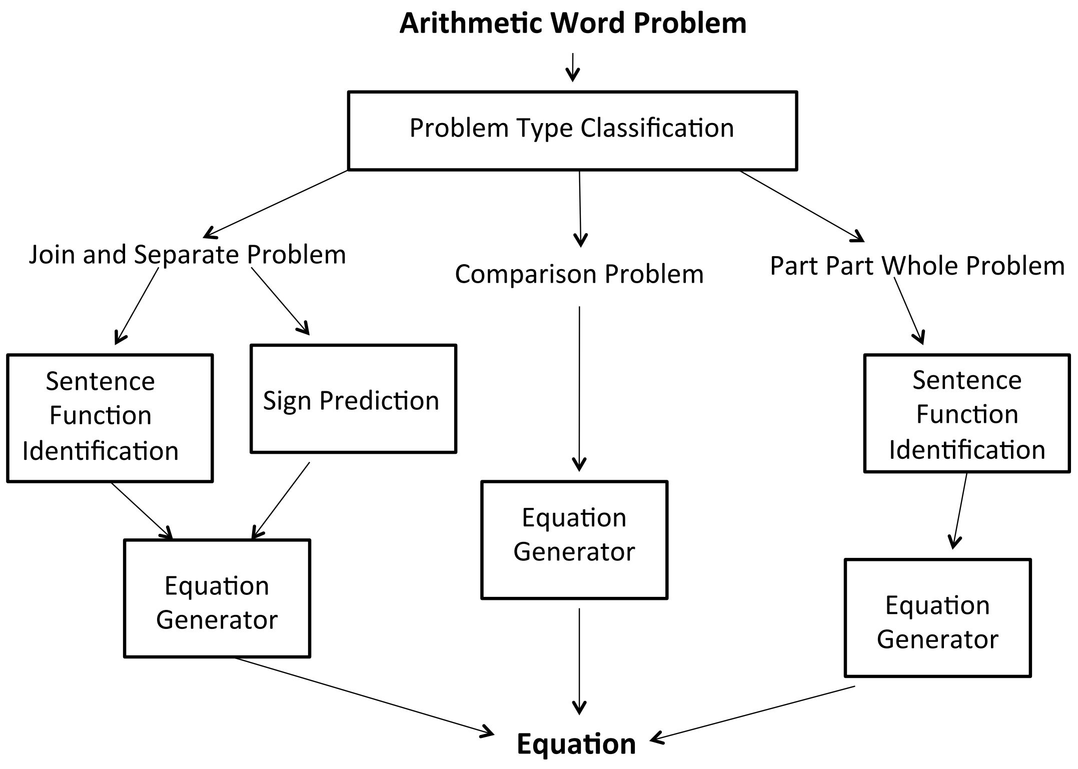


Figure 3.5: Flowchart for the system.

strawberries and Grandpa gave her 8 more? In addition to having a versatile structure, the *Change* sentence could be constructed to modify the *Given* sentence via various arithmetic operations.

Figure 3.6 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.

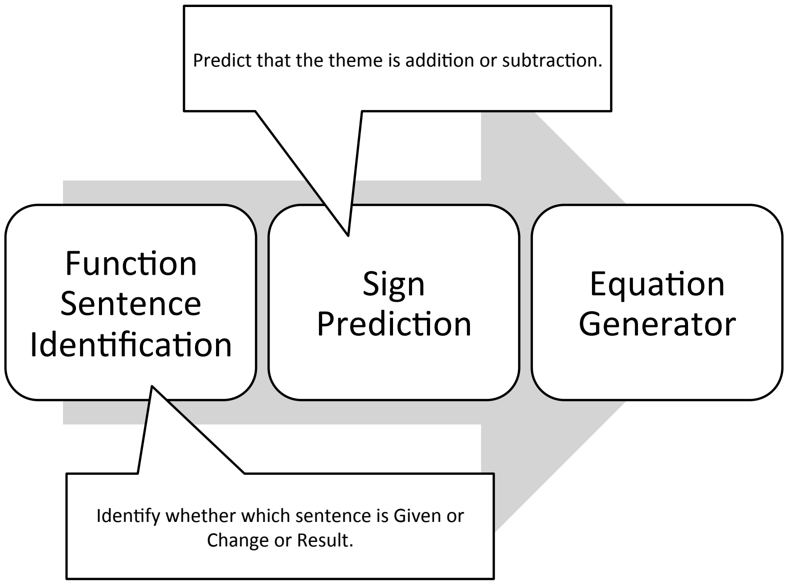

Predict that the theme is addition or subtraction.
Function Sentence Identification
Sign Prediction
Equation Generator
Identify whether which sentence is Given or Change or Result.

Figure 3.6: Flowchart for J-S problem.

3.2.2 Part-Part-Whole

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 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 3.7.

3.2.3 Comparison

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.

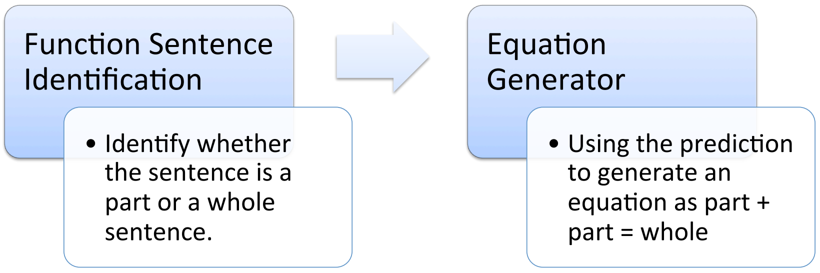

Function Sentence Identification
• Identify whether the sentence is a part or a whole sentence.
Equation Generator
• Using the prediction to generate an equation as part + part = whole

Figure 3.7: Flowchart for PPW problem.

3.3 Implementation

For the tasks of problem type classification, sentence type classification and sign prediction, we use three types of inductive classifiers: generative (Naive Bayes), discriminative (LIBSVM and Logistic Regression), and ensemble method (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 [26], WEKA implementation of Naive Bayes and logistic regression [27] and the LibSVM implementation for SVM [28].

3.3.1 Naive Bayes

Naive Bayes is one of the learning classifiers based on Bayes rule. The classifier approximates an unknown target function $f : X \rightarrow Y$, or $P(Y|X)$, where X is a feature vector and Y is a target label. In short, we would like to select Y such that it maximizes

$P(Y|X)$.

Applying Bayes' rule,

$$P(Y = y_i|X = x_k) = \frac{P(X = x_k|Y = y_i)P(Y = y_i)}{\sum_j P(X = x_k|Y = y_j)P(Y = y_j)}. \tag{3.1}$$

However, unbiased learning of Bayes classifiers is impractical because finding $P(X|Y)$ and P(Y) depend on training data.

We assumes that Y and all X_i are Boolean variables. For any particular y_i, we need to compute $2^n - 1$ independent parameters. That means we must estimate $2(2^n - 1)$ parameters. In order to obtian reliable estimator, we need to observe each parameter several times. Apparently, this method is unrealistic to compute.

To simplify the Bayes classifier, Naive Bayes assumes that the attributes $X_1, .X_2, , X_n$ are all conditional independent of one another, given Y.

$$\begin{aligned} P(X|Y) &= P(X_1, X_2|Y) \\ &= P(X_1|X_2, Y)P(X_2|Y) \\ &= P(X_1|Y)P(X_2|Y). \end{aligned}$$

Therefore, $P(X_1, X_2, , X_n|Y) = \prod_{i=1}^{n} P(X_i|Y)$. With the same assumption on X and Y, the complexity reduces from $2(2^n - 1)$ to $2n$. In short, we want to find Y such that it maximizes $P(X = x_k|Y = y_i)P(Y = y_i)$.

The advantage of this algorithm is simplicity. However, conditional independence assumption that simplifies the problem is not always true for every classification. In other word, Naive Bayes might oversimplify the task which leads to an incorrect classification result [29].

3.3.2 Logistic Regression

Logistic Regression attempts to learn a target function $f : X \to Y$ or $P(Y|X)$ where Y is discrete-valued, and $X =< X_1, X_2, ..., X_n >$ is any attributed feature. Unlike Naive Bayes, logistic regression uses training data to directly estimate $P(Y|X)$. Logistic Regression is often referred to as *discriminative* classifier because we obtain distribution $P(Y|X)$ directly from training data. The parametric model assumed by Logistic Regression in the case when Y is Boolean is:

$$P(Y=1|X) = \frac{1}{1+\exp\left(w_o + \sum_{i=1}^{n} w_i X_i\right)} \tag{3.2}$$

and

$$P(Y=0|X) = \frac{\exp(w_o + \sum_{i=1}^{n} w_i X_i)}{1+\exp(w_o + \sum_{i=1}^{n} w_i X_i)}. \tag{3.3}$$

By maximum likelihood ratio estimator, we will declare Y = 0 if
$P(Y=0|X) > P(Y=1|X)$. Applying equations 3.2 and 3.3, we will declare Y = 0 if $\exp(w_o + \sum_{i=1}^{n} w_i X_i) > 1$ or $w_o + \sum_{i=1}^{n} w_i X_i > 0$.

In addition, the parametric form of $P(Y|X)$ used by Logistic Regression is implied by the assumption of Gaussian Naive Bayes classifier. That means both of them are closely related.

Although Logistic Regression is consistent with the Naive Bayes assumption that the input features X_i are conditionally independent given Y, it is not rigidly tied to this assumption as is Naive Bayes. Given data that disobeys this assumption, the conditional likelihood maximization will adjust its parameters to maximize the fit to the conditional likelihood of the data, even if the result are inconsistent with the Naive Bayes parameter estimates. On the other hand, logistic regression consumes a significant longer time because it tries to find $P(X|Y)$ directly [29].

3.3.3 SVM

Given a training set of instance-label parameters: $(x_i, y_i), i = 1, ..., l$ where $x_i \in \Re^n$ and $y \in -1, 1^l$, the support vector machines (SVM) requires the solution of the following optimization problem:

$$\min_{w,b,\xi} \frac{1}{2} w^T w + C \sum_{i=1}^{l} \xi_i \tag{3.4}$$

$$\text{subject to } y_i(w^T \phi(x_i) + b) \geq 1 - \xi_i \tag{3.5}$$

$$\xi_i \geq 0. \tag{3.6}$$

Training vectors x_i are mapped to a higher-dimensional space by the ϕ function, which is a function that satisfies $K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$ called a kernel function. SVM finds a linear separating hyperplane with the maximum margin in the higher-dimensional space.

The following are basic kernels:

- linear: $K(x_i, x_j) = x_i^T x_j$
- polynimial: $K(x_i, x_j) = (\gamma x_i^T x_j + r)^d, \gamma > 0$
- radial basis function (RBF): $K(x_i, x_j) = \exp(-\gamma ||x_i - x_j||^2), \gamma > 0$
- sigmoid: $K(x_i, x_j) = \tanh(\gamma x_i^T x_j + r)$

Here, γ, r, and d are kernel parameters.

In our experiment, we used the default setting of SVM provided in WEKA. The SVM that we adopted was 1-SVC radial basis function kernel with degree 3 and $\gamma = \frac{1}{k}$.

In order to get an accurate classification result from SVM, some preprocessing step on the data is required. All of the attributes should be represented in real numbers. Also, scaling needs to be applied. Otherwise, the attributes with greater numerical ranges will dominate those in smaller ranges.

If the number of features is large, one may not need to map data to a higher-dimensional space. The nonlinear mapping does not improve the performance. That means, LIBLINEAR (using a linear kernel) is good enough for accurate classification. However, the LIBSVM is at least as good as LIBLINEAR. In particular case, both numbers of instances and features are large. The accuracies between LIBSVM and LIBLINEAR are approximately the same, but speed of LIBLINEAR is significant faster [28].

3.3.4 Random Forest

Random Forest is the collection of individual random trees. A single classification tree is a decision tool that uses a tree-like graph consisted of decisions and their possible consequences. Each tree gives a classification and we say that the tree *votes* for that class. The forest chooses the classification having the most votes. The algorithm is depicted in Figure 3.8.

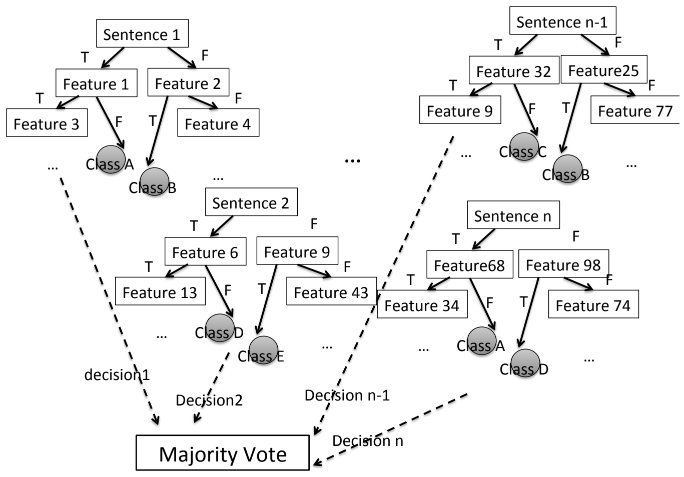


Figure 3.8: Random Forest algorithm.

The Random Forest error rate depends on two things:

- The correlation between any two trees in the forest. Increasing the correlation increases the error rate.
- The strength of each individual tree in the forest. A tree with a low error rate is a strong classifier. Increasing the strength of the individual trees decreases the error rate.

Reducing the number of features used in each tree reduces both correlation and strength. Increasing it increase both items. Ideally, we want to have low correlation, but high strength.

Unlike a regular decision tree, Random Forest does not overfit. It runs fast and efficiently on large data bases. Another advantage of Random Forest over other classifiers is that it can rank the importance of features. That is, we do not have to hand pick a feature such that it will provide benefit in classifing. The Random Forest will exploit all features by itself [30].

3.4 Evaluation

3.4.1 Testing Stage

In order to choose which classifier works best in each sceanario, we employed WEKA to perform the experiment due to the simplicity of the WEKA interface. We used WEKA with the following classifiers: Naive Bayes, Logistic Regression, LIBSVM, and Random Forest. The accuracy were measures from 10-fold cross validation. That is, the data was randomly devided into 10 parts. Then, 9 parts of the data was trained and the other section was used for testing. The results from classifiers consisted of the following:

1. **Accuracy** is the precision of the classification.
2. **Precision** or positive predictive value is the fraction of retrieved documents that are relevant to the prediction.
3. **Recall** or sensitivity is the fraction of the documents that are relevant to the query that are successfully retrieved.
4. **F-measure** is a meansure of a test's accuracy. The quantity is calculated from precision and recall values as following:

$$\mathrm{F}-measure = 2\frac{PR}{P+R}. \tag{3.7}$$

5. **Confusion Matrix** or contingency table is a type of tables that allows visualization of the performance of an algorithm. Each column of a matrix represents the instances in a predicted class, while each row represents the instances in an actual class.

3.4.2 System Stage

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 POS tagger in the NLTK toolkit [31] 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 Chapter 6), a solution is not generated. The generated equation can be solved using Numpy [32].

CHAPTER 4

EXPERIMENTS

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 Stage

We employed the tokenizer and the part-of-speech (POS) tagger from Python NLTK [31] to segment the problems into sentences, perform tokenization and tag the words with their Penn treebank POS tags. We also convert words into lower case and lemmatize all the verbs and nouns using NLTK. We also obtain dependency parse of the sentences using the Stanford parser [33].

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 to Table 3.1).
- Structure that is specific for particular problem type such as the binary value of the comparison structure, which will be in form 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 (binary values such as *more than* and preposition (prepositions) such as *on*.

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*). As a contrastive example, consider a problem of type Compare, *Angela has 6 mittens. Jordan has 4 more mittens more than Angela. How many mittens does Jordan have?* Here the problem involves a static comparison of two sets and so the problem has only one verb *have*.

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. Taking the same example sentence from J-S as before *Grandma had 5 strawberries. Grandpa gave her 8 more strawberries. How many strawberries does Grandma have now?* we notice that the noun phrase *strawberries* in the problem are the same.

4.3 Testing with WEKA Tester

We applied WEKA Tester to explore performances of chosen classifiers for each task so that we could select the best classifier for each stage in our system. We used WEKA tester because it was easy to test the same data on various classifiers. That is what we want to determine in this section. However, we need another preprocessing stage. We had to vectorize all the data in corpus and represented all instances in form of .arff file.

Problem Type Classification

This task is to identify whether the incoming question is J-S, PPW or Comparison. This is the first stage that the question passes. The accuracy is shown in Table 4.1

Table 4.1: Accuracy from WEKA tester for problem type classification.

Classifier	**Accuracy**	**J-S**			**C**			**PPW**		
		P	R	F	P	R	F	P	R	F
Naive Bayes	88.6269	0.94	0.91	0.93	0.90	0.91	0.91	0.78	0.80	0.79
Logistic Regression	90.9847	0.91	0.94	0.93	0.94	0.89	0.92	0.91	0.91	0.91
LIBSVM	64.7712	0.95	0.57	0.71	0.55	0.96	0.7	0.80	0.19	0.3
Random Forest	91.5395	0.95	0.92	0.94	0.92	0.94	0.93	0.86	0.86	0.86

	JS	C	PPW
JS	234	9	14
C	3	274	24
PPW	11	21	131

(a)

	JS	C	PPW
JS	242	5	10
C	17	269	15
PPW	7	11	145

(b)

	JS	C	PPW
JS	146	109	2
C	5	290	6
PPW	3	129	31

(c)

	JS	C	PPW
JS	236	11	10
C	4	283	14
PPW	8	14	141

(d)

Figure 4.1: Confusion matrix of question type classification: (a) Naive Bayes, (b) Logistic Regression, (c) LIBSVM, (d) Random Forest.

From Table 4.1 and Figure 4.1, Random Forest performs the best in all criteria. It is followed by Logistic Regression, Random Forest, and LIBSVM accordingly. Unlike SVM, the performance of Random Forest, Logistic Regression and Naive Bayes are not significantly diffent.

J-S Sign Prediction

This classification is specific to the Join and Separate problem type. It will predict the sign of this problem whether it will be addition or subtraction as show in Table 4.2

Table 4.2: Accuracy from WEKA tester for sign prediction in J-S.

Classifier	**Accuracy**	+			-		
		P	R	F	P	R	F
Naive Bayes	80	0.76	0.91	0.83	0.86	0.68	0.76
Logistic Regression	75.3333	0.77	0.76	0.77	0.73	0.75	0.74
LIBSVM	53	0.53	1	0.69	0	0	0
Random Forest	81.3333	0.81	0.84	0.83	0.82	0.78	0.97

From Table 4.2 and Figure 4.2, Random Forest performs the best. It is followed closedly

	+	-
+	144	15
-	45	96

(a)

	+	-
+	121	38
-	36	105

(b)

	+	-
+	159	0
-	141	0

(c)

	+	-
+	134	25
-	31	110

(d)

Figure 4.2: Confusion matrix of sign prediction in J-S: (a) Naive Bayes, (b) Logistic Regression, (c) LIBSVM, (d) Random Forest.

by Naive Bayes, Logistic Regression and LIBSVM. Observing in Figure 4.2, LIBSVM performs defectively by deciding all questions to be addition. Even though Naive Bayes and Random Forest have approximately the same accuracy, the confusion matrices show that Random Forest is less biased than what the Naive Bayes gives. That is, Random Forest provides a superior performance.

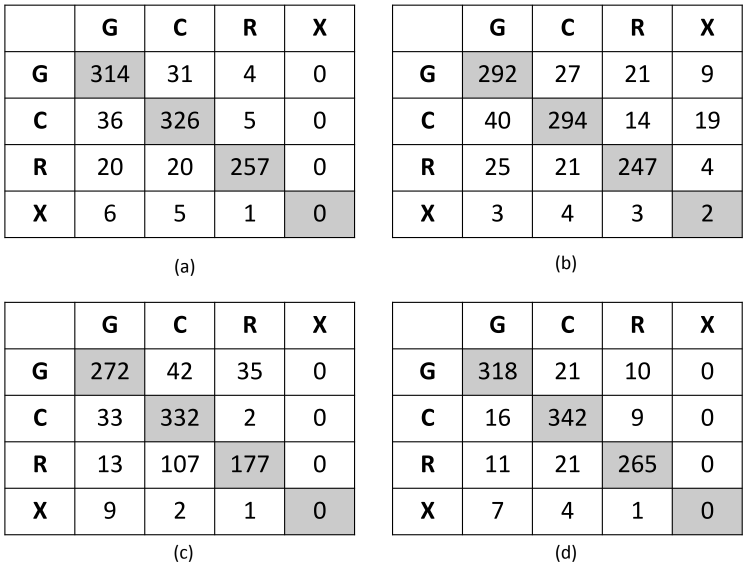
J-S Sentence Function Identification

There are four possible functions in a join and separate problem, which are *given, change, result, and unknown.* This part of the system will identify the function of each sentence in the question.

Table 4.3: Accuracy from WEKA tester for sentence function identification in J-S.

Classifier	**Accuracy**	**G**			**C**			**R**			**X**		
		P	R	F	P	R	F	P	R	F	P	R	F
Naive Bayes	87.5122	0.83	0.9	0.87	0.85	0.89	0.87	0.96	0.86	0.91	0	0	0
Logistic Regression	81.4634	0.81	0.84	0.82	0.85	0.80	0.82	0.87	0.83	0.85	0.06	0.17	0.09
LIBSVM	76.1951	0.83	0.78	0.81	0.69	0.91	0.78	0.82	0.60	0.69	0	0	0
Random Forest	90.2439	0.90	0.91	0.91	0.88	0.93	0.91	0.93	0.89	0.91	0	0	0

According to Table 4.3 and Figure 4.3, the order of classification performance is the



	G	C	R	X
G	314	31	4	0
C	36	326	5	0
R	20	20	257	0
X	6	5	1	0

(a)

	G	C	R	X
G	292	27	21	9
C	40	294	14	19
R	25	21	247	4
X	3	4	3	2

(b)

	G	C	R	X
G	272	42	35	0
C	33	332	2	0
R	13	107	177	0
X	9	2	1	0

(c)

	G	C	R	X
G	318	21	10	0
C	16	342	9	0
R	11	21	265	0
X	7	4	1	0

(d)

Figure 4.3: Confusion matrix of sentence type identification in J-S: (a) Naive Bayes, (b) Logistic Regression, (c) LIBSVM, (d) Random Forest.

same as decribed in Sign prediction. That is, the accuracy of Random Forest > Naive Bayes > Logistic Regression > LIBSVM. For other criteria, they provide the same trendency as accuracy.

PPW Sentence Function Identification

There are three possible types of sentences in a PPW question, which are *part, whole, and unknown*. The stage will identify the function of each sentence in a question.

Table 4.4: Accuracy from WEKA tester for sentence function identification in PPW.

Classifier	**Accuracy**	**P**			**W**			**X**		
		P	R	F	P	R	F	P	R	F
Naive Bayes	69.697	0.67	0.98	0.80	0.90	0.22	0.35	0	0	0
Logistic Regression	86.2471	0.92	0.86	0.89	0.78	0.87	0.82	0	0	0
LIBSVM	62.4709	0.63	1	0.769	0	0	0	0	0	0
Random Forest	92.0746	0.93	0.94	0.94	0.89	0.89	0.89	0	0	0.91

Based on Table 4.4 and Figure 4.4, the order of accuracy is Random Forest, Logistic Regression, Naive Bayes and LIBSVM. The difference between each classifier's accuracy is

	P	W	X
P	264	4	0
W	125	35	0
X	1	0	0

(a)

	P	W	X
P	231	37	0
W	21	139	0
X	0	1	0

(b)

	P	W	X
P	268	0	0
W	160	0	0
X	1	0	0

(c)

	P	W	X
P	252	16	0
W	17	143	0
X	0	1	0

(d)

Figure 4.4: Confusion matrix of sentence type identification in PPW: (a) Naive Bayes, (b) Logistic Regression, (c) LIBSVM, (d) Random Forest.

considerable. That is, Random Forest surpasses other classifiers. On the other hand, s popular LIBSVM is defunct in this classification because it labeled every sentence to be *part* sentence. LIBSVM is biased to the class with majority quantity.

Even though there are several factors in deciding the best classifier to each task, accuracy is the most important factor because at the end we are considered only how precise the equation is. Precision and recall are starting points for understanding relevancy in the classification. In our case, both precision and recall have the same trend as accuracy.

SVM functions inadequately in two out of four tasks by choosing the majority class for any attributes. Logistic regression is not outstanding in any stage of classification. Even though Naive Bayes provides similar accuracy to Random Forest, Naive Bayes has trendency to be more biased to the majority class.

Hence, we decide to use Random Forest as our classifier for every task because Random Forest provides comparable high accuracies in all tasks we considered. Using only Random Forest increases the speed of the end-to-end system because its process is faster compared to other classifiers such as logistic regression and we do not have to train the data every single classification.

4.4 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 fivefold cross validation on the training data set.

We considered three types of maximum features: $\sqrt(n)$, $\log_2(n)$, and n where n is the total number of features we have. Also, we speculated the maximum depth of a tree. By varying the depth from 5 to 50 and unlimited. The best results of each classifier are in Table 4.5.

Table 4.5: Parameters that provide the best accuracy in each classification.

Task	**Max Feature**	**Max Depth**	**Accuracy**
Problem Type	$\sqrt{n}$	15	92.01%
JS Sentence Type	$\log_2 n$	50	93.92%
JS Sign	n	25	89.58%
PPW Sentence Type	n	10	91.79%

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.

CHAPTER 5

EXPERIMENT RESULTS

We use all classifiers mentioned in Chapter 3 for problem type classification. Since the errors from this stage propagate to the subsequent stages, we only consider the best performing in overall classifications: problem type classification, sentence type identification, and sign prediction.

5.1 Problem Type Classification

Table 5.1: Performance comparison of the classifiers for the problem type classification.

Classifier	**RF**	**NB**	**SVM**	**Logistic Regression**
Baseline Accuracy	43.94%			
Classifier Accuracy	91.54%	88.63%	64.77%	90.98%

The majority baseline is the proportion of the largest problem class in the corpus which is about 44% The performance of the three classifiers is shown in Table 5.1. We observe that the ensemble method Random Forest is the best performing classifier. We notice that problem type classification using Random Forest yielded an accuracy of 91.54%, the highest among the classifiers considered. The performance of Random Forest is justified considering that many of our features are correlated (a reason why Naive Bayes fails). Additionally, our data falls in the realm of the “small n, large p” scenario where Random Forest is known to perform best (a plausible reason for SVM’s poor performance). We thus use only Random Forest for classification in the following stages.

Table 5.2: Performance of the Random Forest classifier for sentence type classification after tuning all parameters.

JS		**PPW**	
Baseline	Classifier	Baseline	Classifier
36.12%	91.55%	62.47%	92.32%

Table 5.3: Comparison of the accuracy of the solvers for each problem type.

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

5.2 Sentence Type Classification

For the 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). From Table 5.2 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 three-way classification (for J-S) and the other only two-way (for PPW).

5.3 Sign Prediction

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 53% to be positive owing to the 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.

5.4 Overall Accuracy

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 5.3.

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%.

5.5 Comparison with State of the Art

A general purpose MWP solver is available via the publicly available WolframAlpha engine (a blog post associated with its functionality elaborates the development process and the diagrammatic solution feature of this solver) [13]. 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.

For the purpose of this comparison, we choose a random set of 20% of our corpus and compare the accuracy of solutions produced by the solvers. While our MWP solver had an accuracy of 94.4% on the sample, the performance of WolframAlpha 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 code "Wolfram—Alpha doesn't understand your query". We conclude that the MWP solver aspect of the WolframAlpha engine has a very poor performance, and particularly so when compared to our solver.

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

5.6 Ablation Analysis

Table 5.4: Comparison of the accuracy results (in %) with different feature classes ablated for each classification task with the accuracy where no features were excluded.

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 5.4 summarizes the results of the ablation study 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 problems).

Sign prediction depends not only on verbs but other features as well. The sentence-level features play an important role, followed by the action-related features.

The J-S and PPW sentence type classifiers suffer the most from missing the sentence level features because the decisions are made primarily at a sentence level.

CHAPTER 6

DISCUSSION

6.1 Error Analysis

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 is in complex sentence structure. For instant *How many Yodas flew away from the planet in the space shuttle if 23 Yodas stayed on the planet of 30 Yodas in all?*

Next, we consider the errors that occur in the J-S problem solver. The majority of errors result from incorrect sign prediction, explained by the fact that the sign prediction 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.

6.2 Effect of Unbalanced Corpus

In certain classification algorithms, unbalanced data sets can be a major issue since the classifier will focus on the majority classes, ignoring the minority ones [34]. Indeed, popular classifiers such as SVMs suffer significantly from data imbalance [35]. A severe case in our data set is the number of sentences of type *Unknown* in J-S and PPW. Their role in the discourse structure of the problem is that they constitute irrelevant information setting up the context of the problem, but not actually aiding the solution. The *unknown* type in classifying sentence type is the minority class, with only 23 instances among 1471 samples (1.5%) and its lack of instances affects the ability of the classifier to model them causing them to be labelled as one of the other sentence types.

We observe that the Random Forest classifier provides the highest accuracy (in the sentence function identification problem) among the classifiers tested – as such, ensemble method in conjunction with the randomness in this classifier make it better suited to work with unbalanced data.

6.3 Generalizability

A new set of MWP is collected from DadsWorksheets.com.[1] We were considered only addition and subtraction problems from this website for 400 problems. These problems does not have CGI structure. They are regular elementary MWP. We tested the new corpus in our system. The accuracy became 87%.

[1]www.dadsworksheets.com visited on March 12, 2013.

Beside the set of questions we did not study as mentioned in Section 6.1, we also were not considered some enigmatic problem such as *Sharon has 80 bananas. Chris has 6 bananas. Katherine takes 28 away. How many bananas will Sharon have?.* This particular question is unclear whether Katherine tooks bananas from Chris or Sharon so that there will be two possible answers for this question. All in all, the question with two or more possible answers cannot be processed through our solver.

There are three main sources of errors. One is the error in the sign prediction stage. The features that we had could not capture some structures of questions in the new unseen set so that the system could not provide correct labels. As expected, the accuracy of sign prediction is a bottleneck of our system. Another is sentence type identification in J-S problem. It misclassified between *given* and *change* sentence. The failure were occured on the problem with a sequence structure as *CGXR*. The other one is from problem type classification. This source of error was not significant compared to prior causes. Even though a PPW problem is classified as JS, our system is still able to produce an accurate equation.

CHAPTER 7

CONCLUSION

We present a multistage 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.

Looking ahead, we are working to solve more complicated MWPs of upper elementary grades and certain college-level MWPs such as those in introductory probability, which we have been teaching at an undergraduate level. Another thing that we are currently working on is about classifying the difficulty level of each MWP so that we can personalize the MWP exercise for each student.

REFERENCES

[1] M. Heilman, K. Collins-thompson, J. Callan, and M. Eskenazi, "Classroom success of an intelligent tutoring system for lexical practice and reading comprehension," in *Proceedings of the Ninth International Conference on Spoken Language Processing*, 2006.

[2] K. R. Koedinger, J. R. Anderson, W. H. Hadley, M. A. Mark et al., "Intelligent tutoring goes to school in the big city," *International Journal of Artificial Intelligence in Education (IJAIED)*, vol. 8, pp. 30–43, 1997.

[3] M. D. LeBlanc and S. Weber-Russell, "Text integration and mathematical connections: A computer model of arithmetic word problem solving," *Cognitive Science*, vol. 20, no. 3, pp. 357–407, 1996.

[4] W. Supap, K. Naruedomkul, and N. Cercone, "Mathmaster: An alternative math word problems translation," *Computational Approaches to Assistive Technologies for People with Disabilities*, vol. 253, p. 109, 2013.

[5] D. G. Bobrow, "A question-answering system for high school algebra word problems," in *Proceedings of the October 27-29, 1964, Fall Joint Computer Conference, Part I*, ser. AFIPS '64 (Fall, part I). New York, NY, USA: ACM, 1964. [Online]. Available: http://doi.acm.org/10.1145/1464052.1464108 pp. 591–614.

[6] C. Liguda and T. Pfeiffer, "A question answer system for math word problems," in *First International Workshop on Algorithmic Intelligence*, H. Messerschmidt, Ed., 2011.

[7] C. Liguda and T. Pfeiffer, "Modeling math word problems with augmented semantic networks," in *Natural Language Processing and Information Systems*, ser. Lecture Notes in Computer Science, G. Bouma, A. Ittoo, E. Mtais, and H. Wortmann, Eds. Springer Berlin Heidelberg, 2012, vol. 7337, pp. 247–252. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-31178-9_29

[8] E. De Corte and L. Verschaffel, "The effect of semantic structure on first graders' strategies for solving addition and subtraction word problems," *Journal for Research in Mathematics Education*, pp. 363–381, 1987.

[9] D. D. Cummins, "Children's interpretations of arithmetic word problems," *Cognition and Instruction*, vol. 8, no. 3, pp. 261–289, 1991. [Online]. Available: http://www.jstor.org/stable/3233626

[10] L. Verschaffel, B. Greer, and E. De Corte, *Making Sense of Word Problems.* Lisse, 2000.

[11] Y. Yang and X. Liu, "A re-examination of text categorization methods," in *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval.* ACM, 1999, pp. 42–49.

[12] S. Cetintas, L. Si, Y. P. Xin, D. Zhang, and J. Y. Park, "Automatic text categorization of mathematical word problems," in *FLAIRS Conference*, 2009.

[13] P. Barendse, "Sovling word problems with Wolfram|Alpha@ONLINE," Oct. 2012. [Online]. Available: http://blog.wolframalpha.com/2012/10/04/solving-word-problems-with-wolframalpha/

[14] C. R. Fletcher, "Understanding and solving arithmetic word problems: A computer simulation," *Behavior Research Methods*, vol. 17, no. 5, pp. 565–571, 1985. [Online]. Available: http://dx.doi.org/10.3758/BF03207654

[15] D. Dellarosa, "Solution: A computer simulation of childrens recall of arithmetic word problem solving," *University of Colorado Technical Report*, pp. 85–148, 1985.

[16] D. Dellarosa, "A computer simulation of children's arithmetic word-problem solving." *Behavior Research Methods*, vol. 18, pp. 147–154, March 1986.

[17] S. Cetintas, L. Si, Y. P. Xin, D. Zhang, J. Y. Park, and R. Tzur, "A joint probabilistic classification model of relevant and irrelevant sentences in mathematical word problems," *JEDM-Journal of Educational Data Mining*, vol. 2, no. 1, pp. 83–101, 2010.

[18] T. Matsuzaki, H. Iwane, H. Anai, and N. Arai, "The complexity of math problems – Linguistic, or computational?" in *Proceedings of the Sixth International Joint Conference on Natural Language Processing.* Nagoya, Japan: Asian Federation of Natural Language Processing, October 2013. [Online]. Available: http://www.aclweb.org/anthology/I13-1009 pp. 73–81.

[19] N. Kushman, Y. Artzi, L. Zettlemoyer, and R. Barzilay, "Learning to automatically slove algebra word problems," *ACL Conference 2014*, 2014.

[20] Y. Guo, A. Korhonen, M. Liakata, I. S. Karolinska, L. Sun, and U. Stenius, "Identifying the information structure of scientific abstracts: An investigation of three different schemes," in *Proceedings of the 2010 Workshop on Biomedical Natural Language Processing*, ser. BioNLP '10. Stroudsburg, PA, USA: Association for Computational Linguistics, 2010. [Online]. Available: http://dl.acm.org/citation.cfm?id=1869961.1869974 pp. 99–107.

[21] A. Chaganty, A. Lal, A. V. Nori, and S. K. Rajamani, "Combining relational learning with SMT solvers using CEGAR," in *Computer Aided Verification.* Springer, 2013, pp. 447–462.

[22] L. Liu, C.-H. Lin, and S. Vasudevan, "Word level feature discovery to enhance quality of assertion mining," in *Computer-Aided Design (ICCAD), 2012 IEEE/ACM International Conference*, Nov 2012, pp. 210–217.

[23] T. P. Carpenter, E. Fennema, M. L. Franke, L. Levi, and S. B. Empson, "Cognitively guided instruction: A research-based teacher professional development program for elementary school mathematics research report," 2000.

[24] S. Olson, N. Musser, S. McAdaragh, R. Dyk, J. Weber, T. Mittleider, L. Atwood, and M. Torgrude, "South Dakota counts: CGI problems created by South Dakota math teacher leaders @ONLINE," Jan. 2008. [Online]. Available: http://sdesa.k12.sd.us/esa5/docs/sdcounts/SDCountsMathProblemBooklet.pdf

[25] K. Ebner, "Cognitively guided instruction (CGI) problem types @ONLINE," July 2011. [Online]. Available: http://sdesa.k12.sd.us/esa5/docs/sdcounts/SDCountsMathProblemBooklet.pdf

[26] 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, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[27] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA data mining software: An update," *SIGKDD Explor. Newsl.*, vol. 11, no. 1, pp. 10–18, Nov. 2009. [Online]. Available: http://doi.acm.org/10.1145/1656274.1656278

[28] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 27:1–27:27, 2011, software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[29] T. M. Mitchell, *Machine Learning*, 1st ed. New York, NY, USA: McGraw-Hill, Inc., 1997.

[30] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, Oct. 2001. [Online]. Available: http://dx.doi.org/10.1023/A:1010933404324

[31] E. Loper and S. Bird, "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*, ser. ETMTNLP '02. Stroudsburg, PA, USA: Association for Computational Linguistics, 2002. [Online]. Available: http://dx.doi.org/10.3115/1118108.1118117 pp. 63–70.

[32] T. E. Oliphant, *Guide to NumPy*, Provo, UT, Mar. 2006.

[33] M.-C. De Marneffe, B. MacCartney, C. D. Manning et al., “Generating typed dependency parses from phrase structure parses,” in *Proceedings of LREC*, vol. 6, 2006, pp. 449–454.

[34] R. Longadge and S. Dongre, “Class imbalance problem in data mining review,” *ArXiv e-prints*, May 2013.

[35] R. Batuwita and V. Palade, “Class imbalance learning methods for support vector machines,” *Imbalanced Learning: Foundations, Algorithms, and Applications*, p. 83, 2013.