Computational Linguistics and Chinese Language Processing
Vol. 20, No. 2, December 2015, pp. 27-44

Explanation Generation for a Math Word Problem Solver

Chien-Tsung Huang[*], Yi-Chung Lin[*] and Keh-Yih Su[*]

Abstract

This paper proposes a math operation (e.g., *Summation*, *Addition*, *Subtraction*, *Multiplication*, *Division*, etc.) oriented approach to explain how the answers are obtained for *math word problems*. Based on the reasoning chain given by the inference engine, we search each math operator involved. For each math operator, we generate one sentence. Since explaining math operation does not require complicated syntax, we adopt a specific template to generate the text for each kind of math operator. To the best of our knowledge, this is the first explanation generation that is specifically tailored to solving the math word problem.

Keywords: Explanation Generation, Math Word Problem Explanation, Machine Reading

1. Introduction

Since *Big Data* mainly aims to explore the correlation between surface features but not their underlying causality relationship (Mayer-Schönberger & Cukier, 2013), the "*Big Mechanism*" program[1] has been proposed by DARPA to find out "why" behind the big data. However, the pre-requisite for it is that the machine can read each document and learn its associated knowledge, which is the task of *Machine Reading* (MR) (Strassel *et al.*, 2010). Therefore, the Natural Language and Knowledge Processing Group (under the Institute of Information Science) of Academia Sinica formally launched a 3-year MR project (from January 2015) to attack this problem.

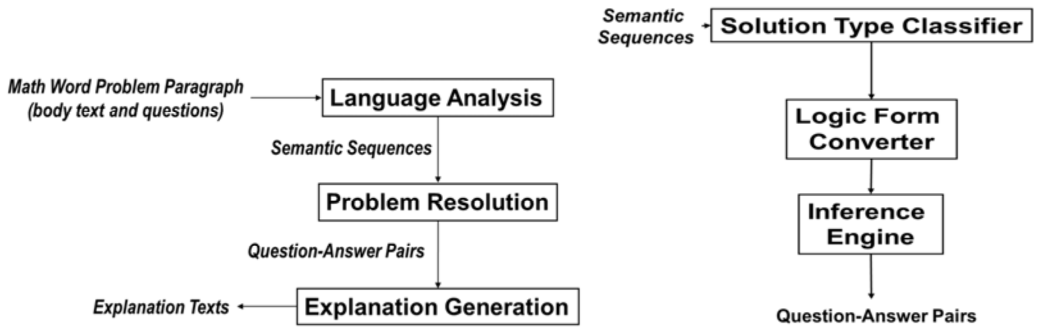
Since a domain-independent MR system is difficult to build, the *Math Word Problem* (MWP) (Mukherjee & Garain, 2008) is chosen as our first test case to study MR. The main reason for that is that it not only adopts less complicated syntax but also requires less amount of domain knowledge; therefore, the researcher can focus more on text understanding and

[*] Institute of Information Science , Academia Sinica
128 Academia Road, Section 2, Nankang, Taipei 11529, Taiwan
E-mail: { joecth; lyc; kysu}@iis.sinica.edu.tw

[1] http://www.darpa.mil/Our_Work/I2O/Programs/Big_Mechanism.aspx

reasoning (instead of looking for a wide coverage parser and acquiring considerable amount of domain knowledge). We thus also choose it as the goal of the first year for studying the MR problem, and propose a tag-based statistical approach (Lin *et al*., 2015) to find out the answer.

The architecture of this proposed approach is shown in Figure 1. First, every sentence in the MWP, including both body text and the question text, is analyzed by the *Language Analysis* module, which transforms each sentence into its corresponding semantic representation tree. The sequence of semantic representation trees is then sent to the *Problem Resolution* module, which adopts logic inference approach, to obtain the answer of each question in the MWP. Finally, the *Explanation Generation* module will explain how the answer is found (in natural language text) according to the given *reasoning chain* (Russell & Norvig, 2009) (which includes all related logic statements and inference steps to reach the answer).



(a) Math Word Problem Solver Diagram (b) Problem Resolution Diagram

Figure 1. The block diagram of the proposed Math Word Problem Solver.

As depicted in Figure 1(b), the *Problem Resolution* module in the proposed system consists of three components: *Solution Type Classifier* (TC), *Logic Form Converter* (LFC) and *Inference Engine* (IE). The TC is responsible to assign a math operation type for every question of the MWP. In order to perform logic inference, the LFC first extracts the related facts from the given semantic representation tree and then represents them in *First Order Logic* (FOL) *predicates/functions* form (Russell & Norvig, 2009). In addition, it is also responsible for transforming every question into an FOL-like utility function according to the assigned solution type. Finally, according to inference rules, the IE derives new facts from the old ones provided by the LFC. Besides, it is also responsible for providing utilities to perform math operations on related facts.

In addition to understanding the given text and then performing inference on it, a very desirable characteristic of an MWP solver (also an MR system) is being able to explain how the answer is obtained in a human comprehensible way. This task is done by the *Explanation*

Generator (EG) module, which is responsible to explaining the associated reasoning steps in fluent natural language from the given *reasoning chain* (Russell & Norvig, 2009). In other words, explanation generation is the process of constructing natural language outputs from a non-linguistic input, and is a task of *Natural Language Generation* (NLG).

Various applications of NLG (such as weather report) have been proposed before (Halliday, 1985; Goldberg *et al.*, 1994; Paris & Vander Linden, 1996; Milosavljevic, 1997; Paris *et al.*, 1998; Coch, 1998; Reiter *et al.*, 1999). However, to the best of our knowledge, none of them discusses how to generate the explanation for WMP, which possesses some special characteristics (e.g., math operation oriented description) that are not shared with other tasks.

A typical architecture for NLG is shown at Figure 2, which is re-drawn from Jurafsky and Martin (Jurafsky & Martin, 2000). Under this architecture, *Communicative Goal*, which specifies the purpose for communication, and *Knowledge Base*, which specifies the content to be generated, are fed as the inputs to *Discourse Planner*. The Discourse Planner will then output a hierarchy form to the *Surface Realizer*, which further solves the issues of selecting lexicons, functional words, lexicon order in the sentence, syntactic form, subject-verb agreement (mainly required for English), tense (mainly required for English), and so on for the texts to be generated.

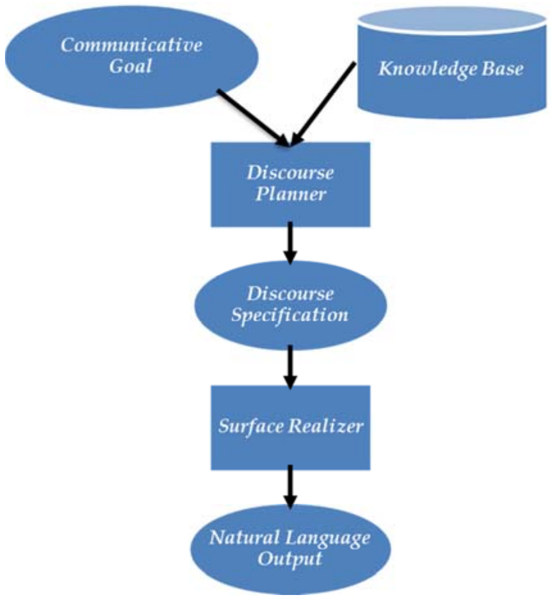


Figure 2. A typical architecture for NLG systems (Jurafsky & Martin, 2000)

To implement the Discourse Planner, D. Jurafsky (Jurafsky & Martin, 2000) proposed to adopt text schemata and rhetorical structure planning to implement the Discourse Planner. On the other hand, Kay proposed to implement the Surface Realizer with both Systemic Grammar, which is a part of Systemic Functional Linguistic proposed by Halliday (Halliday, 1985), and Functional Unification Grammar (Kay, 1979).

Since the description for math operation centering on an operator is in a relatively fixed textual format, which is disparate from other kinds of NLG tasks, those approaches mentioned above might be over-killed for the task of MWP explanation generation (and thus introduce unnecessary complexity). Therefore, we propose an operator oriented approach to search each math operator involved in the reasoning chain. For each math operator, we generate one sentence. Since explaining math operation does not require complicated syntax, a specific template is adopted to generate the text for each kind of math operator. To the best of our knowledge, this is the first approach that is specifically tailored to the MWP task.

Our main contributions are listed as following,

1. We proposed a *math operation oriented Explanation Tree* for facilitating the discourse work on MWP.
2. We propose an *operator oriented algorithm* to segment the Explanation Tree into various sentences, which makes our Discourse Planner universal for MWP and independent to the language adopted.
3. We propose using *operator-based templates* to generate the natural language text for explaining the associated math operation.

The remainder of this paper is organized as follows: Section 2 introduces the framework of our Explanation Generator. Afterwards, various templates of more operators (other than SUM used in Section 2) are introduced in Section 3. Section 4 discusses the future work of our explanation system. Section 5 then reviews the related works. Finally, the conclusions are drawn in Section 6.

2. Proposed Framework for MWP Explanation Generator (EG)

Figure 3 shows the block diagram of our proposed EG. First, the Inference Engine generates the answer and its associated reasoning chain for the given MWP. First, to ease the operation of the EG, we convert the given reasoning chain into its corresponding *Explanation Tree* (shown at Figure 5) to center on each operator appearing in the reasoning chain (such that it is convenient to perform sentence segmentation later). Next, the Explanation Tree will be fed as input to the *Discourse Planner*, which divides the given Explanation Tree into various subtrees such that each subtree will generate one explanation sentence later. Finally, the *Function Word Insertion & Ordering Module* will insert the necessary functional words and order them with those extracted content words (from the segmented Explanation Subtee) to generate the *Explanation Texts*.

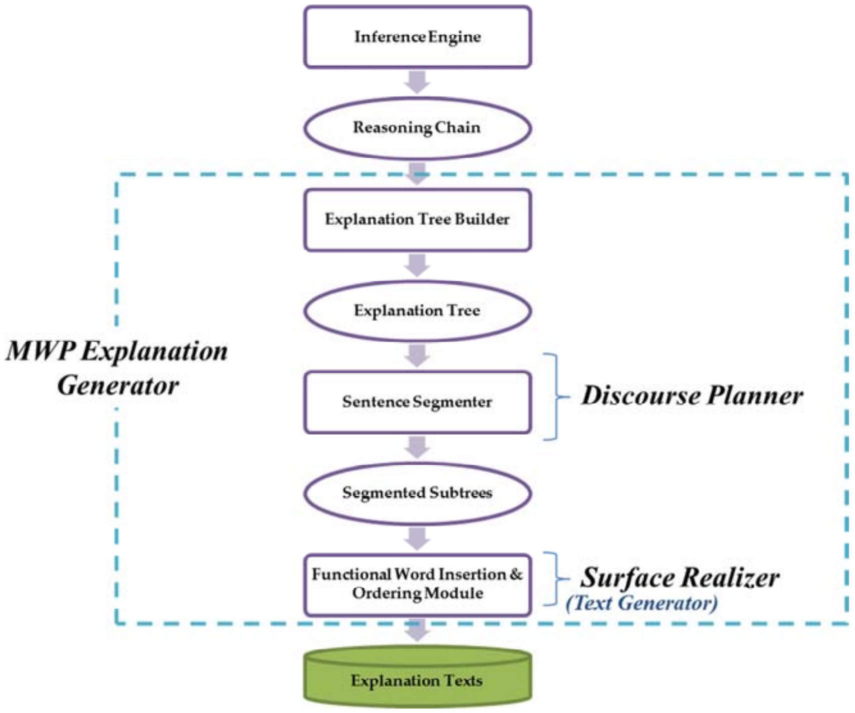


Figure 3. Block Diagram of the proposed MWP Explanation Generator

Following example demonstrates how the framework works. And Figure 4 (a) reveals more details for each part illustrated in Figure 3.

[Sample-1] 阿志買一臺冰箱和一臺電視機，付 2 疊一萬元鈔票、6 張千元鈔票和 13 張百元鈔票，阿志共付了幾元？

(A-Zhi bought a refrigerator and a TV. He paid 2 stacks of ten-thousand-dollar bill, six thousand-dollar bills and 13 hundred-dollar bills. How many dollars did A-Zhi pay in total?)

Facts Generation in Figure 4(a) shows how the body text is transformed into meaningful logic facts to perform inference. In math problems, the facts are mostly related to quantities. The generated facts are either the quantities explicitly appearing in the sentence of the problem or the implicit quantities deduced by the IE. Those generated facts are linked together within the reasoning chain constructed by the IE as shown in Figure 4(b). Within this framework, the discourse planner is responsible for selecting the associated content for each sentence to be generated.

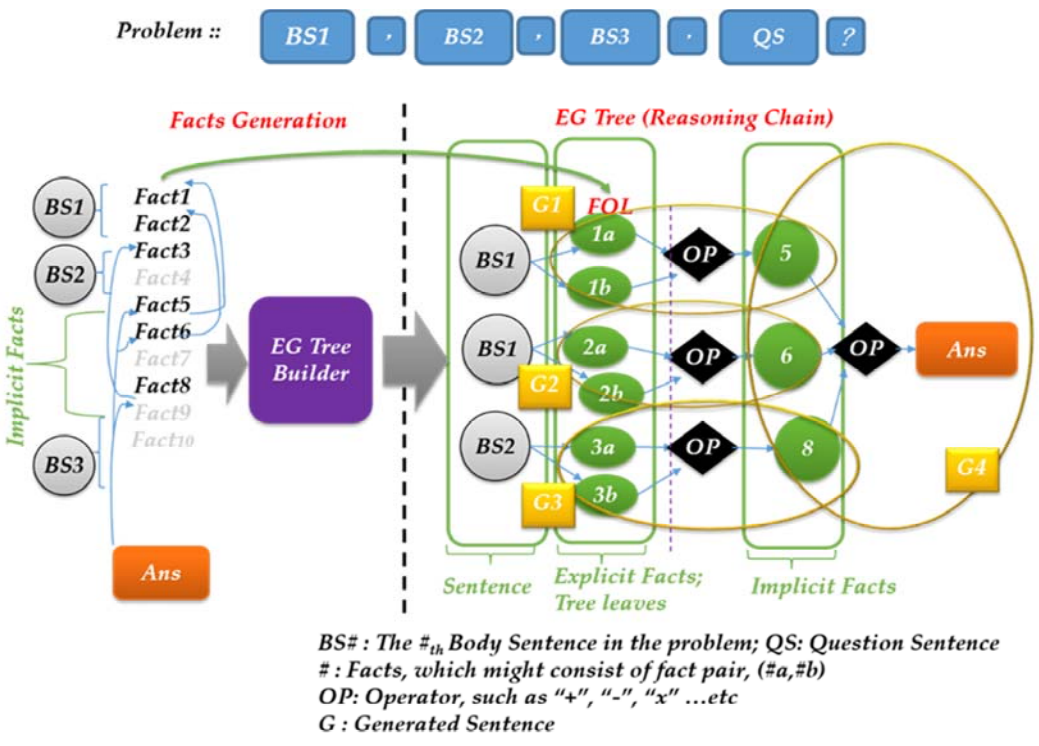

Problem ::
BS1
,
BS2
,
BS3
.
QS
?
Facts Generation
EG Tree (Reasoning Chain)
BS1
BS2
BS3
Fact1
Fact2
Fact3
Fact4
Fact5
Fact6
Fact7
Fact8
Fact9
Fact10
Implicit Facts
Ans
EG Tree Builder
G1
FOL
1a
1b
2a
2b
3a
3b
G2
G3
G4
OP
5
6
8
Ans
Sentence
Explicit Facts; Tree leaves
Implicit Facts
BS# : The #th Body Sentence in the problem; QS: Question Sentence
# : Facts, which might consist of fact pair, (#a,#b)
OP: Operator, such as "+", "-", "x" ...etc
G : Generated Sentence

Figure 4(a). Facts Generation and EG Tree Builder

Figure 4(b). Reasoning Chain (represented as an Explanation Tree for illustration)

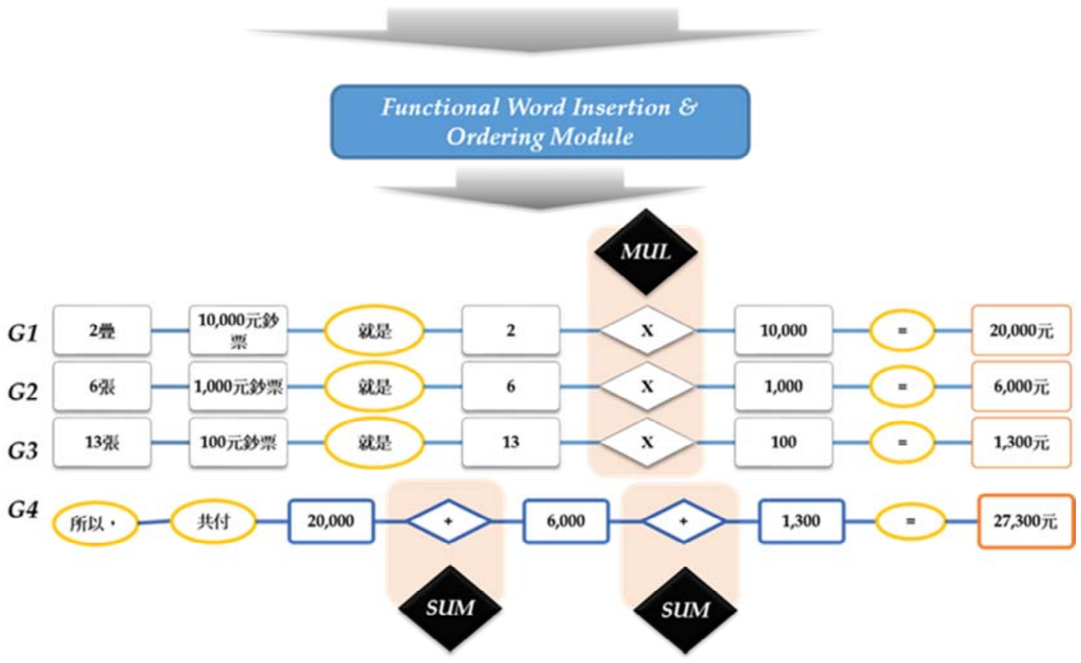


Figure 4(c). Function Word Insertion & Ordering Module, serving as the Surface Realizer. It shows how surface realization is done with pre-specified function words (circled by ellipses) and extracted slot-fillers (enclosed by diamond for operator, and rectangle for quantities).

Figure 4. (a) Facts Generated from the Body Text. (b) The associated Reasoning Chain, where "G#" shows the facts grouped within the same sentence. (c) Explanation texts generated by the TG for this example (labeled as G1~G4). Except those ellipses which symbolize pre-specified function words, other shapes denote extracted slot-fillers. Furthermore, Diamond symbolizes OP_node while Rectangle symbolizes Quan_node.

A typical reasoning chain, represented with an Explanation Tree structure, is shown at Figure 4(b). The *operator-node* (OP_node) layers and *quantity-node* (Quan_node) layers are interleaved within the Explanation Tree, and serving as the input data structure to *OP Oriented Algorithm* in Discourse Planner, which will be further presented as pseudo code in Section 2.2 *(Algorithm 1)*. As shown at Figure 4(b), the (#a, #b) pair denotes facts derived from the body sentences. The OP means the operator used to deduce implicit facts and represented as non-leaf circle nodes. Each "*G?*" expresses a sentence to be generated. Given the reasoning chain, the first step is to decide how many sentences will be generated, which corresponds to the *Discourse Planning phase* (Jurafsky & Martin, 2000) of the traditional NLG task. Currently, we will generate one sentence for each operator shown in the reasoning chain. For the above example, since there are four operators (three IE-Multiplications[2] and one LFC-Sum), we will have four corresponding sentences; and the associated nodes (i.e., content) are circled by "*G?*" for each sentence in the figure.

Furthermore, Figure 5 shows that three sets of facts are originated from the 2nd body sentence (indicated by three *S2* nodes). Each set contains a corresponding quantity-fact (i.e., *q1*(疊), *q2*(張), and *q3*(張)) and its associated object (i.e., *n1*, *n2*, and *n3*). For example, the first set (the left most one) contains *q1*(疊) (for "2 疊") and *n1* (for "一萬元鈔票"). This figure also shows that the outputs of three IE-Multiplication operators (i.e., "20,000 元", "6,000 元", and "1,300 元") will be fed into the last LFC-Sum to get the final desired result "27,300 元" (denoted by the "Ans(SUM)" node in the figure).

After having given the corresponding content (associated with those nodes within the big circle), we need to generate the corresponding sentence with appropriate function words added. This step corresponds to the *Surface Realization* phase (Jurafsky & Martin, 2000) in NLG. Currently, since the syntax of the explanation text of our task is not complicated, we use various templates to take into account the pre-specified fillers ("⬭") and the slots to be filled ("▭" and "◇") and their order for generating the desired explanation sentence. Figure 4(c) shows how a sentence is generated from a selected template based on the given Explanation Tree.

[2] Prefixes "IE-" and "LFC-" denote that those operators are issued by IE and LFC, respectively.

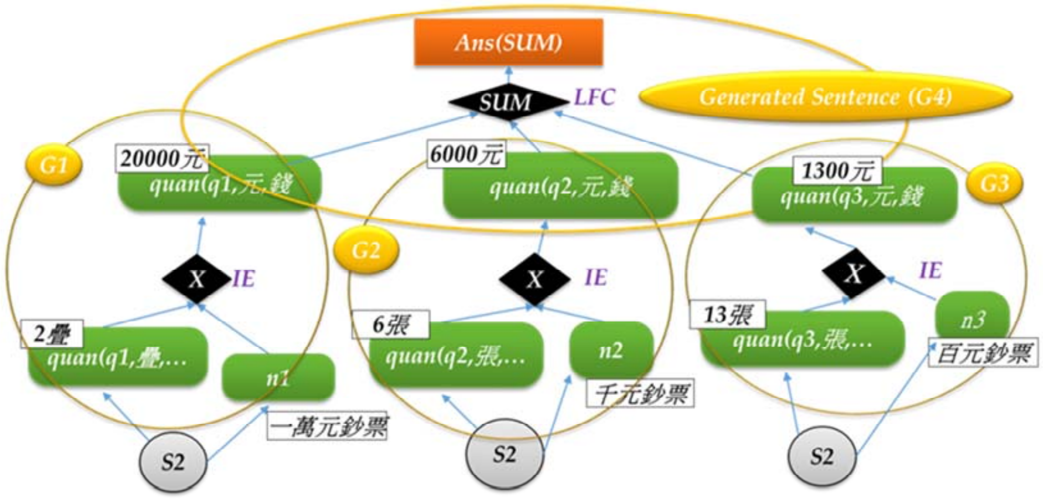


Figure 5. Explanation Tree for Discourse Planning, where S2 means that those facts are from the 2nd body sentence.

2.1 Explanation Tree Builder

The original reasoning chain resulted from the IE is actually a stream of chunks (as shown in Figure 4(a)), in which the causal chain is *implicitly* embedded. Therefore, it is not suitable for explaining inference steps. The *Explanation Tree Builder* is thus adopted to build up the *Explanation Tree*, which centers on the math operations involved in the inference process, to *explicitly* express the causal chain implied.

The Explanation Tree Builder first receives various facts, as a stream of chunks, from the IE. It then creates the nodes of the Explanation Tree according to the content of those chunks. After the Explanation Tree is created, it serves as the corresponding reasoning chain for the following process since then.

With the *root node* serving as the Answer, which is a Quan_node, the Explanation Tree is interleaved with Quan_node layers and OP_node layers, as shown in Figure 4(b). Each OP_node has one Quan_node as its parent node, and has at least one Quan_node as it's child node. On the other hand, each Quan_node (except the root node) serves as the input to an OP_node. With the Explanation Tree, the work of discourse planning can be simply done via traversing those OP_nodes, which will be described in the following section.

2.2 Sentence Segmenter (Discourse Planner)

In NLG, the discourse planner selects the content from the knowledge base according to what should be presented in the output text, and then structures them coherently. To facilitate the explanation process, we first convert the given reasoning chain to its corresponding

Explanation Tree, as shown at Figure 4(b) to ease the following operations. The Explanation Tree is adopted because its structure allows us to regard the OP as a basis to do sentence segmentation for the deductive steps adopted in MWP. Within the Explanation Tree, the layers of OP nodes are interleaved with the layers of quantity nodes, and the root-node is the quantity node which denotes the desired Answer.

Algorithm 1: OP_Oriented_ExplanationGenerator

```
Input : (i) Directed Tree g = (V, E);        where V are either OP_node or Quan_node
            (ii) source node root;
        Every node records node_number & depth as it's functional data member
        Besides,
        OP_node records operators as its content data member
        Quan_node records values as its content data member
Output : Sequence of Explanation Sentences

Initialize L to an empty List;
Initialize ExpSet to an empty List.

for each vertex j ∈ V do
    if j ∈ OP_node
        L.EnList(j) /* Add j into L list */
    else   /* j ∈ Value node */
        pass

while L is not empty do
    l = DeList(L) /* pop-out the node with largest depth;*/
                  /* if more than one, the one with smallest number is selected */
    s = FunctionWordInsertionAndOrderingModule(l)     /* Algorithm 2*/
    ExpSet.EnList(s) /* Add s into ExpSet list */

Output(ExpSet)
```

After having constructed the Explanation Tree, we need to know how to group the nodes within the tree to make a sentence. As one can imagine, there are various ways to combine different quantities and operators (within the Explanation Tree) into a sentence: you can either explain several operations within one complicated sentence, or explain those operations with several simple sentences. Discourse planner therefore controls the process for generating the discourse structure, which mainly decides how to group various Explanation Tree nodes into different discourse segments. The proposed *OP Oriented Algorithm*, as shown above, is introduced to organize various Explanation Tree nodes into different groups (each of them will correspond to a sentence to be generated). Basically, it first locates the lowest operation node, and then traverses each operation node from *left to right* (with the same parent node) and *bottom to top*. For each operation node found, it will group the related nodes around that operation node into one discourse segment (i.e., one sentence). For each group, it will call the *Surface Realizer* module to generate the final sentence. It is named "*OP oriented*" because every generated sentence in the explanation text is based on one operator, which serves as a central hub to associate all quantities directly linked with it. Also, the template for building up a sentence is selected based on the associated operator, which will be further introduced in Section 2.3.

Figure 6 shows three grouped explanation subtrees within the original explanation tree. The arrows between SUM node and its children show the sequence of those subtrees to be presented, and the numbers imposed on tree nodes indicate the indexes of the corresponding sentence to be generated.

2.3 Function Word Insertion and Ordering Module (Surface Realizer)

The sentence segmenter module discussed previously only partitions the explanation tree into various Explanation Subtrees. It has no control over how the components within an explanation subtree should be positioned. Also, we frequently need to insert extra functional words (sometimes even verbs) such as "就是"、"共是"、"等同於" ("are", "equal", "mean") and the like to have a fluent sentence. For example, in Sample-1, to explain what "2 疊一萬元" (2 stacks of 10-thousand-dollar bill) means, we need an extra functional word "就是" ("are") (or "共是"、"等同於" ("equal", "mean") and the like) to make the sentence readable. Furthermore, people prefer to add "所以" ("Thus"), to explicitly hint that the following text is closely related to the answer.

Since the syntax for explaining math operation is not complicated, we adopt the template approach to accomplish both tasks mentioned above in the same time. Currently, for each math operator, a corresponding template is manually created, which contains various slots that will be filled with contents from the nodes in Explanation Tree.

Figure 6 shows the connection between a template and its associated Explanation Tree for Sample-1. It comprises three kinds of nodes: the answer-node (shown by the rectangle □) which denotes the final answer and is basically a Quan_node; the OP_nodes (shown by the diamond ◇) which denote associated operators; and the quantity-nodes (shown by the rounded-corner rectangle ▢) which represent the values extracted by the LFC or inferred by the IE.

Take the last explanation sentence of the above sample 1 as an example,

所以，共付了 20000 + 6000 + 1300 = 27300 元

Since its associated operator is "SUM", the template of "SUM" is first fetched and there are four slots to be filled. The arrow then directs the flow to ① for "20,000" to be printed out and then SUM for the "+". Next on, the flow is directed to the middle child node, ②, and "6,000" is therefore outputted as the subsequent component in this sentence, and then it directs back to SUM again to print "+". Finally, the flow directs to the most right-hand-side node, ③, then goes back to SUM; the "1,300" is then popped out accordingly. We don't print out the "+" for the SUM this time since we know there's no more child node below the SUM node that hasn't been traversed. After all the child nodes are traversed and their contents are copied into the associated slots, the parent node, ④, is traversed and the text "=27,300 元" is printed out to complete the explanation sentence.

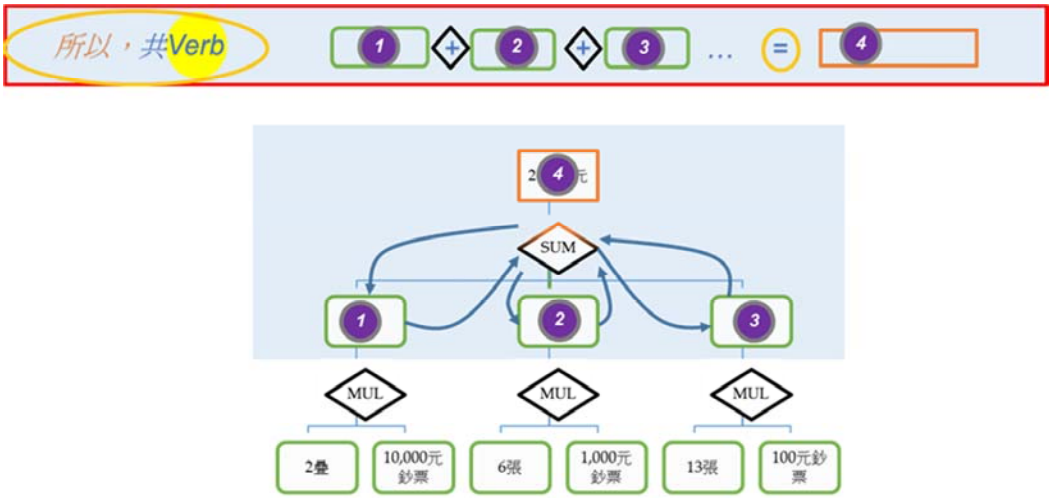


Figure 6(a). Surface Realizer – OP_SUM template

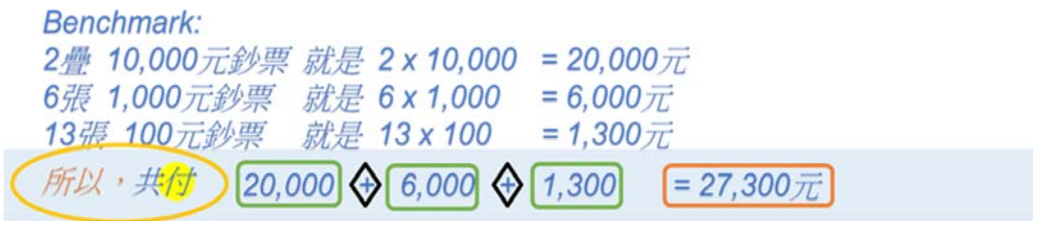

Benchmark:
2疊 10,000元鈔票 就是 2 x 10,000 = 20,000元
6張 1,000元鈔票 就是 6 x 1,000 = 6,000元
13張 100元鈔票 就是 13 x 100 = 1,300元
所以，共付 20,000 + 6,000 + 1,300 = 27,300元

Figure 6(b). Benchmark for the output of Surface Realizer

Figure 6. The template for OP_SUM ("SUM" in Figure 6 (a), and the explanation sentences for Sample-1)

Algorithm 2 shows the *Function Word Insertion and Ordering* algorithm, which illustrates how the surface realizer is implemented. After the list S is initiated at Line 4, the operation type of the OP_node is checked at Line 7 to select a corresponding template, which is assigned to OPtemplate at Line 8 (each kind of operator has its own template). Take Sample-1 for example, the template shown in Figure 6 (a) is selected for the “SUM” operator. Following the “Arrow” notation mentioned above, contents of the OP_node and its connecting nodes are put into List S at Line 9. Later on, the nodes in List S are filled into the template described above at Line 10, which corresponds to the Benchmark shown in Figure 6(b). Finally, at Line 12, the slots of OPtemplate are all filled with appropriate contents. It then returns them as an explanation sentence string.

Algorithm 2: FunctionWordInsertionAndOrderingModule

```
     Input: (i) Directed Tree g=(V,E); where V are either OP_node or Quan_node
               (ii) one specific node v ∈ V
     Output: One sentence string instantiated with components of neighboring nodes of an OP_node

1    if v.type != OP_node      /* Quan_node is returned here */
2        return NULL
3
4    Initial S to an empty list for a generated sentence
5
6    /* Select the template for Surface Realizer according to the type of operator */
7    switch(v.content)      /* for OP_node, v.content shows what kind of operator the OP_node is */
8        OPtemplate = the specified template for the OP_node
9        S.EnList(the contents of “v”, its children, and its parent)
10       Fill contents of nodes in S into the OPtemplate
11
12   return OPtemplate as a String to represent this sentence
```

Since each question will be processed separately and a reasoning chain will be associated with only one question, there is no restriction for the number of allowable question sentences (as the proposed algorithm only handles one reasoning chain each time).

3. Some Other Associated Templates

As described in the previous section, the template adopted is closely related to the associated math operation. However, various templates share a meta-form with some common characteristics:

(1) *Each operator generates a sentence.*

(2) *Each sentence is generated from the operator and the quantities connected to it.*

(3) *The operators and the quantities are inserted into the slots specified in the template.*

(4) *The instantiated template serves as the corresponding explanation sentence string.*

Apart from the OP_SUM, this section introduces a few other templates associated with OP_MUL, OP_COMMON_DIVISION, and OP_UNIT_TRANS as follows. OP_MUL is related to Sample-1 mentioned above (Figure 7). OP_COMMON_DIV is associated with Sample-2 (Figure 8). Also, Figure 9 shows the template associated with "OP_UNIT_TRANS" adopted in Sample-3.

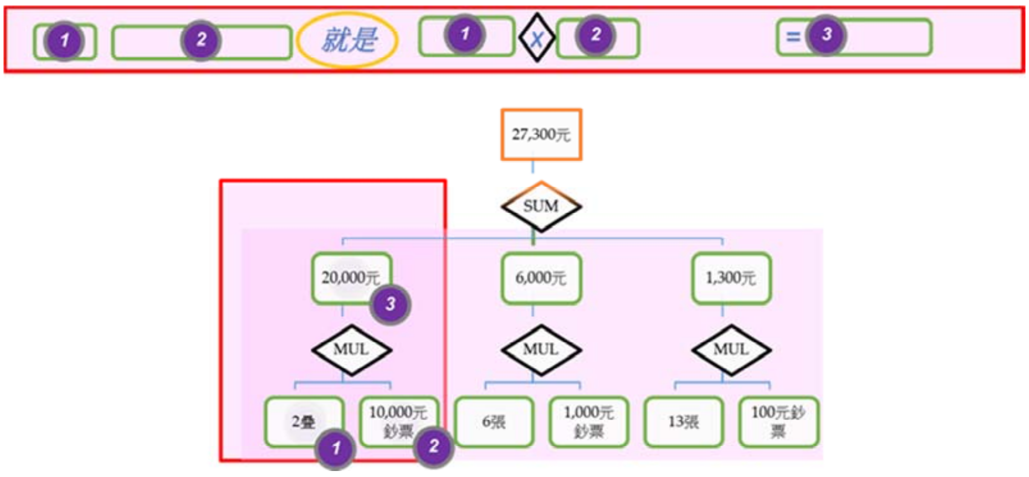


Figure 7(a)

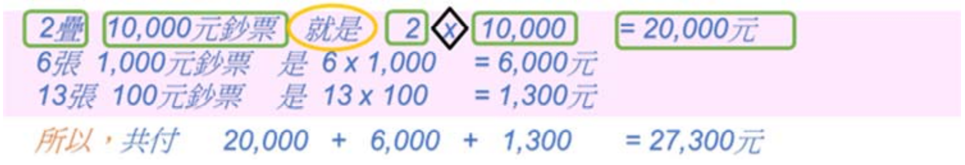

2疊 10,000元鈔票 就是 2 x 10,000 = 20,000元
6張 1,000元鈔票 是 6 x 1,000 = 6,000元
13張 100元鈔票 是 13 x 100 = 1,300元
所以，共付 20,000 + 6,000 + 1,300 = 27,300元

Figure 7(b)

Figure 7. The template for OP_MUL ("MUL" in Figure 7 (a)) and the explanation sentences for Sample-1.

[Sample-2] 1 個平年有 365 天，3 個平年共有幾天？

(One common-year (non-leap year) has 365 days. How many days do 3 common-year have?)

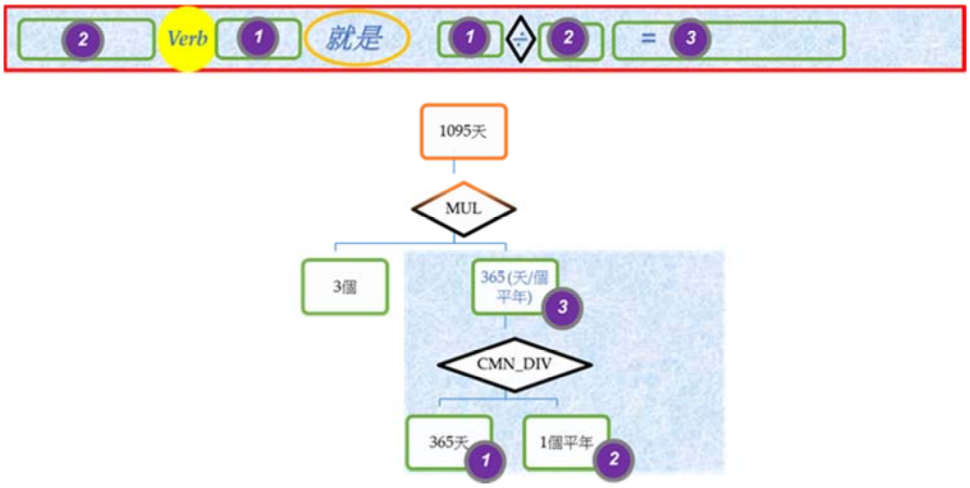


Figure 8 (a)

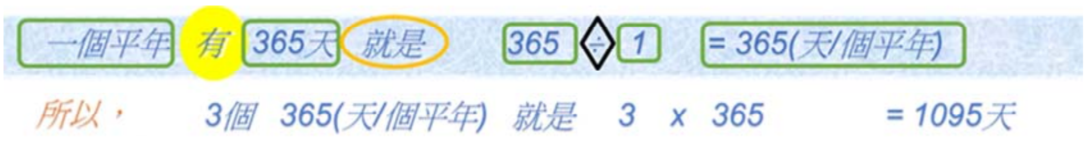


Figure 8 (b)

Figure 8. The template for OP_COMMON_DIV (“CMN_DIV” in Figure 8 (a)) and the explanation sentences for Sample-2.

[Sample-3] 一艘輪船 20 分鐘可以行駛 25 公里，2.5 小時可以行駛多少公里？

(A ship can travel 25 km in 20 minutes. How many kilometers can it travel for 2.5 hours?)

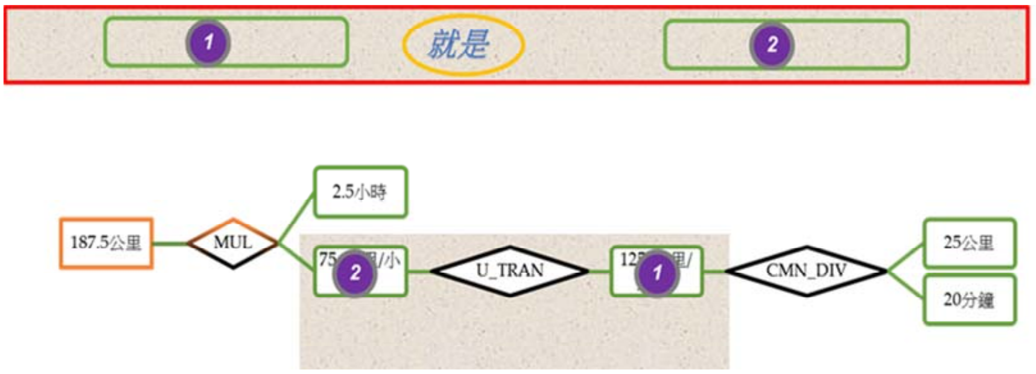
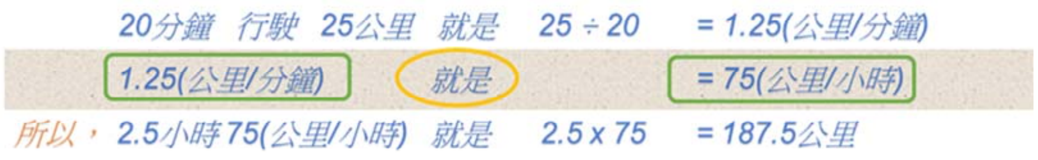


Figure 9 (a)


20分鐘 行駛 25公里 就是 25 ÷ 20 = 1.25(公里/分鐘)

1.25(公里/分鐘) 就是 = 75(公里/小時)

所以，2.5小時75(公里/小時) 就是 2.5 x 75 = 187.5公里

Figure 9 (b)

Figure 9. The template for OP_UNIT_TRANS ("U_TRAN" in Figure 9 (a)), which performs unit conversions, and the explanation sentences for Sample-3.

4. Current Status

Currently, 11 types of operators are supported. They are shown at Figure 10. After having manually checked 37 MWP problems with their associated operations specified in Figure 10,

Operation Utilities

Sum(function[,condition])=value
Add(value1,value2)=value
Subtract(value1,value2)=value
Diff(value1,value2)=value
Multiply(value1,value2)=value
FloorDiv(value1,value2)=value
CeilDiv(value1,value2)=value
Surplus(value1,value2)=value
ArgMin(arg,function,condition)=value
ArgMax(arg,function,condition)=value
UnitTrans(Old-Fact, New-Fact)=value

Figure 10. Supported Operators by EG

it is observed that the proposed approach could generate fluent explanation for all of them.

5. Related Work

Earlier reported NLG applications include generating weather reports (Goldberg *et al.*, 1994; Coch, 1998), instructions (Paris *et al.*, 1998; Wahlster *et al.*, 1993), encyclopedia-like descriptions (Milosavljevic, 1997; Dale *et al.*, 1998), letters (Reiter *et al.*, 1999), and an alternative to machine translation (Hartley & Paris, 1997) which adopts the techniques of connectionist (Ward, 1994) and statistical techniques (Langkilde & Knight, 1998). However, none of them touched the problem of generating explanation for MWPs.

Previous approaches of natural language generation typically consist of a discourse planner that plans the structure of the discourse, and a surface realizer that generates the real sentences (Jurafsky & Martin, 2000). D. Jurafsky adopted the model of text schemata and rhetorical relation planning for discourse planning. Approaches for surface realizer include Systemic Grammar, which is a part of Systemic Functional Linguistic proposed by Halliday (Halliday, 1985), and Functional Unification Grammar (FUG) by Kay (Kay, 1979).

Different from those previous approaches for Discourse Planner (Reiter *et al.*, 1999), we solved the EG for MWP problem through first buildings the Explanation Tree, which is particularly suitable for representing math based problems. The OP oriented algorithm is then proposed for solving the discourse planning work in MWP. Furthermore, different from the FUG proposed by Kay (Kay, 1979), the Function Word Insertion and Ordering Module adopts the OP based template for our Surface Realizer.

6. Conclusion

Since the EG for MWP differs from that of other NLG applications in that the inference process centers on the mathematical operation, an operator oriented algorithm is required. In the proposed framework, we first introduce the Explanation Tree to explicitly show how the answer of a math problem is acquired. Afterwards, an OP Oriented Algorithm performs sentence segmentation (act as Discourse Planner) for MWP. Lastly, for each operator, a corresponding template is adopted to achieve surface string realization.

Our Explanation Generator of MWP solver is able to explain how the answer is obtained in a human comprehensible way, where the related reasoning steps can be systematically explained with fluent natural language. The main contributions of this paper are:

1. *Proposing the Explanation Tree for facilitating the discourse planning on MWP.*
2. *Proposing an Operator oriented algorithm for structuring output sentence sequence.*
3. *Proposing the OP oriented templates for generating final explanation strings.*

References

Coch, J. (1998). Interactive generation and knowledge administration in MultiMétéo. In *Proceedings of the Ninth International Workshop on Natural Language Generation*, 300-303.

Dale, R., Oberlander, J., Milosavljevic, M., & Knott, A. (1998). Integrating natural language generation and hypertext to produce dynamic documents. *Interacting with Computers, 11*(2), 109-135.

Goldberg, E., Driedger, N., & Kittredge, R. (1994). Using natural-language processing to produce weather forecasts. *IEEE Expert: Intelligent Systems and Their Applications*, *9*(2), 45-53.

Halliday, M. A. K. (1985). *An Introduction to Functional Grammar*. London, England: Edward Arnold.

Hartley, A., & Paris, C. (1997). Multilingual document production: From support for translating to support for authoring. *Machine Translation*, *12*(1), 109-128.

Jurafsky, D., & Martin, J. H. (2000). *Speech and Language Processing*. New Jersey: Prentice Hall.

Kay, M. (1979). Functional Grammar. In *BLS-79*, Berkeley, CA, 142-158.

Langkilde, I., & Knight, K. (1998). The practical value of n-grams in generation. In *Proceedings of the Ninth International Workshop on Natural Language Generation*, Niagara-on-the-Lake, Ontario, Canada, 248-255.

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

Mayer-Schönberger, V., & Cukier, K. (2013). *Big Data – A Revolution That Will Transform How We Live, Work, and Think*. Houghton Mifflin Harcourt Publishing Company.

Milosavljevic, M. (1997). Content selection in comparison generation. In *Proceedings of the 6th European Workshop on Natural Language Generation*, Duisburg, Germany, 72-81.

Mukherjee, A., & Garain, U. (2008). A review of methods for automatic understanding of natural language mathematical problems. *Artif Intell Rev*, *29*(2), 93-122.

Paris, C., & Vander Linden, K. (1996). DRAFTER: An interactive support tool for writing multilingual instructions. *IEEE Computer, 29*(7), 49-56.

Paris, C., Vander Linden, K., & Lu, S. (1998). Automatic document creation from software specifications. In *Proceedings of the 3rd Australian Document Computing Symposium (ADCS-98)*, 26-31.

Reiter, E., Robertson, R., & Osman, L. (1999). Types of knowledge required to personalise smoking cessation letters. In *Proceedings of the Joint European Conference on Artificial Intelligence in Medicine and Medical Decision Making*. Springer-Verlag, 389-399.

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

Strassel, S., Adams, D., Goldberg, H., Herr, J., Keesing, R., Oblinger, D., Simpson, H., Schrag, R., & Wright, J. (2010). The DARPA Machine Reading Program - Encouraging Linguistic and Reasoning Research with a Series of Reading Tasks. *LREC 2010*.

Wahlster, W., André, E., Finkler, W., Profitlich, H.-J., & Rist, T. (1993). Plan based Integration of Natural Language and Graphics Generation. *Artificial Intelligence*, *63*(1993) 387-428.

Ward, N. (1994). *A Connectionist Language Generator*. New Jersey: Ablex Publishing Corporation.