Learning to Solve Percentage Word Problems by Parsing into Equations

Abstract—**Solving Math Word Problems is a big challenge in Natural Language Processing. This paper presents an approach to solve Percentage Word Problem (i.e. one of Math Word Problems which described in natural language and contains percentage related problem) by parsing into equations. This is the first approach to solve percentage word problems. In our system we use semantic parsing to generate equation trees with the help of Integer Linear Programming (ILP). We conducted experiments on a test set about 300 problems and our system is able to solve 63.2% of the problems.**

Keywords—Natural Language Processing, Percentage Word Problem Solving, Semantic Parsing.

I. INTRODUCTION

Newspaper articles on stock prices, business news, several advertise on product's current rate, offers on products of super shops are described in Natural Language. Even Sport commentaries, election results, modern Chat bots for Questions and Answers are also in Natural Language. These contains several percentage related problems or context (see Figure 1). So, it's challenging to manually handle all the problems. We need an algorithm to solve those percentage word problems. Percentage Word Problems are the mathematical problem of "percentage" in natural language.

> Clean Machine is Ashland's premier house cleaning service. The company used 9,810 gallons of soap last year, and 60% more than that this year. How many gallons of soap did the company use this year?

Figure 1. Example of a Percentage Word Problem

Solving Word Problems requires semantic parsing and reasoning across sentence to find equations. In our system, we uses verb categorization to ground the problem text and divide them entities, containers, quantities etc. by semantic parsing. Then mapping possible equation trees. In previous, math word problems are tried to solve with verb categorization [1] and template based method [2]. ALGES [3] is hybrid method which combines both verb categorization and template based method for solving single variable addition, subtraction, multiplication and division problems.

In our system, we use ALGES to a broader scope to solve percentage word problems. We have converted the problem text first. Then generate equation trees by semantic parsing and Integer Linear Programming (ILP) to solve the problem.

Our contributions are as follows: (1) We have converted and preprocessed the problem text like, addition, subtraction, multiplication and division problems; (2) A newly build efficient dataset on Percentage Word Problems; and Finally, (3) a system named **PWPS** that can solve 63.02% Percentage Word Problems.

II. RELATED WORK

Automatically solving Math Word Problem is a recently hot topic on understanding Natural Language. But actual journey of solving was started from 1960s. Algebraic problems of Natural Language was transformed into kernel sentences and handles to solve the problems in STUDENT [4]. In CARPS [5], they uses pattern matching based on expressions by the transformed kernel sentences. But they were limited to rate based problems. In [6], they first introduced tree based structure to represent the information in the problem. Recent automatically solving math word problems include number word problems [7], logic puzzle problems [8], geometry word problems [9]-[10], arithmetic word problems [1], [11] and algebra word problems [2]-[3], [12].

In semantic parsing, there has been much works. Language grounding for interpretation of a sentence in world representation has related to many works [13]-[23]. We discuss three pioneering work closely related to our work.

In [1], they tried to solve addition and subtraction problems by verb categories for the purpose of updating a world representation derived from problem text. They ground the problem text to semantic *entities* and *containers*. Based on learned verb categories, their system works well for + and - .

In [2], they introduced a general method for solving algebra problems. Their system maps the problem text to equations with one or two unknown variables by generating templates from global and local features of the problem text.

In ALGES [3], they tried to solve the problem of solving multiple sentenced algebraic word problems by generating and ranking the equation trees. They uses a richer semantic representation of the problem text and a bottom-up approach to learn the relations between spans of texts and arithmetic operators. Then score the equations using global form of the problem to produce the final result. ALGES combined the previous methods to use in broader scope like, Addition, Subtraction, Multiplication and Division for solving single variable problems.

Our work is related to ALGES, where we converted the percent related number to fraction and force the problem text to covert it as the problem for ALGES. That is related to using ILP to enforce global constraints in NLP applications [24]. Like previous [25] - [28], ALGES used ILP to form candidate equations which are then used to generate training data for classifications. ALGES attempts to parser re-rank the equations [29] - [30].

III. PROPOSED METHOD

The goal of our system PWPS is to solve percentage word problems with the help of ALGES which is a step towards solving math word problems automatically.

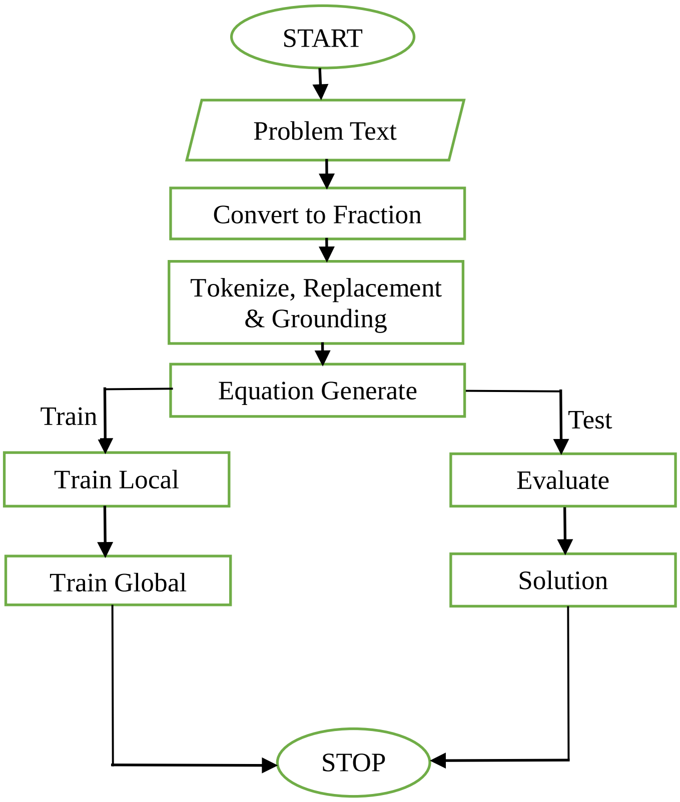

Figure 2. Flowchart of the proposed PWPS system

Figure 2 gives the overview of our proposed system Percentage Word Problem Solver (PWPS). After equation generating there is two phase. One is training phase and another is testing phase. All the training steps in the algorithm is discussed in this paper as *"Learning"* and testing steps is discussed as *"Inference"*. Learning contains train local and global model and inference contains equation evaluation and solution checking with respect to the actual solution of the problem text. Pseudocode of our system is given in Fig. [3].

Learning

1. Make a pair of problem text, *p* and solution, *l*
2. Covert to fraction
3. Tokenize, replace and ground *p* to base Qsets, *S*.
4. T_i is the generated top *M* candidate trees by ILP
5. $T_{li} \leftarrow$ Select best equation trees based on the solution, *l*
6. Extract Local Model features for Qset <s_1, s_2> labeled with *op*.
7. Extract Global Model features by T_i and solution of T_i labeled with positive or negative
8. $L_{local} \leftarrow$ Train Local Model
9. $G_{global} \leftarrow$ Train Global Model

Inference

1. Step 1 – 7 from *Learning*
2. Select best equation based on the score from local and global training model
3. Evaluate the equation
4. Output the solution

Figure 3. Pseudocode of PWPS

We have discussed the process in details below:

A. *Convert to Fraction*

To solve percentage problems as addition, subtraction, multiplication or division, we need to convert the percentage related number to fraction with respect to number *100*. If the given problem text, *p* has a number "x*%"* then, we convert it to *"y"*, where y less than *x*, and $x,y \in \mathbb{R}^+$.

$$x = \frac{x}{100} = y, where, x > 0$$

Problem text of Figure 1, is converted to fractions and replace the '%' with ***times*** which is discussed in the *tokenize, replacement and grounding* section below.

Clean Machine is Ashland's premier house cleaning service. The company used 9,810 gallons of soap last year, and **0.60 times** more than that this year. How many gallons of soap did the company use this year?

Figure 4. Converted to Fraction and '%' sign replace by 'times' in the problem text

B. *Tokenize Replacement and Grounding*

In order to build equation trees from the problem text, *P* we tokenized the text to words. We changed the symbols of the problem text to corresponding word. *'$'* is changed to *'dollar',*

'%' to *'times'* where *'times'* enforce ALGES to count the statement as a multiplication operation.

A *Quantified Set* or *Qset* is a node to model problem text quantities and their properties. To generate equation tress, we need to combine the Qsets. A *base Qset* is a tuple of *ent, qnt, adj, loc, vrb and ctr*. The properties are described in the table below:

TABLE I. THE PROCESS OF FORMING A SINGLE QSET [3]

Item	Properties
qnt	*qnt (Quantity)* is a numerical determiner in the problem text, *P*
ent	*ent (Entity)* is a noun related to *qnt.*
adj	*ctr (Container)* is the subject of the verb that governs *ent*
loc	*loc* (Location) is a noun related to *ent*
vrb	*vrb* (Verb) is a governing verb
ctr	*ctr* (Container) is the subject of the verb governing

A *Qset* is ground as a compact representation of the properties based on Table I. Grounded Qsets are generally two types. One is – Normal Qset and Target Qset. Target is the Qset where *what, how many or how much* words or phrases are present.

Space of possible equation trees are reduced by reordering the Qsets. ALGES [3] employed three some rules to reorder the Qsets as in the TABLE II.

TABLE II. RULES FOR REORDERING QSETS [3]

1. Move Qset s_i to immediately after Qset s_j if the container of s_i is the entity of s_j and is quantified by '*each'*.
2. Move target Qset to the front of list if the question statement includes keywords *start* or *begin*.
3. Move target Qset to the end of the list if the problem text includes keyword *left, remain,* and *finish.*
4. Move target Qset to the textual location of an intermediate reference with the same *ent* if its *num* property is the determiner *some*.

Reordered Qsets are then combine by some arithmetic operators. If *a* and *b* are two Qsets, then a new Qset *c* can be formatted as $c \leftarrow (a, b, op)$, where *op* is the operator.

Qnt: 9810 Ent: Gallons	Qnt: 0.60 Ent: None	Qnt: x Ent: Gallons

Figure 5. Ground problem text of Figure 3 to base Qsets

C. Generate Equations

ALGES uses Integer Linear Programming (ILP) to generate equation trees from the base Qsets. These equations are then used for learning and inferencing the system PWPS and selects best *M* candidate equations for a given problem text, *p*.

For problem text, *p* and *n* base Qsets, ALGES builds ILP(*P*) over the space of postfix equations $E = e_1, e_2, \ldots e_L$ of length *L* and *k* numeric constants, $k' = n - k$ unknowns, *r* binary operators and *q "types"* of Qsets.

TABLE III. ILP NOTATION FOR CANDIDATE EQUATIONS MODEL [3]

INPUT	
p	Problem Text
n	Number of base Qsets
k	Numeric Constant
k'	Number of Unknowns
r	Number of Binary Operators
m	Number of Possible Symbols *(n+r)*
$type_j$	type of *j-th* base Qsets
M	desired number of candidate equations
L	desired length of postfix notations
OUTPUT	
E	Postfix equation to be generated

In the TABLE III, the notations for generating candidate equation trees are givens. In Figure 5, we showed the subset of the candidate equation trees based on the problem text, *p* in Figure 1, *x* is the unknown variable

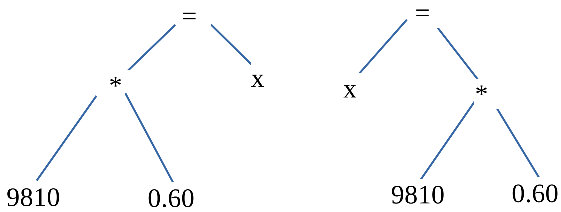

Figure 6. Candidate Equation Generated using ILP

D. Learning

Learning the training step in [2]. In learning, our system will learn from the score of the equations based on the solution of the problem text, *p* like ALGES. Our dataset contains problem text-solution pairs $\{w_i, l_i\}_{i=1,2\ldots N}$. Learning process can be divided into two parts. One is – Local Qset Relationship Model and another is – Global Equation Model. Local Model and Global Models are train based on the problem text, *p* and solution of that problem.

1) Local Qset Relationship Model

Local Qset Relationship model is learned from the equation tree. For each equation tree, two base Qset s_1 and s_2 are used to extract the features and labeled with *op* as train data. If $op \in \{+, -, *, /\}$ then, $L_{local} = \theta^T f_{local}(s_1, s_2)$ where f_{local} is the feature vector between the Qsets.

Features between the Qsets are designed semantic and inter textual relation relationships. Feature vector is included as in the TABLE IV from 1 to 3. Similarity for the features of the Qsets measured based on Lin Similarity [31].

2) Global Equation Model

Our system train global equation model to score the equation trees as in ALGES. $G_{global} = \gamma^T f_{global}(p, t)$ where f_{global} is the feature vector capturing the trees, *t* and the problem text, *p*. Root node will set based on the local models prediction of left and right of the equal operator. And the root node features are extracted as in TABLE IV from 4.

TABLE IV. FEATURES USED FOR LOCAL AND GLOBAL MODEL [3]

1. Single Qset Features • What argument of its governing verb A? • Is A a subset of another set? • Is A a compound? • Math keywords found in context of A? • Verb Lin Distance from known verb categories? 2. Relational features between Qsets • Entity Match • Adjective Overlap • Location Match • Distance in text • Lin Similarity 3. Target Qset Features • Which one is target Qset? • Entity Matched with target entity?
4. Root Node Features • Number of ILP constraints violated by equation • Scores of left and right subtrees of root

E. Inference

Inference is the testing steps in Fig. [2]. In inference for a problem text, P firstly ILP(p) generates the candidate equations. On the candidate equations, score is calculated from local Qset Relationship model and Global Equation Model. And the final score for a candidate equation is calculated through the likelihood $p(t|p)$.

$$p(t \mid p) \propto \left(\prod_{t_i \in t} L_{local}(t_i \mid p) \right) \left(G_{global}(t \mid p) \right)$$

Where, t_i is the subtree and t is the root of the equation. Among all the scores, the candidate equation with highest score is selected for the final equation.

IV. EXPERIMENTS

In this section, we will present the experimental analysis of our system PWPS.

A. Experimental Setup

In our system, we used Stanford Dependency Parser in CoreNLP 3.4 [32] for grounding the problem text and feature extraction. For generating $M = 100$ candidate equations by Integer Linear Programming, we used CPLEX 12.6.1 [33]. To train SVM classifier, LIBSVM [34] is used with RBF kernels.

B. Dataset

For this work, we collected a new dataset from `http://math-aids.com`, `http://ixl.com`, `https://www.khanacademy.org` and `http://algebra.com`. Dataset statistics is given below in TABLE V.

TABLE V. DATASET STATISTICS

Items	#
Number of Problems in Dataset	278
Number of Sentences in Dataset	896
Number of Words in Dataset	8566
Average Sentences per Problem	3.2
Average Words per Problem	30.8

C. Results

For calculating the performance of our system, we applied 5-fold cross validation. In our dataset, total number of problems is 278 and our system could solve 176 problem correctly. Based on this, accuracy of our system is – 63.2%.

a) Ablation Study

For analyzing the performance we perform the following ablations:

- *No Local Model*

  We test our system without local model for generating the equations. And it's only based on Global Model. Likelihood equation without Local Model is like below:

$$p(t \mid p) \propto \left(G_{global}(t \mid p) \right)$$

- *No Global Model*

  Here, we test our system without the global model for generating the equations which is based on the all local score of the equation. Likelihood equation without Global Model is like below:

$$p(t \mid p) \propto \left(\prod_{t_i \in t} L_{local}(t_i \mid p) \right)$$

TABLE VI. ABLATION STUDY OF PWPS

Method	Accuracy
PWPS	0.632
No Local Model	0.451
No Global Model	0.312

b) Error Analysis

Analysis about 10 errors, we summaries that our system is failed to grounding into the problem like TABLE VII.

TABLE VII. EXAMPLE OF PROBLEM OUR SYSTEM CAN'T SOLVE CORRECTLY

Kira's Cafe has regular coffee and decaffeinated coffee. This morning, the cafe served 13 regular coffees and 39 decaffeinated coffees. What percentage of the coffees served were regular?

In the problem in TABLE VII, parser treats only for *regular* and *decaffeinated* words. But it should consider the word *coffees* for grounding Qsets.

CONCLUSIONS

In this system, we have a new outline method for solving Percentage Word Problems. We have collected a new dataset which will help researcher to expand the present condition of mathematical word problem solving. The accuracy of our system can be improve by optimizing the errors. And this system can be farther expanded to other domains like physics, chemistry and so on.

ACKNOWLEDGMENT

This research is supported by Department of Computer Science and Engineering, RUET, Bangladesh. We would like to thanks anonymous reviewers and a big thanks to Rik Koncel-Kedziorski for his email support. Our code and dataset are publicly available at `https://github.com/habibrahmanbd/PWPS`.

REFERENCES

[1] Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman, "Learning to solve arithmetic word problems with verb categorization", in *Empirical Methods in Natural Language Processing*, pages 523–533, 2014.

[2] Nate Kushman, Yoav Artzi, Luke Zettlemoyer, and Regina Barzilay, "Learning to automatically solve algebra word problems" in *Association for Computational Linguistics*, pages 271–281, 2014.

[3] Rik Koncel-Kedziorski, Hannaneh Hajishirzi, Ashish Sabharwal, Oren Etzioni, and Siena Dumas Ang, "Parsing Algebraic Word Problems into Equations", in *Empirical Methods in Natural Language Processing*, pages 1132–1142, 2015.

[4] D.G. Bobrow, "Natural language input for a computer problem solving system", Report *MAC-TR-1*, Project MAC, MIT, Cambridge, 1964a.

[5] E. Charniak, "CARPS: a program which solves calculus word problems", Report *MAC-TR-51*, Project MAC, MIT, Cambridge, 1968.

[6] C. Liguda, T. Pfeiffer, "Modeling Math Word Problems with Augmented Semantic Networks", *NLDB'2012*, pp. 247-252, 2012.

[7] Shuming Shi, Yuehui Wang, Chin-Yew Lin, Xiaojiang Liu and Yong Rui, "Automatically Solving Number Word Problems by Semantic Parsing and Reasoning", *in Empirical Methods in Natural Language Processing*, pages 1132–1142, Lisbon, Portugal, 2015.

[8] Arindam Mitra and Chitta Baral, "Learning to automatically solve logic grid puzzles", in *Empirical Methods in Natural Language Processing*, 2015.

[9] Min Joon Seo, Hannaneh Hajishirzi, Ali Farhadi, and Oren Etzioni, "Diagram understanding in geometry questions", in *AAAI*, 2014.

[10] Minjoon Seo, Hannaneh Hajishirzi, Ali Farhadi, Oren Etzioni, and Clint Malcolm, "Solving geometry problems: Combining text and diagram interpretation", in *Empirical Methods in Natural Language Processing*, 2015.

[11] Subhro Roy and Dan Roth, "Solving general arithmetic word problems", in *Empirical Methods in Natural Language Processing*, 2015.

[12] Lipu Zhou, Shuaixiang Dai, and Liwei Chen, "Learn to solve algebra word problems using quadratic programming", in *Empirical Methods in Natural Language Processing*, 2015.

[13] S. R. K. Branavan, Harr Chen, Luke S. Zettlemoyer, and Regina Barzilay, "Reinforcement learning for mapping instructions to actions", in *ACL/AFNLP*, pages 82–90, 2009.

[14] Percy Liang, Michael I. Jordan, and Dan Klein, "Learning semantic correspondences with less supervision", in *ACL/AFNLP*, pages 91–99, 2009.

[15] David L. Chen, Joohyun Kim, and Raymond J. Mooney, "Training a multilingual sportscaster: Using perceptual context to learn language", *JAIR*, 37:397–435, 2010.

[16] Antoine Bordes, Nicolas Usunier, and Jason Weston, "Label ranking under ambiguous supervision for learning semantic correspondences", in *ICML*, pages 103–110, 2010.

[17] Yansong Feng and Mirella Lapata, "How many words is a picture worth? Automatic caption generation for news images", in *ACL*, pages 1239–1249, 2010.

[18] Hannaneh Hajishirzi, Mohammad Rastegari, Ali Farhadi, and Jessica K. Hodgins, "Semantic understanding of professional soccer commentaries", in *Uncertainty in Artificial Intelligence, 2012.*

[19] Hannaneh Hajishirzi, Julia Hockenmaier, Erik T. Mueller, and Eyal Amir, "Reasoning about robocup soccer narratives", in *Uncertainty in Artificial Intelligence*, pages 291–300, 2011.

[20] Cynthia Matuszek, Evan Herbst, Luke Zettlemoyer, and Dieter Fox, "Learning to parse natural language commands to a robot control system" in Proc. of the 13th *International Symposium on Experimental Robotics (ISER)*, 2012.

[21] Yoav Artzi and Luke Zettlemoyer, "Weakly supervised learning of semantic parsers for mapping instructions to actions", in *TACL*, 1(1):49–62, 2013.

[22] Mark Yatskar, Lucy Vanderwende, and Luke Zettlemoyer, "See no evil, say no evil: Description generation from densely labeled images", in *Lexical and Computational Semantics* (* SEM 2014), page 110, 2014.

[23] Ben Hixon, Peter Clark, and Hannaneh Hajishirzi, "Learning knowledge graphs for question answering through conversational dialog", in *North American Chapter of the Association for Computational Linguistics*, 2015.

[24] Dan Roth and Wen-tau Yih, "A linear programming formulation for global inference in natural language tasks", in Hwee Tou Ng and Ellen Riloff, editors, CoNLL, pages 1–8, *Association for Computational Linguistics*, 2004.

[25] Vivek Srikumar and Dan Roth, "A joint model for extended semantic role labeling" in *EMNLP*, Edinburgh, Scotland, 2011.

[26] D. Goldwasser and D. Roth, "Learning from natural instructions", in *IJCAI*, 2011.

[27] Jonathan Berant, Vivek Srikumar, Pei-Chun Chen, Abby Vander Linden, Brittany Harding, Brad Huang, Peter Clark, and Christopher D. Manning, "Modeling biological processes for reading comprehension", in *EMNLP*, 2014.

[28] Fei Liu, Jeffrey Flanigan, Sam Thomson, Norman Sadeh, and Noah A. Smith, "Toward abstractive summarization using semantic representations", in *North American Chapter of the Association for Computational Linguistics*, 2015.

[29] Michael Collins, "Discriminative re-ranking for natural language parsing", *Computational Linguistics*, 31(1):25–70, 2005.

[30] Ruifang Ge and Raymond J. Mooney, "Discriminative re-ranking for semantic parsing", in *Association for Computational Linguistics*, 2006.

[31] Dekang Lin, "An information-theoretic definition of similarity" in *ICML*, volume 98, pages 296–304, 1998.

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

[33] IBM ILOG. 2014. IBM ILOG CPLEX Optimization Studio 12.6.1.

[34] Chih-Chung Chang and Chih-Jen Lin, "LIBSVM: A library for support vector machines", *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011.