國立臺灣大學電機資訊學院資訊工程學研究所

碩士論文

Graduate Institute of Information Engineering

College of Electrical Engineering and Computer Science

National Taiwan University

Master Thesis

用樹狀結構的算式生成器解數學問題

Tree-Structured Equation Generator for Math Word Problems With Deep Inference

呂紹瑀

Shao-Yu Lu

指導教授：鄭卜壬 博士

Advisor: Pu-Jen Cheng, Ph.D.

中華民國一零八年七月

July, 2019

doi:10.6342/NTU201902039

誌謝

能完成這篇論文，我要感謝我的指導教授鄭卜壬老師，從討論研究方向到確立題目，過程中展現了無比的親切與耐心，並總是為學生著想這點令人深感在心。實驗室的風氣自由活潑但不失認真，大家感情和睦，減輕了許多研究過程中的壓力。實驗室同儕在研究上的積極性，也幫助我接觸到更深更廣的研究領域。

兩年來能順利完成學業，真心地感謝一路上許多人的幫助。感謝前後三位室友的相伴，總是給予我生活上的幫助，並為研究所的日子增添了許多樂趣。謝謝實驗室的學姊推薦宿舍的工讀，保障了住宿機會，也讓我在經濟上少了後顧之憂。也感謝一位打工認識的博士班學姐，對我總是耐心開導，並用自身經驗幫助我面對研究瓶頸。

在最後一段，我要感謝我的家人，母親與兄妹總是維持良好的家中氣氛，盡可能地給予我支持與陪伴。這邊特別向我的父親致謝，從小不吝惜地栽培我，在考上研究所之際應該是他最驕傲的一刻，但惋惜的是，他終究沒能見到我畢業。一年半以來每個禮拜的返家，雖然對研究造成了莫大的影響，但慶幸沒在父親身上留下太多遺憾，也希望他能在天上見證這一刻。

doi:10.6342/NTU201902039

中文摘要

近年來隨著深度學習的發展，數學應用題的自動求解又再度引發研究上的關注。基於深度學習的序列生成模型被大量的應用在生成算式模板，不僅解決了傳統基於模板方法的限制，也是目前的主流解法之一。但鮮少有人試圖將生成模型與基於樹的方法結合。因此本文提出一個結合兩者的方法。我們希望模型可以學習到題目中的數字、所求目標以及過程中產出的數字三者的語意表示。並模仿人類真實的解題情境，將所有運算的中間產出也視為可選擇的數字，供模型學習如何進行進階的運算。

在大型的中文數據集 Math23K 的初步實驗結果中，我們的模型比起傳統基於注意力機制的 seq2seq 生成模型，正確率上升超過 7%。並在後續的實驗中，分析模型裡的一些機制如算式歸一化、數字特徵的自注意力機制，兩者會如何影響準確性。然後在最後一個實驗裡顯示，透過加強選擇數字的能力，模型能學習到類推與抽象化運算的能力，解決資料集中罕見但運算複雜而冗長的題型。即便取得了部分的成功，在一些失敗例子的探討上，我們也揭露了這個模型本身策略帶來的優勢與限制，以及這個領域的主要瓶頸。並以未來可以改善的方向作結。

關鍵字：數學應用問題、二元表示樹、算式生成器、數字語義理解、序列到序列模型、自然語言處理

doi:10.6342/NTU201902039

ABSTRACT

In recent years, the study of solving math word problem automatically receive much attention once again. Deep Neural Networks based method reach a new higher score on large-scale datasets. The generator based on seq2seq model combined with template-based method becomes the mainstream method for this task. However, few people try to introduce the deep learning to Tree-based model.

This paper propose a method to bridge the gap of tree-based method and deep learning method. We hope that model could learn the semantic meanings of quantities in a question, target of calculation and number produced during the process. By strengthening the ability of choosing number, model also perform the analogy and abstraction during complex process of computation.

The preliminary experiments are conducted in a benchmark dataset Math23K, and our model outperforms the seq2seq model with attention mechanism over about 7% accuracy, demonstrating the effectiveness of the selection strategy.

Keywords: math word problems, expression tree, equation generator, number semantic learning, seq2seq model, natural language processing

doi:10.6342/NTU201902039

目錄

doi:10.6342/NTU201902039

doi:10.6342/NTU201902039

圖目錄

doi:10.6342/NTU201902039

表目錄

\

doi:10.6342/NTU201902039

一、研究簡介

數學題的自動求解一直以來被認為是具挑戰性的任務。即使是小學程度的應用例題，還是得在簡短的敘述中，了解題目所求目標，還要將每個字句建立成目前擁有的條件，再去架構算式。這涉及到語意理解與邏輯推理的高度結合，前者用來理解題目所求、每個數字和條件代表的涵義；後者則是架構求解過程的能力與方法，並且涉及到對中間產出數字的類推能力。若缺乏前者，則會看不懂題目；若缺乏後者，便不能安排好運算的順序。因此對多樣化的數學題，兩個能力缺一不可。

在求解器的相關研究中有兩大主流方法，分別是基於模板的方法(Template-based) 與基於樹的方法 (Tree-based)。模板在這裡代表一條算式的架構，其中的數字以代數呈現。在傳統的基於模板方法，會對能以此模板求解的所有題目來抽取模板特徵，之後遇到新問題時，將問題與不同的模板特徵計算分數，選擇匹配度最高的模板，再從題目中選取數字填入其中求解。基於樹的方法，將所求的算式以後序式的型態來表示，並針對題目中的每個數字抽取特徵，判斷其是否為構成算式的元素之一，然後將數字當作葉結點、運算符當作枝節點，列舉出所有可能的二元表達樹，並選出最有可能的一個。以邏輯推理的角度來看，前者利用題目敘述和預定義的模板特徵，直接決定算式架構，再想辦法將數字填入；後者則是先判斷數字本身的意義，再由底至上建立出算式，比起需要預定義、由經驗而論的模板，更能應對沒出現過的算式型態，也象徵更理解數字彼此間的操作。

doi:10.6342/NTU201902039

template
$n_3-\left(n_2+n_1\right)$
$n_3\,n_2\,n_1+-$

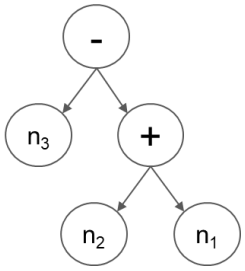

Figure 1: Examples of a template and an expression tree.

近年來，深度學習被引入到這個領域，由於其在自然語言上取得了很大的成就，促進了數學求解器對題目語義的理解，並在大範圍、多樣化的數據集中，得到了更好的準確度。在最近的相關研究中，許多應用了深度學習的求解器是立足於原本基於模板的方法，並以序列到序列的神經網路 (seq2seq) 為主架構，試圖去生成模板來解題。以語意理解的角度切入，模型能在對題目編碼的過程中自動抽取高品質的特徵，相較過去利用統計、基於規則的手工特徵來得方便。於邏輯推理的角度來看，生成過程中模型會學習架構算式的方法，好處是不會像傳統方法受限於模板數量，但不保證能完全生成有效算式。雖然生成出無效算式不是目前主要的瓶頸，但仍為一種隱患。另外，深度學習生成模板無法解釋為何如此架構算式，有著注意力機制的模型可以告訴我們填入每一步時關注的重點，但過去的方法不關注數字彼此間的關聯、怎麼選擇數字結合、和其互相結合後的抽象意義。

因此我們嘗試提出結合樹的深度學習生成器，在生成算式的過程中，強調選取數字與操作數字的邏輯能力。我們應用序列模型來捕捉數字與題目特徵，解碼器決定運算符的序列生成，這對應了我們在現實中的運算順序。從題目裡抽到的數字特徵，儲存在一個會擴增的池 (pool) 中，池裡的數字象徵我們擁有的線索與材料，供解碼器每一輪生成完運算符後來選擇。模型會有序地選出兩個數字，與運算符結合後再擴增原本的池，代表我們在這輪運算中得到的新結果，可以提供新的線索，若我們選取了那些合成的數字，則表示我們基於先前的結果再作進

doi:10.6342/NTU201902039

一步的運算。此外我們會將每一輪結果當作解碼器下一輪的輸入，就像是人會仔細端詳前一條算式再做下一步一樣。經過數輪回合後，模型判斷運算完成時，選取池中可能性最高的數字(通常是前一次運算出的結果) 輸出。

從生成結果的角度來看，由於題目使用的運算符，像加減乘除，都屬於二元運算，因此這個方法保證了算式的有效性，此外我們也加入了一些額外常數，來應對部分數字出現在算式中卻沒有在題目裡的情形。

這個方法的主要貢獻在於結合了深度學習與基於樹的方法，得以將優質的語義特徵用於具解釋性的推理過程，更接近現實中人在推理的情境。而且過程中合成的數字將帶有飽滿的意涵，給解碼器充分的解題線索，得以做進一步的運算，我們也預期這個架構可以將更複雜的運算抽象化，進而提高準確性。操作數字的抽象邏輯能力也讓模型能展現多元的解題方案，並應付各式各樣的問題。在流程上這也是一個更直觀、簡單的模型。我們希望透過這個策略能讓模型有更好的邏輯能力，雖然有些機制尚未完善，但這是一個新的方案試圖訓練機器去學習人類解題的技巧。

doi:10.6342/NTU201902039

二、相關研究

在 1960 年代時，就開始有人設計自動求解器 [1][2][3]。早期多是基於規則的方法，嚴重仰賴人工制定的大量匹配模式與規則，但只能解決幾個預先設想的場景。後來在 2011~2017 年間，研究者開始將題目語義轉化成有結構的邏輯式 [4][5]，以方便推導，這又再度引起眾人對求解器的研究興致，許多特徵工程、基於統計的方法應運而生。

Kushman 等人在 2014 年提出基於模板 (template-based) 的方法 [6]。從預先定義方程模板的資料庫中，選出最佳匹配模板，再從文本中提取數字和訊息來填充模板的未知槽。後續陸續有人針對這個方法進行搜索空間的改善 [7]。FG-Expression [8]則是跳脫了原本模板的框架，將模板切分成片段，訓練模型精確地映射題目與模板片段，再以樹結構將映射到的模板片段組合起來。

Roy and Roth 等人在 2015 年提出第一種基於樹 (tree-based)的方法 [9]，這個方法的泛用性也讓它一度成為求解器的主流方案。整體算法含兩個階段。在第一階段，從文本中提取數字並形成樹的底層，列舉語法上有效但結構不同的候選樹。第二階段則定義評分函數以選擇最佳匹配候選樹。基於樹的方法解決了模板方法受限於訓練實例的缺點，能產生各式各樣的算式，以應付多樣化題型。但列舉樹的可能性太多，後續的相關研究都以降低搜索空間為目標 [10]，甚至在建立樹的過程中用許多規則來限制 [11]。也有研究嘗試對數字抽取更詳實的關聯特徵，讓評分函數的運作更加有效 [12]。

前兩個方法顯示了一些限制，模板方法嚴重仰賴文本特徵，因為在評分時除了問題的特徵，模板也會以其實例歸納出的特徵來表示。這代表我們需要對模板進行額外標注。另外，在訓練時也受限於預定義的模板種類與數量，導致這個方法特別不適用於範圍大、多樣化的數據。基於樹的方法除了搜索空間外，也有抽

doi:10.6342/NTU201902039

取數字特徵的議題存在。如 Huang 等人所評估的 [8]，它們的精確度在更大，更多樣化的數據集中急劇下降。研究結果表明，這一系列研究仍有很大的改進空間，需要更加通用和可靠的解決方案。

在 2017 以後，求解器開始導入深度學習。DNS [13]是第一個不需要手工特徵的基於深度學習方法，其使用了 seq2seq 模型作為算式生成器，並制定了一些語法上的規則，以確保生成有效算式。Seq2SeqET [14]則用後序式作為輸出序列，可以將其視為另一種模板。為了減少模板空間，在 Seq2SeqET 中提出了方程歸一化，以便可以統一無序運算符的等價算式。StackDecoder [15]也是 seq2seq 模型。其編碼器在題目文本中提取數字的語義，並配上有 stack 的解碼器，以便追蹤數字語意在操作過程中的變化。T-RNN [16]一樣生成後序式模板，可被視為 Seq2SeqET 的改進。首先，用 Bi-LSTM [17]和自注意力機制 (self-attention) [18]賦予數字特徵，並將不同的運算符皆視為符號$< op >$，再用 recursive NN，每次以兩個象徵數字的子節點，推斷每個父節點$< op >$屬於哪個運算符。

doi:10.6342/NTU201902039

三、研究方法

在我們的系統架構中，題目需要透過前處理才能被輸入進模型中。而算式也需要經過一系列的轉化，成為基於樹的算式，並供整個系統與生成出的樹狀算式計算 Loss。在訓練過程中，我們會參考正確答案，來輔助模型學習正確的語義；而在測試過程中，我們並不會在每個階段參考答案，而是直接生成整個算式。

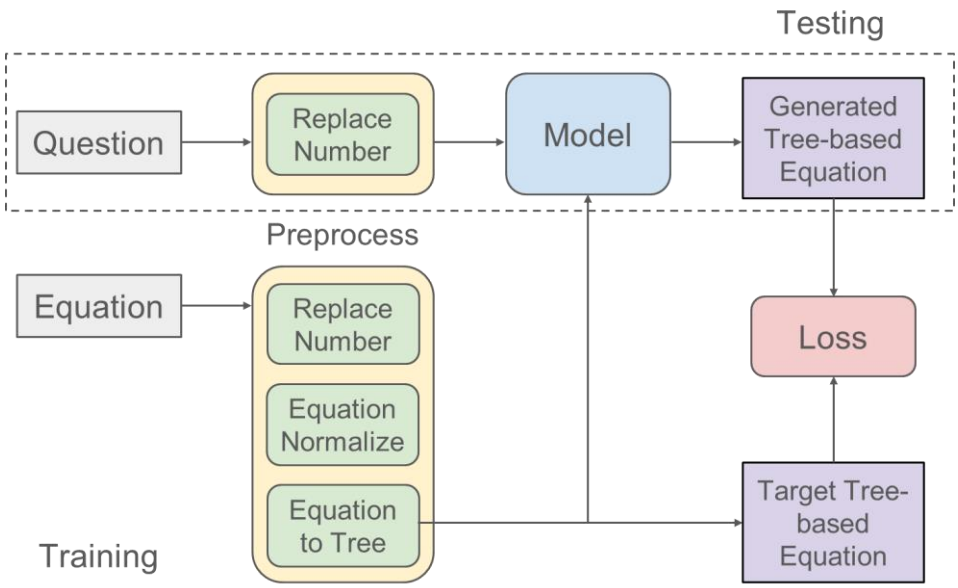

Figure 2: System Architecture Diagram

在這個章節，我們分兩個部分說明，3.1.1 說明題目敘述中，替換數字的前處理與用意，3.1.2 則說明答案式將進行三個階段的前處理，才能被轉為二元樹，分別是：數字抽換、算式歸一化、轉換為樹的表示法。

doi:10.6342/NTU201902039

3.1 前處理

3.1.1 題目敘述的前處理

首先敘述的部分，我們偵測題目裡有的數值，加入數字清單中 (number list)，並將每個數字換成符號$< NUM >$。為的是之後在序列編碼過程中，希望編碼器可以專注學習如何用上下文和在文本中的位置來表示一個數字。因為我們認為，算式結構受文本的影響大於數字的量級本身。題目裡的每個數字用任何未知數來表示，都不影響解題的邏輯與算式。另外，這樣的表示法也有一個好處，模型抽取的數字特徵會更穩定，同篇文本同樣位置的數字，不論字面上是 400 還是 70，我們都降低了它們混淆編碼器的可能性。

另外，我們以字元為單位進行斷詞，此做法在先前的研究中顯示 [15]，比以詞為單位能帶來略優的表現。因為數學題目的敘述通常很精簡，這樣的情形下每個字的重要性提高，斷詞的好壞就會影響結果。所以這個做法反而能減少出錯的機率，並讓數字在用自注意力(self-attention) 擷取特徵時，能更加精準。

3.1.2 算式的前處理

緊接著進行算式的前處理。第一步驟一樣將數字做抽換，我們將算式中有出現在清單的數字換成符號temp_i，i 代表數字在清單中的順序，目的是要確立我們要結合的數字是哪一個。而不使用數字本身視同將運算元抽象化，幫助模型學習與歸納。

第二步驟進行算式歸一化。如果我們檢查算式的過程中，發現有兩個數字進行了加或乘的運算，但是順序的擺放不符合它們在清單中的排序，則進行調換，調換後的版本取代原版成為答案式。因為加或乘的運算結果不受運算元順序影響，所以可以對調。此做法是避免模型在選擇前後運算元時產生模糊性，透過事前的處理，讓模型在面對無序運算元符時有明確的優先順序，也減少處罰到另一個正解的機

doi:10.6342/NTU201902039

會。另外歸一化僅在運算元都是題目中的數字時才會進行。

第三步驟則是將原本的中序式轉成後序式，後序式可以看作成一顆二元樹，其中運算符是內節點，數字則為葉節點。

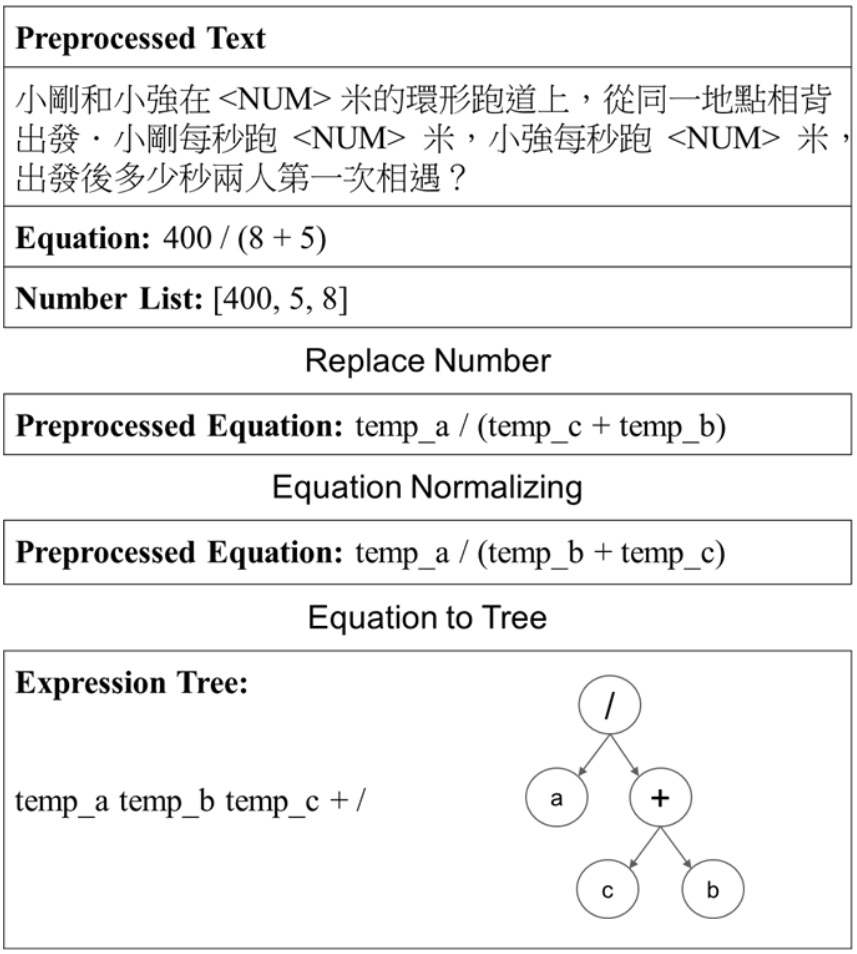

Figure 3: An example for preprocess of an equation

doi:10.6342/NTU201902039

3.2 模型架構

模型的組成主要有 5 個部分，3.2.1 會介紹編碼器的構造，以及他會如何對問題進行編碼、對每個數字抽取特徵，有關數字的資訊會額外使用自注意力機制加強。3.2.2 的池構造用來儲存每個出現過的數字特徵，包括一開始從題目中取得的數字以及在運算過程中產生的數字，供 3.2.4 做選擇，另外有兩個獨立於題目以外的常數特徵，也會在這部分一併介紹。3.2.3 介紹解碼器與運用了哪些階段性資訊，以及生成運算符的方法。3.2.4 則說明數字選擇器如何運用前一階段關注的資訊，依序選擇出前運算元與後運算元。最後一個構造在 3.2.5 介紹，數字合成器如何使用運算符，將前後運算元合成出新數字。

3.2.1 編碼階段 (Encode)

在編碼階段為了獲取數字在文本中的有效特徵，我們選擇 BiLSTM 作為編碼器。並將長度為 m 的題目經過一層嵌入層 (embedding layer) 後，得到詞嵌入 (word embedding)，並輸入至 BiLSTM 得到 m 個 d 維的隱藏層向量 $\{h_i^E\}_{i=1}^m$。

我們認為最後一個隱藏層向量挾帶著題目的總體訊息，所以將它與 $\{h_i^E\}_{i=1}^m$ 計算注意力分數，並做加權平均得到 e^{var} 當作是題目未知數的表徵向量。

然後針對在前處理抽取到的數字清單 $\{Num_i\}_{i=1}^n$，我們使用它們對應在文本中的位置 $\{p_i\}_{i=1}^n$，取出對應的隱藏層向量 $\{h_{p_i}^E\}_{i=1}^n$，個別計算他們與 $\{h_i^E\}_{i=1}^m$ 的注意力分數，並做加權平均，當作是每個數字的特徵向量。

doi:10.6342/NTU201902039

$\{w_t^Q\}_{t=1}^m$: *words in the text of question Q*

$$e_t^Q = Embedding\left(w_t^Q\right)$$

$$h_t^E,\ c_t^E = BiLSTM\left(h_{t-1}^E,\ c_{t-1}^E,\ e_t^Q\right)$$

$$e^{var} = Attention\left(h_{m-1}^E,\ H^E\right) = \sum_t \alpha_t\, h_t^E$$

$$\alpha_t = \frac{exp\left(s_t\right)}{\sum_i exp\left(s_i\right)},\ s_t = h_t^E\, W_{var}^T\, h_{m-1}^E,$$

where W_{var} *is a trainable matrix with size* $d \times d$

3.2.2 池架構 (Pool)

編碼完成後，我們對池進行初始化。將做完自注意力的數字特徵向量依序加入池。並額外加入兩個常數的特徵向量，分別是 1 和 π。在後續選數字的階段，這兩個向量會如同池中的其他數字一樣，在後續的階段被平等的選取。

$$P = \left\{e_1^{num},\ e_2^{num},\ \ldots,\ e_n^{num},\ e^1,\ e^\pi,\ e_1^{comb},\ e_2^{comb},\ \ldots,\ e_k^{comb}\right\}$$

因為算式會出現題目中不存在的數字，前者經常在比率問題中出現，後者則會在圓的幾何計算中出現，但題目沒有出現任何相關的數字，這類型的問題屬於領域知識 (domain knowledge) 的範疇。為了處理這個問題，我們將這兩個額外的數字特徵視為模型參數之一，並在開始時隨機初始化，期望透過大量數據能夠訓練出它們的表徵。

$\{p_i\}_{i=1}^n$: *positions of numbers in the text*

$$e_i^{num} = Attention\left(h_{p_i}^E,\ \{h_j^E\}_{j=1}^m\right)$$

$e^1,\ e^\pi$: *trainable embeddings for external constants* 1 *and* π

$$P = \left\{e_1^{num},\ e_2^{num},\ \ldots,\ e_n^{num},\ e^1,\ e^\pi\right\}$$

doi:10.6342/NTU201902039

Example
畫圓時，圓規兩腳叉開的距離是2cm，畫出的圓的周長=多少cm。 **Equation:** (3.14 * 2) * 2 **Number List:** [2]
一個養雞專業戶養了一些雞，賣出20%後還剩1280只。這個養雞專業戶原來有雞多少只？ **Equation:** 1280 / (1 - 20%) **Number List:** [20%, 1280]

Figure 4: Examples of question have 1 and π.

3.2.3 解碼階段 (Decode)

我們選擇 2 層的單向 LSTM 作為解碼器，解碼器的輸入分別是前一階段的隱藏層 h_{t-1}^{D}, c_{t-1}^{D}、前一階段的運算結果 st_{t-1}。我們用編碼器最後一層的隱藏層為解碼器初始化，並將編碼階段產生的變數特徵 e^{var} 當作第一步的輸入，雖然 e^{var} 嚴格來說不算是運算結果，但也代表我們在著手第一個運算時要先以目標變數提供的訊息做參考。在第二步以後會將前一步合成的數字 e_{t-1}^{comb} 當作輸入，這個用意是在模仿人解題時回顧「目前算到哪裡」的行為，並用來做下一步運算的參考。

LSTM 會輸出一個隱藏層 h_t^D，我們將它與問題的隱藏層向量 $\{h_i^E\}_{i=1}^{m}$ 計算注意力分數，象徵這一輪的選擇主要關注在題目的哪個部分，並計算出當前的問題文本特徵 q_t，我們將隱藏層 h_t^D、運算結果特徵 st_{t-1}、問題文本特徵 q_t，3 個向量合併成一個長向量 HNQ_t，作為我們解題要考慮的三個面向，除了用於運算符的選擇外，也會被送入下一階段作為選則運算元的參考特徵。

在運算元的選擇上，我們將 HNQ_t 作為分類器的特徵輸入，過一層線性轉換與 $softmax$ 函數，選擇機率最高的運算符以數字索引的型式輸出。

doi:10.6342/NTU201902039

$$h_t^D,\ c_t^D = LSTM\left(h_{t-1}^D,\ c_{t-1}^D,\ st_{t-1}\right)$$

$$st_{t-1} = \begin{cases} e^{var}\ ,\ if\ \ t=1 \\ e_{t-1}^{comb}\ ,\ else \end{cases}$$

$$HNQ_t = \left[h_t^D;\ st_{t-1};\ q_t\right]$$

$$q_t = Attention\left(h_t^D,\ \{h_j^E\}_{j=1}^m\right)$$

$$P\left(O_t \mid \{o_i\}_{i=1}^{t-1},\ \{w_i\}_{i=1}^m\right)$$
$$= sof\,tmax\left(W_{op}\,HNQ_t\right)$$

W_{op} : *a trainable matrix with size* $5\times 4d$.

3.2.4 數字選擇器 (Number Selector)

在數字的選擇上，我們依序選擇出一前一後的數字，以滿足減與除運算的有序性，並且透過剛剛的歸一化，我們也把有序性帶給部分的加與乘運算。如果我們換另一個做法，求機率最高的兩個數字，在遇到減與除時免不了還要決定兩者的順序，而阻斷了某些運算有選取兩個相同數字的需求，例如求正方形面積。

在選取前運算元時，我們將 HNQ_t通過一層網路，過濾出 3 個層面中，選擇前運算元要著重的資訊，然後將它跟前一階段生成的運算元 O_t合併成 r_t^{fr}，維度為 $d+1$的向量。接下來將 r_t^{fr}和池中的每個數字計算分數，並用$softmax$計算出機率，選取出機率最高的數字特徵向量，作為是前運算元的特徵向量 e_t^{fr}。

doi:10.6342/NTU201902039

O_t : *an index of the operator produced in previous stage*

V^{fr}: *a trainable matrix with size* $d \times (2d+1)$

W^{fr} : *a trainable matrix with size* $d \times 4d$

b^{fr} : *a trainable vector with size d*

$$P\left(i_t | \{O_j\}_{j=1}^{t}, \{w_k\}_{k=1}^{m}\right) = sof\,tmax\left(\alpha_{t,k}\right)$$

$$\alpha_{t,k} = V^{fr}\left[r_t^{fr}; P_k\right]$$

$$r_t^{fr} = \left[HNQ_t^{fr}; O_t\right]$$

$$HNQ_t^{fr} = W^{fr} HNQ_t + b^{fr}$$

選取後運算元的過程跟前者相似，一樣將 HNQ_t通過一層網路過濾資訊，但在合併成 r_t^{bh}的方式不太一樣，除了運算元 O_t ，我們還要加入剛剛計算出的 e_t^{fr}，合併成維度是 $2d+1$的 r_t^{fr}。接下來一樣將 r_t^{bh}和池中每個數字向量計算出機率，選擇機率最高者作為後運算元。

e_t^{fr} : *the embedding selected for front number in previous stage*

$$P\left(i_t | \{O_j\}_{j=1}^{t}, \{w_k\}_{k=1}^{m}, e_t^{fr}\right) = sof\,tmax\left(\alpha_{t,k}\right)$$

$$\alpha_{t,k} = V^{bh}\left[r_t^{bh}; P_k\right]$$

$$r_t^{bh} = \left[HNQ_t^{bh}; O_t; e_t^{fr}\right]$$

另外，若前一階段的運算符為等號，則宣告計算即將完成，我們在此階段只要選擇前運算元並回傳對應的算式當作答案。

doi:10.6342/NTU201902039

3.2.5 合成器 (Combinator)

當解碼階段輸出的運算符不是等號時，會進入合成數字的階段。我們根據加減乘除不同運算，訓練四種合成器，並用運算符 O_t 去選擇對應的合成器。將前後運算元的特徵向量合併，過兩層網路，每層分別通過激活函數 $ReLU$ 和 $tanh$，得到的運算結果作為兩者合成的向量。合成的向量會加入池，並被當作下一階段解碼器的輸入。

$$op : an\ operator \in \{+, -, *, /\}$$

$$W_{op}^{comb} : a\ trainable\ matrix\ with\ size\ d \times 2d$$

$$V_{op}^{comb} : a\ trainable\ matrix\ with\ size\ d \times d$$

$$b_{op}^{comb} : a\ trainable\ vector\ with\ size\ d$$

$$c_{op}^{comb} : a\ trainable\ vector\ with\ size\ d$$

$$e_t^{comb} = combinator_{op}\left(e_t^{fr}, e_t^{bh}\right)$$
$$= tanh\left(V_{op}^{comb} ReLU\left(W_{op}^{comb}\left[e_t^{fr}; e_t^{bh}\right] + b_{op}^{comb}\right) + c_{op}^{comb}\right)$$

3.3 損失函數 (Loss Function)

關於 Loss function 的計算，主要是加總每一輪運算符、前後運算元分類器的 Cross Entropy 來計算，以最小化以下算式為目標：

$$L(\theta) = -\sum_{t=1}^{n} \log P\left(O_t = o_t \middle| \{o_i\}_{i=1}^{t-1}, \{w_k\}_{k=1}^{m}\right)$$
$$-\sum_{t=1}^{n} \log P\left(F_t = i_t \middle| \{o_i\}_{i=1}^{t}, \{w_k\}_{k=1}^{m}\right)$$
$$-\sum_{t=1}^{n} \log P\left(B_t = i_t \middle| \{o_i\}_{i=1}^{t}, \{w_k\}_{k=1}^{m}, e_t^{fr}\right)$$

doi:10.6342/NTU201902039

訓練時，我們會設定 Teacher forcing ratio 為 1。參考每一輪運算符、前後運算元的正確答案，主要是因為每次做完選擇不能回頭修改答案，卻都會影響下一輪的運算，像是利用錯誤的前運算元去求後運算元。所以要在選擇後馬上將錯誤修正，以免讓模型學習到錯誤的語義和因果關係。

doi:10.6342/NTU201902039

四、實驗結果

在 4.1 的部分，針對我們所做的相關實驗做介紹，包括使用的資料集、實驗設計、實驗結果。在 4.2 則透過數個模型生成的實例，說明模型所展現的能力與現制，以及這個領域尚未解決的議題。

4.1 實驗方法與分析

4.1.1 資料集簡介

這次使用的資料集是 Math23K，是一份大規模的中文資料集，有 23,162 道題目並標注了正確算式，每個問題都符合兩點原則：1.每個問題都能使用一條算式就解出答案。2.除了 1 和 Pi，所有算式中使用的數字都能在題目敘述中找到

4.1.2 實驗設計與比較

第一個實驗中，我們將模型生成的結果跟正確答案的算式做比較，全對才算正確。採用較嚴格的算式比對計算方式，是希望能確認模型具備生成正確算式的能力。比較的基線模型(Baseline model)為帶有注意力機制的 seq2seq 模型。並同樣以生成後序式為目標做訓練，以比較兩者的能力。

第二個實驗，則是觀察模型中的重要機制對模型表現的影響。我們嘗試分別移除對數字特徵的自注意力機制(self-attention)，以及前處理中的算式歸一化(equation normalizing)，並展示其在選擇運算符和運算元的表現，來分析兩者影響模型的層面與重要程度。

第三個實驗，則比較不同的目標算式的運算符的長度對正確率的影響。這個實驗的目的在於面對複雜程度深淺、實例數量落差大的題目，模型是否有將複雜算式抽象化，並推廣操作的能力。

doi:10.6342/NTU201902039

4.1.3 實驗結果分析

Model	Accuracy
seq2seq + attention	53.1%
Generator w/o equation normalizing	56.5%
Generator w/o self-attention	59.2%
Generator	**60.6%**

Table 1: Generation result on Math23K

在第一個實驗中我們發現，我們的模型在效能上比基線模型高了 7.5%，即使是能力限縮後的模型依然高出了 3.4%，這意味著這個生成策略對算式生成有一定的幫助，相較 seq2seq 模型有機會產生無效的算式，我們的模型並不會如此。

doi:10.6342/NTU201902039

Model	Accuracy	Operator	Number Select
Generator w/o equation normalizing	56.5%	66.6%	64.2%
Generator w/o self-attention	59.2%	67.8%	66.6%
Generator	**60.6%**	68.9%	67.6%

Table 2: Elimination of mechanism test

第二個實驗中看到 2 種機制的影響，首先移除掉數字的自注意力，代表僅使用 BiLSTM 對數字抽取特徵，造成整體正確率、運算元選擇正確率、運算符選擇正確率分別下降 1.4%,1.1%和 1%，發現下降是相當平均的，代表自注意力機制有小幅度但全面的影響力。另外算是歸一化帶來較大的震盪，整體正確率、運算元選擇正確率、運算符選擇正確率分別下降 4.1%,2.3%和 3.4%，可以看出我們在訓練時，將順序歸一化，能有效降低模型對運算元順序的混淆。

doi:10.6342/NTU201902039

# Operators	# Instances	Proportion	Accuracy
0	15	0.03%	80.0%
1	929	20.1%	73.6%
2	2263	48.9%	63.4%
3	935	20.2%	40.7%
4	295	6.4%	29.5%
5	125	2.7%	29.6%
6	40	0.8%	**17.5%**
7	13	0.3%	**30.7%**
8	9	0.2%	**11.1%**
9	3	0.04%	**33.3%**
10 or >10	5	0.1%	0%

Table 3: Accuracy for increasing length of operators.

第三個實驗一樣嚴格計算正確率，結果顯示模型在海量數據中，對於實例數量少、但運算較複雜的題目，仍具有一定的學習與推理能力。我們針對「選擇數字結合的能力」來訓練模型，或許象徵模型將能更好地解讀高階算式中的抽象義涵，並用來操作與運算。

Example
一条路，修了的和没修的长度比是6：7，又修了210千米后，修了的和没修的长度比是9：4，这条路全长=？ **Correct Answer:** 210 / ((9 / (9+4)) - (6 / (6+7))) **Generated Equation:** 210 / ((9 / (9+4)) - (6 / (6+7)))
甲乙丙3种糖果每千克的价格分别是9元，7.5元，7元，现把甲种糖果5千克，乙种糖果4千克，丙种糖果3千克混合在一起，那么36元可买多少千克这种糖果． **Correct Answer:** 36 / ((((9*5) + (7.5*4)) + (7*3)) / ((5 + 4) + 3)) **Generated Equation:** 36 / ((((9*5) + (7.5*4)) + (7*3)) / ((5 + 4) + 3))

Figure 5: Examples of generation result for longer length of operators.

doi:10.6342/NTU201902039

4.2 案例分析

4.2.1 等價算式

例子 1 中，模型顯示了組合順序不同但等價的算式。例 2 中，正確答案是先計算出需要的總天數再減掉已經過去的天數。但模型產生的算式是利用未完成率去除以每天的完成率，計算出剩餘需要的天數。例 3 要求第一次的木料使用量，中間的算式 (2.2/((5/6)-(3/8))) 的意義為木料原本的總長度。在正確答案中用地一天使用掉的比例去乘以全長，但在模型生成的算式中，則是用第二次用掉的木料去減掉兩次的總使用量，雖然計算出的答案為負，是錯誤的，但邏輯上的操作僅差在兩者的順序選錯而已。其他部分都象徵此模型能學習到等價算式的概念，即使在複雜的運算中亦同。

Example
一塊長方形花圃，它的 (1/9) 種月季，(2/9) 種菊花，其余的種玫瑰．種玫瑰的面積占這塊花圃的幾分之幾？ **Correct Answer:** 1 - ((1/9) + (2/9)) **Generated Equation:** (1 - (1/9)) - (2/9)
修路隊鋪一條公路，4天鋪了全長的(2/5)，剩下還要幾天鋪完？ **Correct Answer:** (4 / (2/5)) - 4 **Generated Equation:** (1 - (2/5)) / ((2/5) / 4)
一根木料，第一次用去全長的(3/8)，第二次用去2.2米，兩次共用全長的(5/6)，第一次用去多少米？ **Correct Answer:** (3/8) * (2.2 / ((5/6) - (3/8))) **Generated Equation:** 2.2 - (2.2 / ((5/6) - (3/8))) * (5/6)

Figure 6: Examples of generation results: 1, 2, 3

doi:10.6342/NTU201902039

4.2.2 語意模糊與領域知識

在例 4 中，顯示一部分的錯誤也是因為語意上的模糊所導致，雖然題目強調了第二天開始，並在末句加上「一周」的字眼，但計算上卻將天數減 1，而沒有出現任何可以判斷需要減 1 的線索。在例 5 則與領域知識的議題相關，模型顯然不具備「最大公約數」和「最小公倍數」的知識，即便能從大量數據中學習兩者分別的數學涵義，但要了解兩者的關係仍然具有一定難度。例 6 是涉及到幾何領域的知識，模型在計算時，可能不清楚題目中的正方形在邊長減少後，仍要維持以「正方形的面積」來計算差，而非計算減少一邊的邊長後「成為長方形」的差。

Example
一個工程隊挖土，第一天挖了 316 方，從第二天開始每天都挖 230 方，連續挖了 6 天，這個工程隊一周共挖土多少方？ **Correct Answer:** 316 + 230 * (6 - 1) **Generated Equation:** 316 + 230 * 6
甲、乙兩數的最大公約數是3，最小公倍數是 45，若甲數為 9，則乙數=多少： **Correct Answer:** 45 * 3 / 9 **Generated Equation:** 3 * (45 / 3)
一個正方形的邊長是 10 厘米，如果邊長減少 4 厘米，則它的面積減少多少平方厘米． **Correct Answer:** (10 * 10) - ((10 - 4) * (10 - 4)) **Generated Equation:** 10 * (10 - 4)

Figure 7: Examples of generation results: 4, 5, 6

doi:10.6342/NTU201902039

4.2.3 不正確的語意理解

例 7 是一個有趣的例子，模型看上去可能僅僅以「少 84」的字句來選擇減號和 84，但模型在生成算式時是缺乏數值資訊的，如果將 84 這個數字換做是某種比率，那這條算式會是正確的。例 8 顯然沒有將數字的訊息解讀正確，因而將把乘法選成加法。例 9 則是順序選擇錯誤，這也表示若模型在前運算元就選錯了，那在後面可能會缺乏選擇的餘地。例 10 則是把計算的主客體顛倒了，也沒有將字面的意思推論成反函數的能力。

Example
甲數是305，比乙數少84，乙數=・ **Correct Answer:** 305 + 84 **Generated Equation:** 305 / (1 - 84)
學校組織去看電影，光明小學全體師生共720人觀看，團體票每張9元，學校需要帶多少錢去買票？ **Correct Answer:** 720 * 9 **Generated Equation:** 720 + 9
一桶油重5千克，多少桶這樣的油重30千克・ **Correct Answer:** 30 / 5 **Generated Equation:** 5 / 30
科幻書每本32元，科幻書比故事書的3倍還少1元，故事書每本幾元？ **Correct Answer:** (32 + 1) / 3 **Generated Equation:** (32 * 3) - 1

Figure 8: Examples of generation results: 7, 8, 9, 10

doi:10.6342/NTU201902039

五、結論與未來研究方向

透過實驗一的結果，我們確實提出了有效的生成策略，並在效能上超越了比有注意力機制的 seq2seq 模型約 7.5%。在實驗二也證明部分重要的機制有助於提升模型表現，並是第一個分開討論這些機制如何影響數字與運算符的選擇，結果證明加入注意力的特徵抽取會有助於整體效益，算式歸一化則在選擇數字上有大幅的幫助，並改善高達 4.1%的總體正確率。實驗三則顯示模型在少量且運算冗長的實例中，依然能展現高度抽象化的推理能力。

在例 4 和例 7 中，我們發現文本依然有其解釋的極限，前者基於問題本身的模糊；後者則像是數學題領域的一句多義，而在這兩個例子中，數值本身便具有參考價值，能多少補足文本解讀的不足。或許能將數字本身的型態分類成分數、整數、浮點數，或是針對量級做各種處理，或許能成為選擇時的參考依據之一。

透過模型在例 9 中的表現，推測不可逆的選擇流程可能會影響模型的效能。因此考量在這方面較可行的改善可能是加入波束搜索，在不破壞流程、不增加太多搜索空間的前提下，考慮更多可能的組合方式，讓模型有機會做出更好的選擇。

在例 1~3 中，我們的模型能針對複雜的算式產生出等價結果，展現了其在邏輯類推上的能力，而抽象類推也是在數學上相當重要的能力。尤其針對比率問題，模型已經能熟稔地將常數 1 跟這個能力結合並應用。

但在其他例子中，我們其實還是能找到誤用的情形，Pi 的使用也有同樣的狀況，也許這表示我們還沒辦法將純語義與這些數字的使用精確地連結。這件事代表著一個潛在的隱患，即使已經有一定數量的數據去訓練這些常數的意義，但還是難以克服他們應用的場合需要別的知識。這種能具象化訓練的知識已經迎來了瓶頸，同樣的瓶頸也會發生在其他無法具象化訓練的知識上。

doi:10.6342/NTU201902039

從一個長遠的角度來看，我們終究要處理領域知識 (domain knowledge) 的議題，即便我們已經使用了預訓練的 word2vec，但它針對數學領域的語意理解還是有其限制。而我們認為，引入更先進的 NLP 研究成果是改進這個議題較可行的方案。

doi:10.6342/NTU201902039

六、參考文獻

[1] D. Bobrow, “Natural language input for a computer problem solving system,” in Semantic information processing, M. Minsky, Ed. MIT Press, 1964, pp. 146–226.

[2] E. A. Feigenbaum and J. Feldman, Computers and Thought. New York, NY, USA: McGraw-Hill, Inc., 1963.

[3] E. Charniak, “Computer solution of calculus word problems,” in IJCAI, 1969, pp. 303–316

[4] D. Goldwasser and D. Roth, “Learning from natural instructions,” in IJCAI, 2011, pp. 1794–1800

[5] T. Kwiatkowski, E. Choi, Y. Artzi, and L. S. Zettlemoyer, “Scaling semantic parsers with on-the-fly ontology matching,” in EMNLP, 2013, pp. 1545–1556.

[6] N. Kushman, L. Zettlemoyer, R. Barzilay, and Y. Artzi, “Learning to automatically solve algebra word problems,” in ACL, 2014, pp. 271– 281.

[7] L. Zhou, S. Dai, and L. Chen, “Learn to solve algebra word problems using quadratic programming,” in EMNLP, 2015, pp. 817–822.

[8] D. Huang, S. Shi, J. Yin, and C.-Y. Lin, “Learning fine-grained expressions to solve math word problems,” in EMNLP, 2017, pp. 805–814

[9] S. Roy and D. Roth, “Solving general arithmetic word problems,” in EMNLP, 2015, pp. 1743–1752.

[10] R. Koncel-Kedziorski, H. Hajishirzi, A. Sabharwal, O. Etzioni, and S. D. Ang, “Parsing algebraic word problems into equations,” TACL, vol. 3, pp. 585–597, 2015.

doi:10.6342/NTU201902039

[11]S. Roy, S. Upadhyay, and D. Roth, “Equation parsing : Mapping sentences to grounded equations,” in EMNLP, 2016, pp. 1088–1097.

[12]S. Roy and D. Roth, “Unit dependency graph and its application to arithmetic word problem solving,” in AAAI. AAAI Press, 2017, pp. 3082–3088

[13]Y. Wang, X. Liu, and S. Shi, “Deep neural solver for math word problems,” in EMNLP, 2017, pp. 845–854.

[14]L. Wang, Y. Wang, D. Cai, D. Zhang, and X. Liu, “Translating math word problem to expression tree,” in EMNLP. Association for Computational Linguistics, 2018, pp. 1064–1069.

[15]T. Chiang and Y. Chen, “Semantically-aligned equation generation for solving and reasoning math word problems,” CoRR, vol. abs/1811.00720, 2018.

[16]L. Wang, D. Zhang, J. Zhang, X. Xu, L. Gao, B. Dai, and H. T. Shen, “Template-based mathword problem solvers with recursive neural networks,” in AAAI. AAAI Press, 2019

[17]Sepp Hochreiter and Jurgen Schmidhuber. 1997. ¨ Long short-term memory. Neural Computation, 9(8):1735–1780.

[18]Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1412–1421.

doi:10.6342/NTU201902039