A Tag-based English Math Word Problem Solver with Understanding, Reasoning and Explanation

Chao-Chun Liang, Kuang-Yi Hsu, Chien-Tsung Huang,
Chung-Min Li, Shen-Yu Miao, Keh-Yih Su
Institute of Information Science, Academia Sinica, Taiwan
{ccliang, ianhsu, joecth, cmli, jackymiu, kysu}@iis.sinica.edu.tw

Abstract

This paper presents a tag-based statistical math word problem solver with understanding, reasoning, and explanation. It analyzes the text and transforms both body and question parts into their tag-based logic forms, and then performs inference on them. The proposed tag-based approach provides the flexibility for annotating an extracted math quantity with its associated syntactic and semantic information, which can be used to identify the desired operand and filter out irrelevant quantities. The proposed approach is thus less sensitive to the irrelevant information and could provide the answer more precisely. Also, it can handle much more problem types other than addition and subtraction.

1 Introduction

The math word problem (MWP) (Mukherjee and Garain, 2008) is frequently chosen to study natural language understanding due to the following reasons: (1) Since the answer for the MWP cannot be extracted by simply performing keyword/pattern matching, MWP can clearly show the merits of understanding and inference. (2) As MWP usually possesses less complicated syntax and requires less amount of domain knowledge, it can let the researcher focus on the task of understanding and reasoning. (3) The body part of MWP (which mentions the given information for solving the problem) consists of only a few sentences. The understanding and reasoning procedure could be checked more efficiently. (4) The MWP solver has its own standalone applications such as *Computer Math Tutor* and *Helper for Math in Daily Life*.

Previous English MWP solvers can be classified into three categories: (1) Rule-based approaches with logic inference (Bobrow, 1964; Slagle, 1965), which apply rules to get the answer (via identifying entities, quantities, operations, etc.) with a logic inference engine. (2) Rule-based approaches without logic inference (Charniak, 1968 and 1969; Gelb, 1971; Ballard, 1979; Biermann and Ballard, 1980; Biermann et al., 1982; Fletcher, 1985; Dellarosa, 1986; Bakman, 2007; Liguda and Pfeiffer, 2012; Hosseini et al., 2014), which apply rules (usually defined as schemata) to get the answer without a logic inference engine. (3) Purely statistic-based approaches (Kushman et al., 2014; Roy et al., 2015), which use statistical models to identify entities, quantities, operations, and get the answer without conducting language analysis or inference.

The main problem of the rule-based approaches mentioned above is that the coverage rate problem is serious, as rules with wide coverage are difficult and expensive to construct. Also, it is awkward in resolving ambiguity problems. Besides, most of them only handle *addition and subtraction* these two math operations. On the other hand, the main problem of those approaches without adopting logic inference is that they cannot share the common reasoning part among various problem types. In contrast, the main problems of those purely statistical approaches are that they are sensitive to irrel-

Proceedings of NAACL-HLT 2016 (Demonstrations), pages 67–71,
San Diego, California, June 12-17, 2016.

evant information and that the performance deteriorates significantly when they encounter complicated problems (Hosseini et al., 2014), because the problem is solved without first understanding the text. Besides, they only handle algebra problems.

A *tag-based statistical English MWP solver* is thus proposed to perform understanding and reasoning, and avoid the problems mentioned above. The text of the MWP is first analyzed into its corresponding syntactic tree and then annotated with resolved co-reference chains. Afterwards, it is converted into the logic form via a few mapping rules. The obtained logic form is further mapped into the corresponding domain dependent generic concepts (also expressed in logic form). Finally, the logic inference is performed on those logic statements to get the answer. Various statistical classifiers are applied when there are choices.

Since different questions could be asked for the same given body text, we keep all syntactic relations in the logic form, which are regarded as various tags for selecting the appropriate operands related to the specified question. For example, "*Fred picked 36 limes*" will be converted into "*quan*(*q1*, *#*, *lime*)&*verb*(*q1*, *pick*)&*nsubj*(*q1*, *Fred*) = *36*" (where tags are connected with logic "&"), in which the quantity "*36*" is identified with a label "*q1*" and attached with its associated tags. The proposed tag provides the flexibility for annotating a given math quantity with associated syntactic and semantic information, which can be used to identify the desired operand and filter out irrelevant quantities. It thus makes our MWP solver less sensitive to the irrelevant information and could provide the answer more precisely.

2 System Framework

The block diagram of proposed math word problem solver is shown in Figure 1. First, each sentence in an MWP is analyzed by the *Language Analyzer* (LA) module. The associated linguistic information is then sent to the *Solution Type Classifier* (STC) to find out the corresponding math operation. Afterwards, they are converted into the logic form by the *Logic Form Converter* (LFC). The *Inference Engine* (IE) then obtains the answer from those obtained logic expressions. Finally, the *Explanation Generator* (EG) module will explain how the answer is obtained according to the given

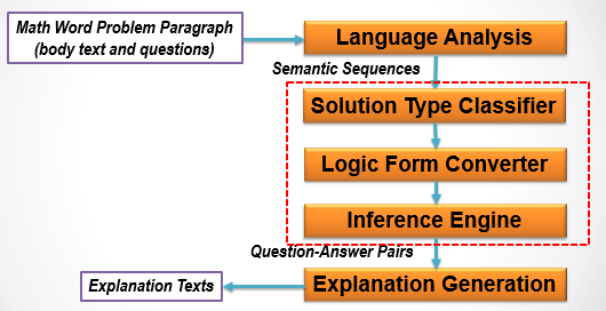

Figure 1: The block diagram of the proposed MWP solver

reasoning chain (Russel and Norvig, 2009).

Among those modules, the STC is responsible for suggesting a way (i.e., a solution type such as addition, subtraction, multiplication, division, etc.) to solve the problem for each question of the MWP. The LFC extracts the related facts from the given linguistic information and then represents those facts as the *first-order logic* (FOL) predicates/functions (Russel and Norvig, 2009). It also transforms each question into a FOL-like utility function according to the suggested solution type. The IE then derives new facts according to inference rules and old facts provided by the LFC. It is also responsible for providing utilities to perform math operations on related facts to get the answer. Detailed description of each module is given below.

2.1 Language Analysis

The Stanford CoreNLP suite (Manning et al., 2014) is adopted as our LA, which enables a list of annotators to generate the necessary linguistic information. The list includes: tokenization, sentence splitting, POS tagging, lemmatization, named entity recognition, parsing and co-reference resolution. The generated linguistic representation mainly depicts the syntactic relations between its words. To solve MWPs, it is crucial to know the relations between various entities. Dependency relation and co-reference resolution will provide such information.

2.2 Solution Type Identification

The STC will select a math operation (that LFC should adopt to solve the problem) based on the global information across various input sentences. Table 1 shows 12 different solution types currently provided. A SVM classifier with linear kernel functions (Chang and Lin, 2011) is used, and it

adopted various feature-sets: (1) verb category related features, (2) various keyword indicators, and (3) different pattern-matching indicators for various specified aggregative patterns.

Addition	Multiplication	Surplus
Subtraction	Common-Division	Comparison
Sum	Floor-Division	Algebra
Difference	Ceil-Division	Time Variant

Table 1: Various solution types for solving the MWP

2.3 Logic Form Transformation

A two-stage approach is adopted to transform the linguistic representation of a sentence into its corresponding logic forms. In the first stage, the FOL predicates are generated (via a few deterministic mapping rules) by traversing the input linguistic representation. For example, the sentence "*Fred picks 36 limes*" will be transformed into the following FOL predicates separated by the logic AND operator "&":

verb(*v1*,*pick*)&*nsubj*(*v1*, *Fred*)&
dobj(*v1*,*n1*)&*head*(*n1*, *lime*)&*nummod*(*n1*, *36*)

All the first arguments of the above FOL predicates (i.e., *v1* and *n1*) are identifies, and the predicate-names are the domain-independent syntactic dependency relation of the constituents in the dependency structure.

The domain-dependent logic forms are non-deterministically generated in the second stage, which are derived from crucial math facts associated with quantities and relations between quantities. The following FOL function is used to describe the facts about quantities:

$$quan(quan_{id}, unit_{id}, object_{id}) = number \quad (1)$$

For example, "*quan*(*q1*, *#*, *lime*)=*36*" means "*36 limes*" (the second argument is the unit adopted, and ignored here). Besides domain-dependent facts, some auxiliary domain-independent facts associated with the math fact are also created in this stage to help the IE to find the solutions. For example, "*verb*(*q1*, *pick*)&*nsubj*(*q1*, *Fred*)" is the associated auxiliary facts of "*quan*(*q1*, *#*, *lime*)=*36*". Those auxiliary facts are our proposed tags to make the system less sensitive to the irrelevant information. They also provide the flexibility of handling various kinds of possible questions.

The questions in the MWP will be transformed into FOL-like utility functions provided by the IE according to the suggested solution type. Take the question "*How many limes were picked in total?*" as an example. The STC will assign the "*Sum*" operation type to it. Based on that, the LFC will generate the FOL function "*Sum*(*quan*(*?q*, *#*, *limes*), *verb*(*?q*, *pick*)" to search all quantities that are associated with object "*lime*" and also attached with the verb tag "*pick*".

2.4 Logic Inference

The IE is used to find the solution of an MWP. It is responsible for providing utilities to select desired facts and then obtain the answer by taking math operations on those selected facts. In addition, it is also responsible for using inference rules to derive new facts from those facts which are directly derived from the description of the MWP. Consider the example shown in Figure 2, the IE will first select all qualified quantities which match "*quan*(*?q*, *#*, *lime*)" and with a "*pick*" verb tag, and then perform a "*Sum*" operation on them. The irrelevant quantity "*quan*(*q4*, *#*, *pear*)" in that example is thus pruned out as its verb tag is "*drop*", not "*pick*". The answer is then obtained by summing those quantities *q1*, *q2* and *q3*.

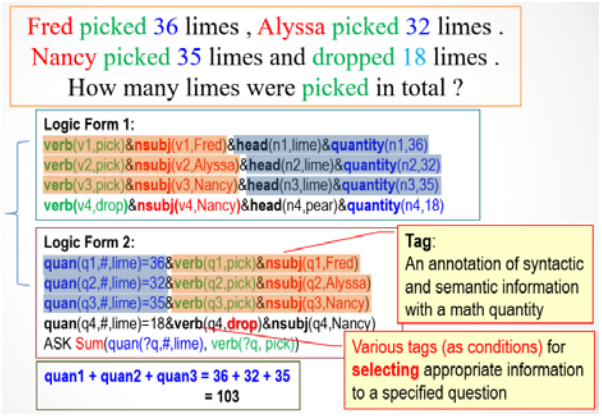

Figure 2: Logic form and logic inference of a *Sum* operation

2.5 Explanation Generation

Based on the reasoning chain generated from the IE (an example is shown in Figure 3), a math operation oriented approach is adopted to explain how the answer is obtained (Huang et al., 2015). A specific template is used to generate the explanation text for each kind of operation. Consider the ex-

ample given in Figure 2, the template for "*Sum*" operation would be "*Totally* ***verb Child_1 + Child_2 + Child_3 + …+ Child_n = Parent***.". According to that template and the reasoning chain shown in Figure 3, The EG will generate the explanation text "*Totally pick 36 limes + 32 limes + 35 limes = 103 limes*", which explains that the obtained answer is a summation of "*36 limes*", "*32 limes*" and "*35 limes*".

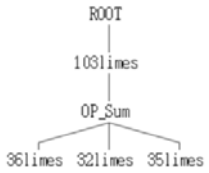

Figure 3: The reasoning Chain from the Inference Engine

3 Demonstration Outline

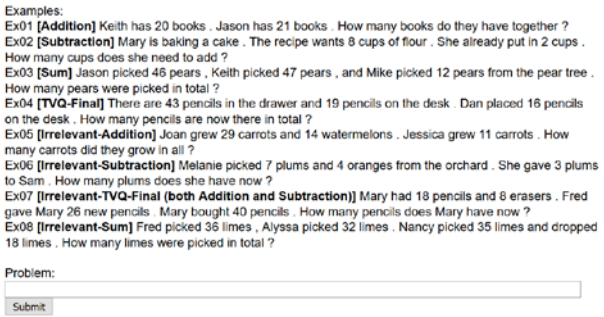

Figure 4: A web interface of the MWP solver

The MWP solver comprises a web user interface and a processing server. The web interface is used to input the problem and display the processing outputs (from each module) of the submitted MWP. The server is responsible to process the submitted problem to get the answer.

The user can use the web interface (Figure 4) to submit various MWPs. After an MWP is submitted, various processing modules will be invoked in a pipelined manner (shown in Figure 1) to solve the given problem. Once the process is finished, the user can browse the outputs generated from each module: (1) Parse Trees, Dependency and Co-Reference chains, which are from the language analyzer. (2) Corresponding linguistic representations, which are converted from the above language analysis result. (3) Suggested solution type, which identifies the desired math operation that the LFC should adopt. (4) Obtained logical forms, which are transformed from the linguistic representation. (5) Generated reasoning chains and explanation text, which explains how the problem is solved.

An online demo can be found at: `http://nlul.iis.sinica.edu.tw/EnglishMathSolver/mathDemo.py`.

4 Experiments

The experiments are performed on the datasets MA1, MA2 and IXL provided by Hosseini et al. (2014), which are the first publically available datasets that can be used to compare various systems. The datasets include 395 problems and 1,483 sentences in total. MA1 covers simple MWPs on addition and subtraction for third, fourth, and fifth graders. Problems in MA2 includes more irrelevant information compared to the other two datasets, and IXL includes more information gaps (Hosseini et al., 2014). The performance of our system is compared with ARIS (Hosseini et al., 2014) which is a rule-based system that changes the entity attribute according to the schema. The result is also compared with KAZB (Kushman et al., 2014), which is a purely statistical approach that aligns the text with various pre-extracted equation templates. We follow the same evaluation setting. Table 2 shows that our system significantly outperforms them in overall performance.

	MA1	IXL	MA2	Total
3-fold Cross validation				
Our System	94.8	71.9	88.4	84.8
ARIS	83.6	75.0	74.4	77.7
KAZB	89.6	51.1	51.2	64.0
Gold Solution Type				
Our System	99.3	97.8	95.0	97.5

Table 2: Performance Comparison

5 Conclusion

A tag-based statistical framework is proposed to perform understanding and reasoning for solving MWPs. The adopted tag can help identify desired operands and filter out irrelevant quantities. Many rule-based approaches only handle addition and subtraction math operations, but we can solve much more problems types, such as Multiplication, Division, Comparison, Algebra, etc.

References

A. Mukherjee, U. Garain. 2008. A review of methods for automatic understanding of natural language mathematical problems, Artif Intell Rev.

D.G. Bobrow. 1964. Natural language input for a computer problem solving system, Ph.D. Dissertation, Massachusetts Institute of Technology.

J.R. Slagle. 1965. Experiments with a deductive question-answering program, J-CACM 8:792-798.

E. Charniak, CARPS. 1968. A program which solves calculus word problems, Report MAC-*TR-51*, Project MAC, MIT.

E. Charniak. 1969. Computer solution of calculus word problems, In Proc. of International Joint Conference on Artificial Intelligence.

D. Dellarosa. 1986. A computer simulation of children's arithmetic word-problem solving, Behavior Research Methods, Instraments, & Computers, 18 147-154.

Y. Bakman. 2007. Robust Understanding of Word Problems With Extraneous Information.

J.P. Gelb. 1971. Experiments with a natural language problem solving system, In Pros. of IJCAI-71.

B. Ballard. A. Biermann. 1979. PROGRAMMING IN NATURAL LANGUAGE : "NLC" AS A PROTOTYPE, ACM-Webinar.

A.W. Biermann, B.W. Ballard 1980. Toward Natural Language Computation American Journal of Computational Linguistic, 6.

A. Biermann, R. Rodman, B. Ballard, T. Betancourt, G. Bilbro, H. Deas, L. Fineman, P. Fink, K. Gilbert, D. Gregory, F. Heidlage. 1982. INTERACTIVE NATURAL LANGUAGE PROBLEM SOLVING:A PRAGMATIC APPROACH In Pros. of the first conference on applied natural language processing.

C.R. Fletcher. 1985. COMPUTER SIMULATION -- Understanding and solving arithmetic word problems: A computer simulation, Behavior Research Methods, Instruments, & Computers, 17 565-571.

M.J. Hosseini, H. Hajishirzi, O. Etzioni, N. Kushman. 2014. Learning to Solve Arithmetic Word Problems with Verb Categorization, EMNLP.

N. Kushman, Y. Artzi, L. Zettlemoyer, R. Barzilay, 2014. Learning to Automatically Solve Algebra Word Problems, ACL.

C. Liguda, T. Pfeiffer, 2012, Modeling Math Word Problems with Augmented Semantic Networks, NLDB.

S.I. Roy, T.J.H. Vieira, D.I. Roth. 2015. Reasoning about Quantities in Natural Language, TACL, 3 1-13.

S. J. Russell and P. Norvig, 2009. Artificial Intelligence: A Modern Approach (3rd Edition), Prentice Hall.

C.D. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. J. Bethard, and D. McClosky. 2014. The Stanford CoreNLP Natural Language Processing Toolkit In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations, pp. 55-60.

C.-C. Chang, C.-J. Lin. 2011. LIBSVM: A library for support vector machines. ACM Transactions on Intelligent Systems and Technology.

Y.-C. Lin, C.-C. Liang, K.-Y. Hsu, C.-T. Huang, S.-Y. Miao, W.-Y. Ma, L.-W. Ku, C.-J. Liau, K.-Y. Su. 2015. Designing a Tag-Based Statistical Math Word Problem Solver with Reasoning and Explanation, International Journal of Computational Linguistics and Chinese Language Processing, 20(2), 1-26.

C.-T. Huang, Y.-C. Lin, K.-Y. Su. 2015. Explanation Generation for a Math Word Problem Solver, International Journal of Computational Linguistics and Chinese Language Processing, 20(2), 27-44.