Using Intermediate Representations to Solve Math Word Problems: Supplementary Material

Danqing Huang[1,*], **Jin-Ge Yao**[2], **Chin-Yew Lin**[2], **Qingyu Zhou**[3], **and Jian Yin**[1]
{huangdq2@mail2,issjyin@mail}.sysu.edu.cn
{Jinge.Yao,cyl}@microsoft.com
qyzhou@hit.edu.cn
[1] The School of Data and Computer Science, Sun Yat-sen University.
Guangdong Key Laboratory of Big Data Analysis and Processing, Guangzhou, P.R.China
[2]Microsoft Research [3] Harbin Institute of Technology

Abstract

This note contains supplementary materials to *Using Intermediate Representations to Solve Math Word Problems*.

1 Definition of Intermediate Forms

For our intermediate forms, we define six classes and The definitions of twenty-three functions. They are shown in Table S1 and Table S2 respectively.

2 Rules for Deriving Intermediate Forms

In this section, we describe the derivation process of our intermediate forms.

Table S3 lists the rules that we use to derive intermediate forms from equation systems. For example, we use regular expressions to match variables (*num, int, unk*).

With the rules in Table S3, we map equation systems to intermediate forms based on operator precedence: (1) variables (e.g. num, math#consecutive, middle()); (2) functions of exponents and roots (e.g. math#square); (3) functions of multiplication and division (e.g. math#opposite, math#product); (4) functions of addition and subtraction (e.g. math#more, math#less). We show the derivation process of two math problems in Table S4.

3 Error Analysis

There are two main types of errors produced by our model: (1) Natural language variations; (2) Nested operations. In Table S5, we show some example problems in each category.

* Work done while this author was an intern at Microsoft Research.

Class	Example	Semantic
int	1, 2, 3	integers
float	1.1, 3.14	floating numbers
num	1, 3.14	parent class of *int* and *float*
unk	m, n	unknown variables
var	3.14, m	parent class of *num* and *unk*
list	[2, 3, m]	a list elements of type *var*

Table S1: Classes in our intermediate representation.

Function	Parameters	Return	Semantic	Expression
math#reciprocal	$1: var	var	the reciprocal of a variable	1/$1
math#oppsotite	$1: var	var	the opposite of a variable	(-1)*$1
math#sum	$1: list	var	the sum of the elements in the list	list[0]+...
math#sum	$1: int	var	the sum of $1 unknown variables	m+n+...
math#diff	$1,$2: var	var	the difference of $1 and $2	$2-$1
math#product	$1: list	var	the product of the elements in the list	list[0]*...
math#product	$1: int	var	the sum of $1 unknown variables	m*n*...
math#more	$1: var, $2:num	var	$1 is $2 more than return variable	$1+$2
math#less	$1: var, $2:num	var	$1 is $2 less than return variable	$1-$2
math#quotient	$1,$2: var	var	the quotient of $1 and $2	$1/$2
math#square	$1: var	var	the square of a variable	$(\$1)^2$
math#pow	$1: var, $2: int	var	base $1 raised to the power exponent $2	$1^$2
math#consecutive	$1: int	list	a list of $1 consecutive variables	[x, x+1,...]
math#odd	$1: int	list	a list of $1 consecutive odd variables	[2*x-1,...]
math#even	$1: int	list	a list of $1 consecutive even variables	[2*x,...]
count	$1: list	int	the size of the list	-
max	$1: list	var	variable with maximum value in the list	-
min	$1: list	var	variable with minimum value in the list	-
middle	$1: list	var	variable with middle value in the list	-
next	$1: list, $2: var	var	variable in the list after $2	-
ordinal	$1: list, $2: int	var	element in the list with value ranked in $2	-
math#digit	$1: int	var	variable with $1 digits	x+10*y+...
=	$1,$2:var	-	$1 is equal to $2	$1=$2

Table S2: Functions in our intermediate representation.

Regex/Rules	Class/Function
\-?[0-9\.]+	num
\-?[0-9]+	int
[a-z]	unk
<num>|<unk>	var
\(\-1\)*<var>	math#opposite($1:var)
1/<var>	math#reciprocal($1:var)
(<var>\+)+<var>	math#sum($1:list)
(<unk>\+)+<unk>, $1=count of unk in the match	math#sum(count:$1:int)
<var>\-<var>	math#diff($1,$2:var)
(<var>*)+<var>	math#prodcut($1:list)
(<unk>*)+<unk>, $1=count of unk in the match	math#prodcut(count:$1:int)
<var>\+<num>	math#more($1:var, $2:num)
<var>\-<num>	math#less($1:var, $2:num)
<var>/<var>	math#quotient($1,$2:var)
<var>\^2|<var>*<var>	math#square($1:var)
<var>\^<int>	math#pow($1:var, $2:int)
if there are x+10*y+100*z+... in the equation system	math#digit($1:int)
if there are *unk*, *unk*+1, ... in the equation system	math#consecutive($1:int)
if there are 2**unk*-1, 2**unk*+1, ... in the equation system	math#odd($1:int)
if there are 2**unk*, 2**unk*+2,... in the equation system	math#even($1:int)
if *var* has max value in $1	max($1:list)
if *var* min max value in $1	min($1:list)
if *var* has middle value in $1	middle($1:list)
if *var* has the value ranked $2 in $1	ordinal($1:list, $2:int)

Table S3: Rules for deriving intermediate form candidates from equations.

Problem 1: *Phil found that the sum of twice a number and -21 is 129 greater than the opposite of the number, What is the number?*
Equation system: $2 * m + (-21) = 129 + ((-1) * m)$
Derivation:
⇒ **math#product(2, m)** + (-21) = 129 + **math#opposite(m)**
⇒ **math#sum**(math#product(2, m), -21) = **math#sum**(129, math#opposite(m))

Problem 2: *The sum of 3 integers is 251. The sum of the 1st and 2nd integers exceeds the 3rd by 45. The 3rd integer is 42 less than the 1st. Find the 3 integers.*
Equation system: $m + n + o = 251, m + n = o + 45, o = m - 42$
Derivation:
$m + n + o = 251$ ⇒ math#sum(cnt: 3) = 251
or math#sum(m, n, o) = 251
or math#sum(ordinal(1), ordinal(2), ordinal(3)) = 251
$m + n = o + 45$ ⇒ math#sum(m, n) = math#sum(o, 45)
or math#sum(oridnal(1), ordinal(2)), math#sum(ordianl(3), 45)
or math#sum(min(), middle()), math#sum(max(), 45)
$o = m - 42$ ⇒ o = math#diff(m, 42)
or ordinal(3)=math#diff(ordianl(1), 42)
or max()=math#diff(min(), 42)
(combination of the three equations)

Table S4: Examples in intermediate form derivations.

Error Type 1: Natural language variations
Problem 1 Two pieces of equipment were purchased for a total of $9000. If one piece cost $370 more than the other, find the price of the less expensive piece of equipment. **Problem 2** A new oil tank holds 75 barrels of oil more than an old tank. Together they hold 515 barrels of oil. How much will each tank hold? **[Explanation]** Problem 1 and 2 belongs to the same type of problems, with the template $x + y = n_1, x - y = n_2$ **Problem 3** A landscaping company charges $100 plus $15 per hour. Another company charges $75 plus $17 per hour. How long is a job that costs the same no matter which company is used? **Problem 4** To deliver mulch, lawn and garden charges $30 per cubic yard of mulch plus a $30 delivery fee. Yard depot charges $25 per cubic yard of mulch plus a $55 delivery fee. For how many cubic yards will the cost be the same? **[Explanation]** Problem 3 and 4 belongs to the same type of problems. with the template $n_1 * x + n_2 = n_3 * x + n_4$

Error Type 2: Nested operations
Problem 5 I think of a number, double it, add three, multiply the answer by three and then add on twice the number I first thought of. If the final answer is 145 what was the number I first thought of? **[Explanation]** step 1: double a number $\rightarrow 2 * x$ step 2: add three $\rightarrow 2 * x + 3$ step 3: multiple the answer by three $\rightarrow 3 * (2 * x + 3)$ step 4: add twice the number $\rightarrow 3 * (2 * x + 3) + 2 * x$ step 5: final answer is 145 $\rightarrow 3 * (2 * x + 3) + 2 * x = 145$ **Problem 6** The difference between two numbers is 10. If the numbers are doubled, what is the difference between them? **[Explanation]** step 1: difference of two numbers $\rightarrow x - y = 10$ step 2: double the numbers $\rightarrow 2 * x, 2 * y$ step 3: calculate the new difference $\rightarrow 2 * x - 2 * y$ **Problem 7** find three consecutive integers such that three times the middle integer is equal to the sum of five times the first and three times the third integer decreased by 13. **[Explanation]** step 1: three times the middle integer $\rightarrow 3 * (x + 1)$ step 2: sum of five times the first and three times the third integer $\rightarrow 5 * x + 3 * (x + 2)$ step 3: decreased by 13 $\rightarrow 5 * x + 3 * (x + 2) - 13$

Table S5: Example error problems produced by our model.