

國立臺灣大學電機資訊學院資訊工程學研究所

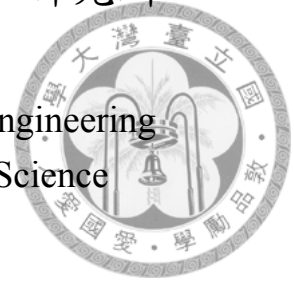
碩士論文

Department of Computer Science and Information Engineering

College of Electrical Engineering and Computer Science

National Taiwan University

Master Thesis



數理推演與對話中的自然語言理解之研究

Investigating Language Understanding in  
Arithmetic Reasoning and Conversation Modeling

江廷睿

Ting-Rui Chiang

指導教授：陳縉儂博士

Advisor: Yun-Nung Chen, Ph.D.

中華民國 109 年 1 月

January, 2020





# Acknowledgements

I would like to appreciate my advisor Yun-Nung Chen. She shows me the world. I appreciate MiuLab, which sparked the fire of my life, and has been providing me with the fuel. I appreciate every member in MiuLab<sup>1</sup>, especially who bore with my paroxysms of ignorance at those nights. I appreciate my family, who taught me the virtue of honesty. Wish everyone in the lab, stay passionate, stay genuine. Then, we will dare to whisper, “we have not lived in vain.”

---

<sup>1</sup>黃兆緯, 蔡尚錡, 張婷雲, 翁子騰, 王佑安, 蘇建嘉, 葉奕廷, 蘇上育, 鄧逸軒, 葉浩同 (order is decided by python3.8 random.shuffle).





## 摘要

本論文主要嘗試研究目前深度學習技術對於自然語言理解的能力。所研究的理解能力主要包含兩者：第一是模型對於數理問題理解與推演的能力，第二則是模型理解對話的能力。以上兩者能力都是人類所具有的能力，但卻是還沒有被自然語言處理界使用深度學習深入調查的問題。

為了研究深度學習模型對於數理問題理解與推演的能力，本論文的第一部份著重在數學應用問題的解題任務之上。受到人類解決數學應用問題的方式的啟發，本篇論文設計了一個讓機器可以根據符號的語意產生算式的架構。結果證明，使用符號的語意果然可以增強機器的推演能力。

本論文的第二部份則著重於調查前人所提出的對話問答模型對於對話理解的能力。這部份主要關注兩個著名的對話問題資料集 QuAC 以及 CoQA。本篇論文在這個部份提出了一系列的實驗以檢驗模型理解的能力，並發現了一些潛在的問題。期望這些貢獻能夠有助於未來在這個方向的研究。

透過這兩方面的研究，本篇論文深入地探討了現有模型在語言理解能力上的不足。在數學應用問題的解題任務方面，仍然需要有更完善的方式檢驗模型泛化的能力。而在對話理解方面，如何設計出一個具有理解對話能力的模型也是一個未解的問題。這些都是未來的研究能夠繼續探討的方向。

關鍵字：語言理解、對話問答、數學應用問題解題





# Abstract

This work mainly attempt to investigate the natural language understanding capability of current deep learning models. The main understanding capabilities include two: first, the capability of language understanding and arithmetic reasoning capability for math word problems; second, the capability of conversation understanding. The above capabilities are possessed by human, but have not been well explored in the natural language processing filed with deep learning.

To investigate the arithmetic reasoning capability of deep learning models, the first part of this work focuses on the task of math work problems solving. Motivated by the solving process of human, this work proposes a framework that allows the model to generate math expressions by manipulating the symbols based on their semantics. The results show its effectiveness of improving the arithmetic reasoning capability.

The second part of this work investigates the conversation understanding capability of previous proposed models. Two renown datasets QuAC and CoQA are focused here. This part proposed a series of experiments that can serve as a tool to diagnose the conversation understanding capability of models, discovering some potential hazards.

By investigation in this two aspects, this work scrutinizes the incapacibilities of current models. For the task of math word problems solving, some more efforts are still required to validate the generalizability of current models. For the task of conversation understanding, the way to design a model

that understands conversations remains an unsolved problem. All of these are prospective future research directions.

**Keywords:** language understanding, conversational question answering, math word problem solving







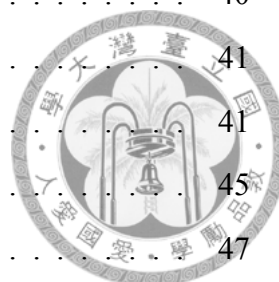
# Contents

<b>Acknowledgements</b>	<b>iii</b>
摘要	v
<b>Abstract</b>	<b>vii</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Motivation . . . . .	1
1.2 Problem Description . . . . .	4
1.3 Main Contributions . . . . .	4
1.4 Thesis Structure . . . . .	5
<b>2 Background</b>	<b>7</b>
2.1 Recurrent Neural Models . . . . .	7
2.1.1 Recurrent Neural Network (RNN) . . . . .	7
2.1.2 Long-Short Term Memory (LSTM) . . . . .	8
2.1.3 Bi-Directional Long-Short Term Memory (BiLSTM) . . . . .	8
2.2 Transformer Model . . . . .	9
2.3 Learning Objectives . . . . .	11
2.3.1 Maximum Likelihood for Seq2Seq Learning . . . . .	11
2.3.2 Maximum Likelihood for Answer Span Selection . . . . .	12
<b>3 Related Work</b>	<b>13</b>
3.1 Question Answering . . . . .	13

3.2	Arithmetic Reasoning . . . . .	13
3.3	Conversation Comprehension . . . . .	14
<b>4</b>	<b>Math Word Problem Solving</b>	<b>17</b>
4.1	Encoder . . . . .	18
4.2	Decoder . . . . .	19
4.3	Decoding State Features . . . . .	20
4.3.1	Stack Action Selector . . . . .	21
4.3.2	Stack Actions . . . . .	21
4.3.3	Operand Selector . . . . .	23
4.3.4	Semantic Transformer . . . . .	23
4.4	Training . . . . .	24
4.5	Inference . . . . .	24
4.6	Experiments . . . . .	25
4.6.1	Settings . . . . .	25
4.6.2	Results . . . . .	25
4.6.3	Ablation Test . . . . .	26
4.7	Qualitative Analysis . . . . .	28
4.7.1	Constant Embedding Analysis . . . . .	28
4.7.2	Decoding Process Visualization . . . . .	29
4.7.3	Error Analysis . . . . .	31
4.7.4	Discussion . . . . .	31
<b>5</b>	<b>Conversation Modeling</b>	<b>35</b>
5.1	Models . . . . .	37
5.1.1	FlowQA . . . . .	37
5.1.2	BERT . . . . .	38
5.1.3	SDNet . . . . .	38
5.2	How Well the Performance Reflects Content Comprehension? . . . . .	39
5.2.1	Experimental Settings . . . . .	39



5.2.2	Discussion . . . . .	40
5.3	Do Models Understand Conversation Content? . . . . .	41
5.3.1	Repeat Attack . . . . .	41
5.3.2	Predict without Previous Answer Text . . . . .	45
5.3.3	Predict without Previous Answer Position . . . . .	47
5.3.4	Implication of Above Experiments . . . . .	48
5.4	Dataset and Model Analysis . . . . .	48
5.4.1	Discussion . . . . .	50
<b>6</b>	<b>Conclusion and Future Work</b>	<b>51</b>
	<b>Bibliography</b>	<b>53</b>

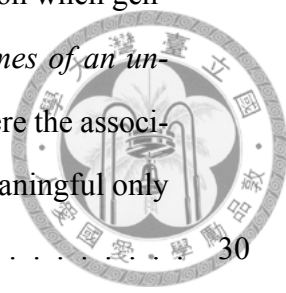






# List of Figures

1.1	The solving process of the math word problem “ <i>Each notebook takes \$0.5 and each pen takes \$1. Tom has \$10. How many notebook can he buy after buying 5 pens?</i> ” and the associated equation is $x = (10 - 1 \times 5) \div 0.5$ . The associated equation is $x = (10 - 1 \times 5) \div 0.5$ . . . . .	2
2.1	The illustration of the encoding part in the highway recurrent transformer. The input of multi-head attention module includes key, query, and value sequences; the bottom branches implies that $X$ is fed as the three parameters at the same time. . . . .	9
2.2	The illustration of the multi-head attention layer. Note that the input <i>Query, Key, Value</i> and the output are sequences of vectors. . . . .	10
4.1	The encoder-decoder model architecture of the proposed neural solver machine. . . . .	17
4.2	Illustration of the inference process. The purple round blocks denote the transformed semantics, while the green ones are generated by the variable generator. . . . .	19
4.3	The self-attention map visualization of operands’ semantic expressions for the problem “ <i>There are 58 bananas. Each basket can contain 6 bananas. How many bananas are needed to be token off such that exactly 9 baskets are filled?</i> ”. . . . .	28



4.4 Word attention and gate activation ( $g^{sa}$  and  $g^{opd}$ ) visualization when generating stack actions for the problem “6.75 deducting 5 times of an unknown number is 2.75. What is the unknown number?”, where the associated equation is  $x = (6.75 - 2.75) \div 5$ . Note that  $g^{opd}$  is meaningful only when the  $t$ -th stack action is push\_op. . . . . 30

5.1 The histogram of distance between answers to consecutive questions of a conversation in words. The bin size is 5 words. The distance is counted as zero if the two answer spans are overlapped, and is positive if the current is after the previous answer, negative otherwise. . . . . 36

5.2 Histogram of answer length distribution in QuAC and CoQA. The bin size is 1 word. . . . . 36

5.3 An example of repeat attack on QuAC and the model results. The red italic text is the inserted attack. . . . . 42

5.4 An example of repeat attack on CoQA. The red italic text is the inserted attack. In this example, the attack increases the distance between answer 2 and 3. . . . . 42

5.5 Histogram of distance between answers to consecutive questions in words *after* the attack. The bin size is 5. . . . . 43

5.6 F1 change between before and after repeat attack in terms of the answer’s distance to its previous answers on QuAC (left: FlowQA; right: BERT). The x-coordinate is the answer’s distance to the previous answer in words, and the meaning of zero, positive and negative is same as in Figure 5.1. The y-coordinate is the average F1 score. “no answer” sample are ignored here. The dotted line is the average F1 score of all samples. . . . . 43

5.7 An example of repeat attack on QuAC and the model prediction which BERT predicts *no answer*. . . . . 45

5.8 Relation between the F1 score before/after repeat attack and the answer’s distance to its previous answer on CoQA. (Left: FlowQA, Right: SDNet) 46



# List of Tables

4.1	5-fold cross validation results on Math23K. . . . .	26
4.2	5-fold cross validation results of ablation tests. . . . .	27
4.3	Randomly sampled incorrect predictions. . . . .	31
5.1	Model performance on the validation set of QuAC and CoQA. Note that the original SDNet model does not utilize the position information, so <i>SDNet + position - text</i> is a modified SDNet with additional one dimension feature indicating the previous answer as the input. . . . .	40
5.2	F1 score on the validation of QuAC and CoQA with or without attack. ”- <i>position</i> ” indicates training without the position information of answers to previous questions. . . . .	46
5.3	Impact on F1 score by the attack. . . . .	47
5.4	F1 results on the validation sets of QuAC and CoQA. Models are inferred without/with applying masks on the previous answers. Models without suffix are trained with full access to conversation, while “- <i>position</i> ” indicates that the models are trained without the previous answer position information. Note that in both training settings, masks are not applied. . . . .	47
5.5	F1 score on the validation sets of QuAC and CoQA. Models are inferred with/without access to position information, but trained with access to position information. . . . .	48
5.6	F1 score of models on shuffled validation set of CoQA. Models here do not use the position information of the previous answers. . . . .	49
5.7	An example of random shuffled CoQA passage. . . . .	49







# Chapter 1

## Introduction

### 1.1 Motivation

Machines' capability of understanding natural language is always an important scientific topic. The powerful deep learning technology has created many possibilities in the natural language process field. Till now, deep learning models have achieved remarkable performance in several natural language processing tasks, from linguistic tasks, such as POS tagging, sentence parsing, to real world applications, including chat bot systems, question answering systems, etc. However, it is not sufficient to assert those models' capability of language understanding solely based on their appealing performance on those tasks. It is questionable to what degree those models understand the language.

To investigate machines' capability of language understanding from holistic aspects, this work focuses on the question answering tasks requiring capability of arithmetic reasoning and conversation modeling. Specifically, this work focuses on two tasks: 1) math word problems solving and 2) conversational question answering. The two tasks require very different sets of natural language understanding skills, and both of them have not been well explored before. It is anticipated that by investigation on these two very different tasks, better knowledge of machines' capability can be acquired.

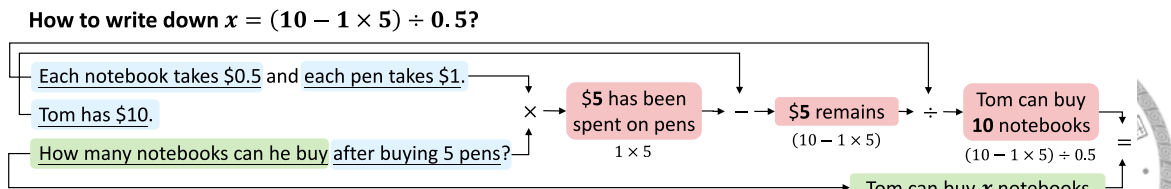


Figure 1.1: The solving process of the math word problem “Each notebook takes \$0.5 and each pen takes \$1. Tom has \$10. How many notebook can he buy after buying 5 pens?” and the associated equation is  $x = (10 - 1 \times 5) \div 0.5$ . The associated equation is  $x = (10 - 1 \times 5) \div 0.5$ .

## Math Word Problem Solving

Automatically solving math word problems has been viewed as a way of evaluating machines’ ability [1] of both language understanding and mathematical reasoning. For human, writing down an equation that solves a math word problem requires the ability of reading comprehension, reasoning, and sometimes real world understanding. Specifically, to solve a math word problem, we first need to know the goal of the given problem, then understand the semantic meaning of each numerical number in the problem, perform reasoning based on the comprehension in the previous step, and finally decide what to write in the equation. Therefore, this task should be an ideal test stone to explore machines’ language understanding capability.

Most prior work about solving math word problems relied on hand-crafted features, which required more human knowledge. Because those features are often in the lexical level, it is not clear whether machines really understand the math problems. Also, most prior work evaluated their approaches on relatively small datasets, and the capability of generalization is concerned.

To investigate model’s capability of language understanding, this work considers the semantic of mathematical symbols involved in the reasoning procedure. Figure 1.1 illustrates a reasoning process that solves a math problem. The illustration shows that human actually assigns the semantic meaning to each number when manipulating symbols, including operands (numbers) and operators (+ − × ÷). The semantic meaning of operands may be helpful when deciding the operator to use. For example, the summation of “price of one pen” and “number of pens Tom bought” is meaningless; therefore the ad-

dition would not be chosen. Beholding this observation, it is worthy of investigation to see if machine can also solve math word problems in this way. Therefore, in this work an end-to-end math word problem solving approach is designed.



## Conversation Modeling

Answering questions in the conversational manner requires special language understanding skills. Different from traditional machine reading comprehension [2, 3, 4] whose questions are context-free, questions and answers in QuAC and CoQA are collected in a conversational manner. Same as in general machine reading comprehension tasks, questions related to a given passage are asked. However, questions may be also related to the given conversational history, and should be answered accordingly. Such conversational setting is regarded as more practical because people tend to seek for information in a conversational way. This work focus on two benchmark conversational question answering datasets, QuAC [5] and CoQA [6]. They feature many linguistic phenomena unique to conversations, so they are believed to be important materials for investigation of language understanding in conversation.

This part of work focuses on investigating *how well the performance of a model on these two benchmark datasets reflects its capability of comprehension*. If higher performance on these datasets does not necessarily imply better conversation comprehension, then further investigation must be done when models claim their better understanding performance. However, it has not been well investigated by any of the prior work [5, 7, 8, 9].

We further analyze *whether the recent models achieving competitive performance rely on content comprehension*. It is motivated by the fact that the position of the answer to the previous question is widely utilized in many of previous conversational question answering models, such as BiDAF-with-ctx [10] and FlowQA [7]. Those models leverage the datasets' property that answers or rationals can always be found as a span of the given passage. Thus those models can access the informative content provided by the position information. Models are expected to learn to understand the conversation based on its content. However, it is not clear whether the models rely on the previous conversation

content or merely the position information.



## 1.2 Problem Description

With regard to the above discussion, in this work two questions are subject to investigation:

1. How can machines do arithmetic reasoning over natural language as human do?
2. How do the previous models understand the language in conversations?

## 1.3 Main Contributions

### Math Word Problem Solving

Contributions in this work include

- This work is the first attempt to model semantic meanings of operands and operators for math word problems.
- This work proposes an end-to-end neural math solver with a novel decoding process that utilizes the stack to generate associated equations.
- The proposed architecture achieves the state-of-the-art performance on the large benchmark dataset Math23K.
- The proposed architecture is capable of providing interpretation and reasoning for the math word problem solving procedure.

### Conversational Question Answering

- Series of experiments are designed. They can serve as an analysis tool for conversational question answering models in the future.
- We identify potential hazards in the conversation question answering task:

- Higher performance on QuAC and CoQA does not necessarily imply better content comprehension.
- Models trained on QuAC show the tendency of heavily relying on the previous answers’ positions rather than their textual content.



## 1.4 Thesis Structure

In the follow chapters, technical backgrounds will be reviewed in the Chapter 2, and related work will be reviewed in the Chapter 3. Then the arithmetic reasoning capability will be investigated in Chapter 4, which has been presented in [11]: Ting-Rui Chiang and Yun-Nung Chen, “Semantically-Aligned Equation Generation for Solving and Reasoning Math Word Problems,” in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)* pages 2656–2668. The conversation comprehension capability will be examined in the Chapter 5, part of which will be published in [12]: Ting-Rui Chiang, Hao-Tong Ye, and Yun-Nung Chen “An Empirical Study of Content Understanding in Conversational Question Answering,” in *Proceedings of the AAI Conference on Artificial Intelligence (AAAI)*. Finally, the Chapter 6 concludes this work.





# Chapter 2

## Background

In this chapter, some background about the building blocks of models in this work, as well as the models analyzed will be introduced.

### 2.1 Recurrent Neural Models

Recurrent neural models are introduced to model a sequence of vectors. Its recurrent structure equips it with the potential to encode the order information of the input. They are widely used in previous work as well as this work.

#### 2.1.1 Recurrent Neural Network (RNN)

The basic ideal of recurrent model is first embodied in the recurrent neural network [13]. For a given series of input vectors  $(x_1, x_2, \dots, x_T)$ , the representation  $h_t$  of  $x_t$  is encoded with a recurrent formula

$$h_t = \sigma(W_h x_t + U_h h_{t-1} + b_h), \quad (2.1)$$

where  $\sigma$  is typically a sigmoid function, and  $W_h, U_h, b_h$  are parameters to train.

## 2.1.2 Long-Short Term Memory (LSTM)

The long-short term memory model [14] is an improved version of RNN that fixes its gradient vanish problem. For a RNN, when calculating the gradient of the outputs with respect to the parameters, the scale of the gradient can diminish exponentially. To fix this problem, gates are added in the design of the long-short memory model. Specifically, it has three gates:

$$\begin{aligned}f_t &= \sigma(W_f x_t + U_f h_{t-1} + b_f) \\i_t &= \sigma(W_i x_t + U_i h_{t-1} + b_i) \\o_t &= \sigma(W_o x_t + U_o h_{t-1} + b_o) \\c_t &= f_t \circ c_{t-1} + i_t \circ \tanh(W_c x_t + U_c h_{t-1} + b_c) \\h_t &= o_t \circ \tanh(c_t),\end{aligned}$$

where  $W_f, W_i, W_o, W_c, U_c, b_f, b_i, b_o, b_c$  are parameters to learn. Generally the hidden state  $h_t$  is used as the representation of input  $x_t$  encoded by this long-short term memory model, and is used by following neural network layers.

## 2.1.3 Bi-Directional Long-Short Term Memory (BiLSTM)

A bi-directional long-short term memory model consists of two LSTM models. Each LSTM model encode each input vector  $x_t$  in the input sequence in one direction into  $\overleftarrow{h}_t$  and  $\overrightarrow{h}_t$  respectively. And then the bi-directional representation  $\overleftrightarrow{h}_t$  of  $x_t$  is formed by concatenating the two vectors. This way of encoding gives  $\overleftrightarrow{h}_t$  the potential to contain information of all input vectors, and thus enrich the information in the encoded representation.



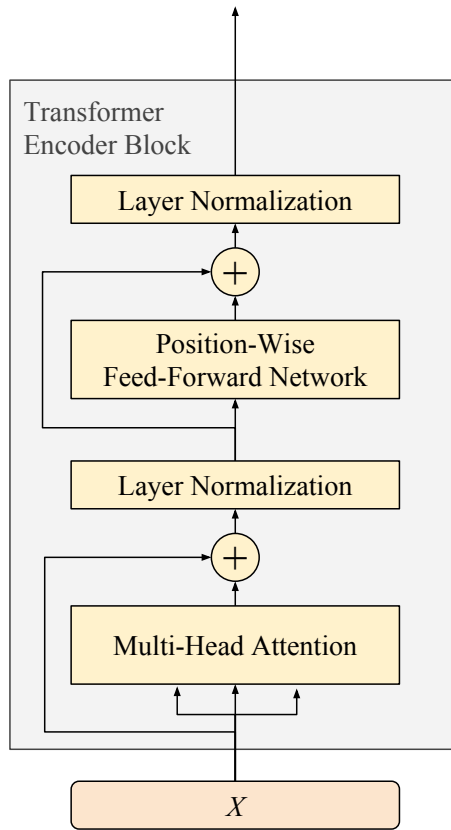


Figure 2.1: The illustration of the encoding part in the highway recurrent transformer. The input of multi-head attention module includes key, query, and value sequences; the bottom branches implies that  $X$  is fed as the three parameters at the same time.

## 2.2 Transformer Model

A transformer encoder block (Figure 2.1) proposed in [15] consists of a multi-head attention layer and a position-wise feed-forward network, residual connection and layer normalization are used to connect the two components, details are specified as follows.

### Multi-Head Attention Layer

A multi-head attention [15] (Figure 2.2) consists of the heads of attention, each head performs linear transformation before performing attention operation; with different sets of trainable parameters, each attention head potentially models different relationship between two sequences. Specifically, the inputs of the multi-head attention layer are three sequences of vectors: *query*  $Q \in \mathbb{R}^{l_1 \times d_f}$ , *key*  $K \in \mathbb{R}^{l_2 \times d_f}$ , *value*  $V \in \mathbb{R}^{l_2 \times d_f}$ , where  $l_1, l_2$  are the length of the first and second sequence respectively. Then for the  $h$ -th head, three weight matrices  $W^{Qh}, W^{Kh}, W^{Vh} \in \mathbb{R}^{d_f \times d_p}$  are used to project the three inputs to a lower

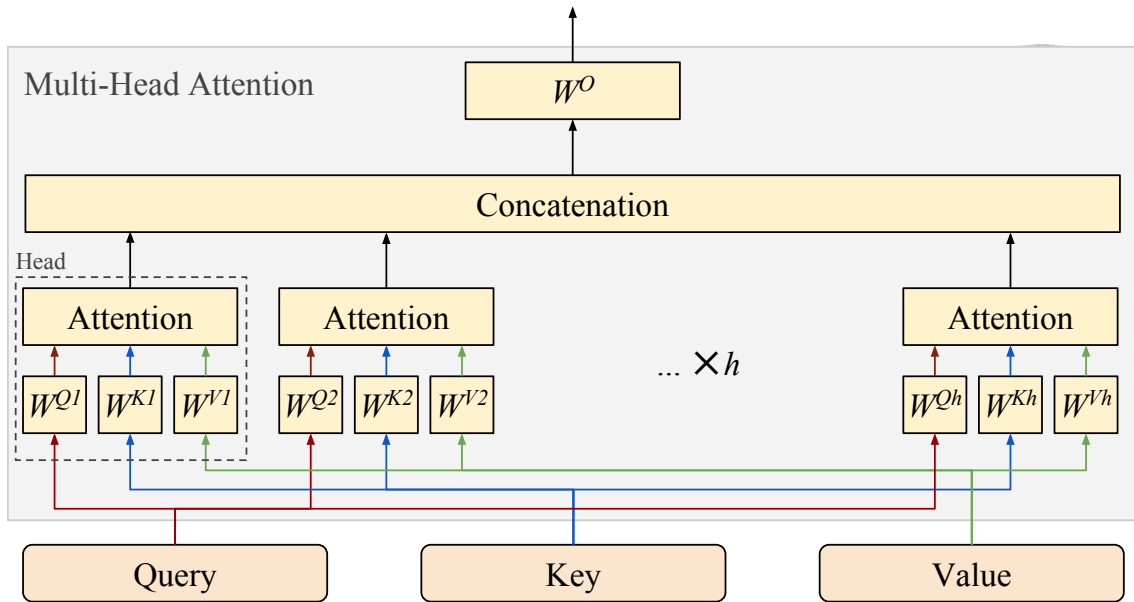


Figure 2.2: The illustration of the multi-head attention layer. Note that the input *Query*, *Key*, *Value* and the output are sequences of vectors.

dimension  $d_p$ , and then an attention function is performed

$$A^h = \text{Attention}(Q^h, K^h, V^h),$$

where  $Q^h = QW^{Qh}$ ,  $K^h = KW^{Kh}$ ,  $V^h = VW^{Vh}$ . The attention function generates a vector for each vector in the query sequence  $Q$ . Let the outputs of the attention function be  $A^h \in \mathbb{R}^{l_1 \times d_p}$ , which is weighted sum of value  $V$  based on similarity matrices  $S^h$ . For  $a = 1, 2, \dots, l_1$ , the  $a$ -th output is calculated as below:

$$S^h = Q^h(K^h)^T,$$

$$A_a^h = \sum_{p=1}^{l_2} \frac{\exp s_{a,p}^h}{\sum_{t=1}^{l_2} \exp s_{a,t}^h} V_p^h, \quad (2.2)$$

where  $s$  are the similarity scores in the similarity matrix  $S^h$ .

Then the output of the multi-head attention is the linear transformed concatenation of the outputs from attention heads:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(A^1, A^2, \dots, A^H)W^O$$

where  $H$  is the number of heads, and  $W^O \in W^{H \cdot d_p \times d_f}$  is a trained weight matrix.

### Position-Wise Feed-Forward Network

The position-wise feed-forward network (FFN) transforms each vector in a sequence identically as follows:

$$\text{FFN}(X) = \max(0, XW_1 + b_1)W_2 + b_2$$

### Residual Connection and Layer Normalization

Then the above two components are connected with residual connection and layer normalization [16]:

$$\text{ResiNorm}(f, X) = \text{LayerNorm}(X + f(X)), \quad (2.3)$$

$$\text{TransformerBlock}(X) = \text{ResiNorm}(\text{FFN}, \text{ResiNorm}(\text{MultiHead}, (X, X, X))).$$

Note that here we use the same sequence for the query, key, value arguments of the multi-head attention for self-attention.

## 2.3 Learning Objectives

### 2.3.1 Maximum Likelihood for Seq2Seq Learning

Maximum likelihood is used in the learning objective function for conditional sequence generation. Given an input sequence  $x_1, x_2, \dots, x_{T_{in}}$  and a target output sequence  $y_1^*, y_2^*, \dots, y_{T_{out}}^*$ , the learning objective is to maximize the sum of the factorized conditional log probability:

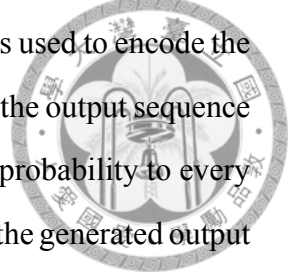
$$\log P(y_1^*, y_2^*, \dots, y_{T_{out}}^* | x_1, x_2, \dots, x_{T_{in}}) \quad (2.4)$$

$$= \log \prod_{t=1}^{T_{out}} P(y_t^* | y_{t-1}^*, y_{t-2}^*, \dots, y_1^*, x_1, x_2, \dots, x_{T_{in}}) \quad (2.5)$$

$$= \sum_{t=1}^{T_{out}} \log P(y_t^* | y_{t-1}^*, y_{t-2}^*, \dots, y_1^*, x_1, x_2, \dots, x_{T_{in}}) \quad (2.6)$$



In this work, the conditional probability in equation 2.6 is modeled by an encoder-decoder seq2seq model. Specifically, in a seq2seq model, an encoder is used to encode the input sequence into a representation, and a decoder is used to generate the output sequence one by one in order. At the decoding time step  $t$ , the decoder assigns probability to every output candidate according to the representation of the input as well as the generated output before time  $t$ . The training objective is to maximize the probability.



### 2.3.2 Maximum Likelihood for Answer Span Selection

Maximum likelihood is used in the training objective function for answer span selection. Generally, answer span selection is formulated as a task to select the start point and the end point of the answer in a passage. Models considered in this work select the start point and the end point conditionally independently given the passage. Therefore, given a passage  $X$ , and the target start point  $p_{start}$ , end point  $p_{end}$ , the objective is to maximize

$$\log P(p_{start}, p_{end}|X) \tag{2.7}$$

$$= \log P(p_{start}|X)P(p_{end}|X) \tag{2.8}$$

$$= \log P(p_{start}|X) + \log P(p_{end}|X). \tag{2.9}$$

In this work,  $P(p_{start}|X)$  and  $\log P(p_{end}|X)$  are modeled by the conversation comprehension models.



# Chapter 3

## Related Work

### 3.1 Question Answering

The task of question answering has attracted lots of interests in recent years. It is believed that all of the natural language processing tasks can be reduced to question answering problems. Therefore, many datasets [2, 4, 3, 17, 18, 19, 20, 21] have been created, and there has been many models [22, 23, 24, 25, 26, 27] crafted for advancing the tasks.

### 3.2 Arithmetic Reasoning

There is a lot of prior work that utilized hand-crafted features, such as POS tags, paths in the dependency trees, keywords, etc., to allow the model to focus on the quantities in the problems [28, 29, 30, 31, 32, 33, 34, 35, 36, 37]. Recently, [38, 39, 40] attempted at learning models without predefined features.

[28] first extracted templates about math expressions from the training answers, and then trained models to select templates and map quantities in the problem to the slots in the template. Such two-stage approach has been tried and achieved good results [35]. The prior work highly relied on human knowledge, where they parsed problems into equations by choosing the expression tree with the highest score calculated by an operator classifier, working on a hand-crafted “trigger list” containing quantities and noun phrases in the problem, or utilizing features extracted from text spans [30, 33, 32]. [41] defined a

Dolphin language to connect math word problems and logical forms, and generated rules to parse math word problems. [34] parsed math word problems without explicit equation annotations. [36] classified math word problems into 4 types and used rules to decide the operators accordingly. [37] trained the parser using reinforcement learning with hand-crafted features. [29] modeled the problem text as transition of world states, and the equation is generated as the world states changing. Our work uses a similar intuition, but hand-crafted features are not required and our model can be trained in an end-to-end manner. Some end-to-end approaches have been proposed, such as generating equations directly via a seq2seq model [39]. [40] tried to generate solutions along with its rationals with a seq2seq-like model for better interpretability.

### 3.3 Conversation Comprehension

Recently, with the trend of conversational interactions, people started to focus on conversational understanding. Therefore, datasets for conversational question answering were built [5, 6], and the corresponding models [7, 8] were proposed to tackle the challenges.

Investigations on what machine comprehension models learn are mainly for those context-free question answering systems. [42] proposed a method to generate adversarial examples in order to test the model robustness. [43] conducted experiments to verify the reading required to answer questions in the dataset. [23] found a feature indicating if a word appears in the question important, and suggested that questions can be answered with some rules that rely only on superficial features. [44] validated the suggestion by a series of systematic experiments. However, the conversational question answering systems are rarely explored. Although [45] compared CoQA, SQuAD 2.0 and QuAC qualitatively, there was no any investigation on what conversational question answering models capture.

The phenomenon that models does not utilize all useful features is common in diverse areas. For example, [46] showed that natural language inference models with high accuracy relied much on the lexical-level features but utilized little compositional semantics. [47] found that not whole conversation history is used in neural dialogue generation sys-

tems. Similar phenomena happened in the computer vision area, where [48] indicated that CNN trained on ImageNet relied much on textual information rather than shape information, and [49] further showed that only textual information can achieve very high accuracy on ImageNet.









# Chapter 4

## Math Word Problem Solving

To investigate the reasoning capability of machines, a model composed of an encoder and a decoder is proposed. The model is designed by viewing the process of solving math word problems as transforming multiple text spans in the problems into the target information the problems ask for. In the example shown in Figure 1.1, all numbers in the problem are attached with the associated semantics. Motivated by the observation, we design an encoder to extract the semantic representation of each number in the problem text. Considering that human usually manipulates those numbers and operators (such as addition, subtraction, etc.) based on their semantics for problem solving, a decoder is designed to construct the equation, where the semantics is aligned with the representations extracted by the encoder. The idea of the proposed model is to imitate the human reasoning process for solving math word problems. The model architecture is illustrated in Figure 4.1.

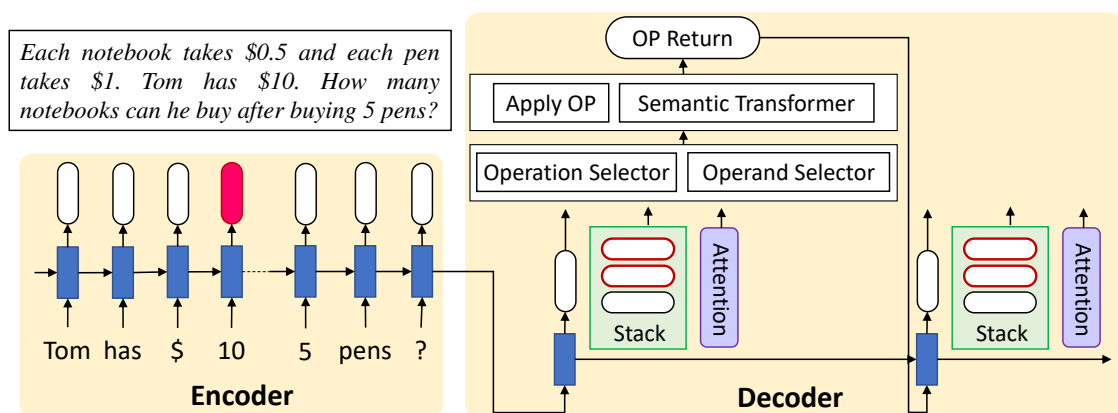


Figure 4.1: The encoder-decoder model architecture of the proposed neural solver machine.

## 4.1 Encoder

The encoder aims to extract the semantic representation of each constant needed for solving problems. However, the needed constants may come from either the given problem texts or domain knowledge, so we detail these two procedures as follows.



### Constant Representation Extraction

For each math word problem, we are given a passage consisting of words  $\{w_t^P\}_{t=1}^m$ , whose word embeddings are  $\{e_t^P\}_{t=1}^m$ . The problem text includes some numbers, which we refer as constants. The positions of constants in the problem text are denoted as  $\{p_i\}_{i=1}^n$ . In order to capture the semantic representation of each constant by considering its contexts, a bidirectional long short-term memory (BLSTM) is adopted as the encoder [14]:

$$h_t^E, c_t^E = \text{BLSTM}(h_{t-1}^E, c_{t-1}^E, e_t^P), \quad (4.1)$$

and then for the  $i$ -th constant in the problem, its semantic representation  $e_i^c$  is modeled by the corresponding BLSTM output vector:

$$e_i^c = h_{p_i}^E. \quad (4.2)$$

### External Constant Leveraging

External constants, including 1 and  $\pi$ , are leveraged, because they are required to solve a math word problem, but not mentioned in the problem text. Due to their absence from the problem text, we cannot extract their semantic meanings by BLSTM in (4.2). Instead, we model their semantic representation  $e^\pi, e^1$  as parts of the model parameters. They are randomly initialized and are learned during model training.

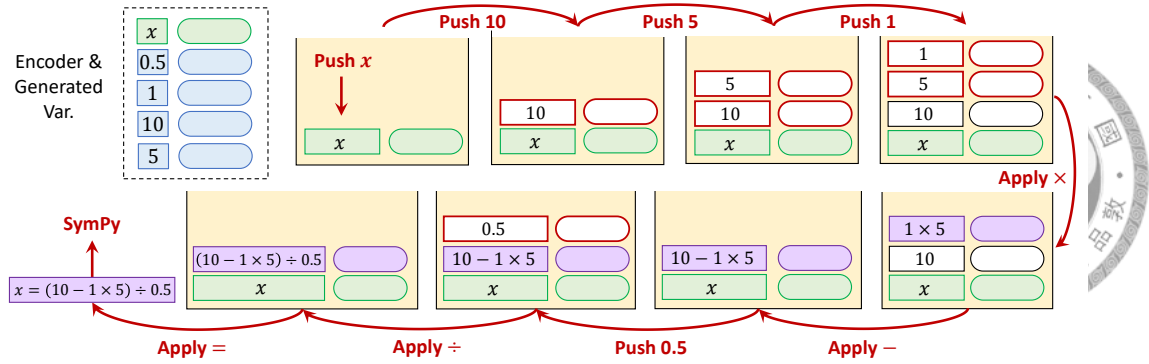


Figure 4.2: Illustration of the inference process. The purple round blocks denote the transformed semantics, while the green ones are generated by the variable generator.

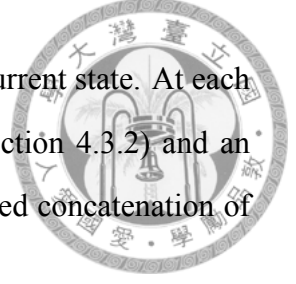
## 4.2 Decoder

The decoder aims at constructing the equation that can solve the given problem. We generate the equation by applying stack actions on a stack to mimic the way how human understands an equation. Human knows the semantic meaning of each term in the equation, even compositing of operands and operators like the term ”(10 – 1 × 5)” in Figure 1.1. Then what operator to apply on a pair operands can be chosen based on their semantic meanings accordingly. Hence we design our model to generate the equation in a postfix manner: a operator is chosen base on the semantic representations of two operands the operator is going to apply to. Note that the operands a operator can apply to can be any results generated previously. That is the reason why we use “stack” as our data structure in order to keep track of the operands a operator is going to apply to. The stack contains both symbolic and semantic representations of operands, denoted as

$$S = [(v_{l_t}^S, e_{l_t}^S), (v_{l_t-1}^S, e_{l_t-1}^S), \dots, (v_1^S, e_1^S)], \quad (4.3)$$

where  $v^S$  of each pair is the symbolic part, such as  $x + 1$ , while  $e^S$  is the semantic representation, which is a vector. The components in the decoder are shown in the right part of Figure 4.1, each of which is detailed below.

### 4.3 Decoding State Features



At each decoding step, decisions are made based on features of the current state. At each step, features  $r^{sa}$  and  $r^{opd}$  are extracted to select a stack action (section 4.3.2) and an operand to push (section 4.3.3). Specifically, the features are the gated concatenation of following vectors:

- $h_t^D$  is the output of an LSTM, which encodes the history of applied actions:

$$h_t^D, c_t^D = \text{LSTM}(h_{t-1}^D, c_{t-1}^D, \text{res}_{t-1}), \quad (4.4)$$

where  $\text{res}_{t-1}$  is the result from the previous stack action similar to the seq2seq model [50]. For example, if the previous stack action  $o_{t-1}$  is “push”, then  $\text{res}_{t-1}$  is the semantic representation pushed into the stack. If the previous stack action  $o_{t-1}$  is to apply an operator  $\diamond$ , then  $\text{res}_{t-1}$  is the semantic representation generated by  $f_\diamond$ .

- $s_t$  is the stack status. It is crucial because some operators are only applicable to certain combinations of operand semantics, which is similar to the type system in programming languages. For example, operating multiplication is applicable to the combination of “*quantity of an item*” and “*price of an item*”, while operating addition is not. Considering that all math operators supported here (+, −, ×, ÷) are binary operators, the semantic representations of the stack’s top 2 elements at the time  $t - 1$  are considered:

$$s_t = [e_t^S; e_t^S]. \quad (4.5)$$

- $q_t$  incorporates problem information in the decision. It is believed that the attention mechanism [51] can effectively capture dependency for longer distance. Thus, the attention mechanism over the encoding problem  $h_1^E, h_2^E, \dots$  is adopted:

$$q_t = \text{Attention}(h_t^D, \{h_i^E\}_{i=1}^m), \quad (4.6)$$

where the attention function in this paper is defined as a function with learnable

parameters  $w, W, b$ :

$$\text{Attention}(u, \{v_i\}_{i=1}^m) = \sum_{i=1}^m \alpha_i h_i, \quad (4.7)$$

$$\alpha_i = \frac{\exp(s_i)}{\sum_{l=1}^m \exp(s_l)}, \quad (4.8)$$

$$s_i = w^T \tanh(W^T [u; v_i] + b). \quad (4.9)$$



In order to model the dynamic features for different decoding steps, features in  $r_t^{sa}$  is gated as follows:

$$r_t^{sa} = [g_{t,1}^{sa} \cdot h_t^D; g_{t,2}^{sa} \cdot s_t; g_{t,3}^{sa} \cdot q_t], \quad (4.10)$$

$$g_t^{sa} = \sigma(W^{sa} \cdot [h_t^D; s_t; q_t]), \quad (4.11)$$

where  $\sigma$  is a sigmoid function and  $W^{sa}$  is a learned gating parameter.  $r_t^{opd}$  is defined similarly, but with a different learned gating parameter  $W^{opd}$ .

### 4.3.1 Stack Action Selector

The stack action selector is to select an stack action at each decoding step (section 4.3.2) until the unknowns are solved. The probability of choosing action  $a$  at the decoding step  $t$  is calculated with a network NN constituted of one hidden layer and ReLU as the activation function:

$$\begin{aligned} P(Y_t | \{y_i\}_{i=1}^{t-1}, \{w_i\}_{i=1}^m) &= \text{StackActionSelector}(r_t^{sa}) \\ &= \text{softmax}(\text{NN}(r_t^{sa})), \end{aligned}$$

where  $r_t^{sa}$  is decoding state features as defined in section 4.3.

### 4.3.2 Stack Actions

The available stack actions are listed below:

- **Variable generation:** The semantic representation of an unknown variable  $x$  is generated dynamically as the first action in the decoding process. Note that this procedure provides the flexibility of solving problems with more than one unknown variables. The decoder module can decide how many unknown variables are required to solve the problem, and the semantic representation of the unknown variable is generated with an attention mechanism:

$$e^x = \text{Attention}(h_t^D, \{h_i^E\}_{i=1}^m). \quad (4.12)$$

- **Push:** This stack action pushes the operand chosen by the operand selector (section 4.3.3). Both the symbolic representation  $v_*$  and semantic representation  $e_*$  of the chosen operand would be pushed to the stack  $S$  in (4.3). Then the stack state becomes

$$S = [(v_*^S, e_*^S), (v_t^S, e_t^S), \dots, (v_1^S, e_1^S)]. \quad (4.13)$$

- **Operator  $\diamond$  application** ( $\diamond \in \{+, -, \times, \div\}$ ): One stack action pops two elements from the top of the stack, which contains two pairs,  $(v_i, e_i)$  and  $(v_j, e_j)$ , and then the associated symbolic operator,  $v_k = v_i \diamond v_j$ , is recorded. Also, a semantic transformation function  $f_\diamond$  for that operator is invoked, which generates the semantic representation of  $v_k$  by transforming semantic representations of  $v_i$  and  $v_j$  to  $e_k = f_\diamond(e_i, e_j)$ . Therefore, after an operator is applied to the stack specified in (4.3), the stack state becomes

$$S = [(v_t^S \diamond v_{t-1}^S, f_\diamond(e_t^S, e_{t-1}^S)), (v_{t-2}^S, e_{t-2}^S), \dots, (v_1^S, e_1^S)].$$

- **Equal application:** When the equal application is chosen, it implies that an equation is completed. This stack action pops 2 tuples from the stack,  $(v_i, e_i)$ ,  $(v_j, e_j)$ , and then  $v_i = v_j$  is recorded. If one of them is an unknown variable, the problem is solved. Therefore, after an OP is applied to the stack specified in (4.3), the stack

state becomes

$$S = [(v_{t-2}^S, e_{t-2}^S), \dots, (v_1^S, e_1^S)]. \quad (4.14)$$



### 4.3.3 Operand Selector

When the stack action selector has decided to push an operand, the operand selector aims at choosing which operand to push. The operand candidates  $e$  include constants provided in the problem text whose semantic representations are  $e_1^c, e_2^c, \dots, e_n^c$ , unknown variable whose semantic representation is  $e^x$ , and two external constants 1 and  $\pi$  whose semantic representations are  $e^1, e^\pi$ :

$$e = [e_1^c, e_2^c, \dots, e_n^c, e^1, e^\pi, e^x]. \quad (4.15)$$

An operand has both symbolic and semantic representations, but the selection focuses on its semantic meaning; this procedure is the same as what human does when solving math word problems.

Inspired by addressing mechanisms of neural Turing machine (NTM) [52], the probability of choosing the  $i$ -th operand candidate is the attention weights of  $r_t$  over the semantic representations of the operand candidates as in (4.8):

$$\begin{aligned} P(Z_t \mid \{y_i\}_{i=1}^{t-1}, \{w_i\}_{i=1}^m) &= \text{OperandSelector}(r_t^{opd}) \\ &= \text{AttentionWeight}(r_t^{opd}, \{e_i\}_{i=1}^m \cup \{e^1, e^\pi, e^x\}), \end{aligned}$$

and  $r_t^{opd}$  is defined in section 4.3.

### 4.3.4 Semantic Transformer

A semantic transformer is proposed to generate the semantic representation of a new symbol resulted from applying an operator, which provides the capability of interpretation and reasoning for the target task. The semantic transformer for an operator  $\diamond \in \{+, -, \times, \div\}$

transforms semantic representations of two operands  $e_1, e_2$  into

$$f_{\diamond}(e_1, e_2) = \tanh(U_{\diamond} \text{ReLU}(W_{\diamond}[e_1; e_2] + b_{\diamond}) + c_{\diamond}), \quad (4.16)$$

where  $W_{\diamond}, U_{\diamond}, b_{\diamond}, c_{\diamond}$  are model parameters. Semantic transformers for different operators have different parameters in order to model different transformations.



## 4.4 Training

Both stack action selection and operand selection can be trained in a fully supervised way by giving problems and associated ground truth equations. Because our model generates the equation with stack actions, the equation is first transformed into its postfix representation. Let the postfix representation of the target equation be  $y_1, \dots, y_t, \dots, y_T$ , where  $y_t$  can be either an operator ( $+, -, \times, \div, =$ ) or a target operand. Then for each time step  $t$ , the loss can be computed as

$$L(y_t) = \begin{cases} L_1(\text{push\_op}) + L_2(y_t) & y_t \text{ is an operand} \\ L_1(y_t) & \text{otherwise} \end{cases},$$

where  $L_1$  is the stack action selection loss and  $L_2$  is the operand selection loss defined as

$$L_1(y_t) = -\log P(Y_t = y_t \mid \{o_i\}_{i=1}^{t-1}, \{w_i\}_{i=1}^m),$$

$$L_2(y_t) = -\log P(Z_t = y_t \mid r_t).$$

The objective of our training process is to minimize the total loss for the whole equation,  $\sum_{t=1}^T L(y_t)$ .

## 4.5 Inference

When performing inference, at each time step  $t$ , the stack action with the highest probability  $P(Y_t \mid \{\tilde{y}_i\}_{i=1}^{t-1}, \{w_i\}_{i=1}^m)$  is chosen. If the chosen stack action is “*push*”, the operand



with the highest probability  $P(Z_t|\{\tilde{Y}_i\}_{i=1}^{t-1}, \{w_i\}_{i=1}^m)$  is chosen. When the stack has less than 2 elements, the probability of applying operator  $+, -, \times, \div, =$  would be masked out to prevent illegal stack actions, so all generated equations must be legal math expressions. The decoder decodes until the unknown variable can be solved. After the equations are generated, a Python package SymPy [53] is used to solve the unknown variable. The inference procedure example is illustrated in Figure 4.2. The detailed algorithm can be found in Algorithm 1.

## 4.6 Experiments

To evaluate the performance of the proposed model, we conduct the experiments on the benchmark dataset and analyze the learned semantics.

### 4.6.1 Settings

The experiments are benchmarked on the dataset Math23k [39], which contains 23,162 math problems with annotated equations. Each problem can be solved by a single-unknown-variable equation and only uses operators  $+, -, \times, \div$ . Also, except  $\pi$  and 1, quantities in the equation can be found in the problem text. There are also other large scale datasets like Dolphin18K [41] and AQuA [40], containing 18,460 and 100,000 math word problems respectively. The reasons about not evaluating on these two datasets are 1) Dolphin18k contains some unlabeled math word problems and some incorrect labels, and 2) AQuA contains rational for solving the problems, but the equations in the rational are not formal (e.g. mixed with texts, using  $x$  to represent  $\times$ , etc.) and inconsistent. Therefore, the following experiments are performed and analyzed using Math23K, the only large scaled, good-quality dataset.

### 4.6.2 Results

The results are shown in Table 4.1. The retrieval-based methods compare problems in test data with problems in training data, and choose the most similar one's template to solve

Model		Accuracy
Retrieval	Jaccard	47.2%
	Cosine	23.8%
Classification	BLSTM	57.9%
	Self-Attention	56.8%
Generation	Seq2Seq w/ SNI	58.1%
	Proposed Word-Based	<b>65.3%</b>
	Proposed Char-Based	<b>65.8%</b>
Hybrid	Retrieval + Seq2Seq	64.7%

Table 4.1: 5-fold cross validation results on Math23K.

the problem [28, 35]. The classification-based models choose equation templates by a classifier trained on the training data. Their performance are reported in [54]. The seq2seq and hybrid models are from [39], where the former directly maps natural language into symbols in equations, and the latter one ensembles prediction from a seq2seq model and a retrieval-based model. The ensemble is the previous state-of-the-art results of Math23K.

Our proposed end-to-end model belongs to the generation category, and the single model performance achieved by our proposed model is new state-of-the-art ( $> 65\%$ ) and even better than the hybrid model result (64.7%). In addition, we are the first to report character-based performance on this dataset, and the character-based results are slightly better than the word-based ones. Among the single model performance, our models obtain about more than 7% accuracy improvement compared to the previous best one [39]. The performance of our character-based model also shows that our model is capable of learning the relatively accurate semantic representations without word boundaries and achieves better performance.

### 4.6.3 Ablation Test

To better understand the performance contributed by each proposed component, we perform a series of ablation tests by removing components one by one and then checking the performance by 5-fold cross validation. Table 4.2 shows the ablation results.

**Char-Based v.s. Word-Based** As reported above, using word-based model instead of character-based model only causes 0.5% performance drop. To fairly compare with prior

Model	Accuracy
Char-Based	65.8%
Word-Based	65.3%
Word-Based - Gate	64.1%
Word-Based - Gate - Attention	62.5%
Word-Based - Gate - Attention - Stack	60.1%
Word-Based - Semantic Transformer	64.1%
Word-Based - Semantic Representation	61.7%

Table 4.2: 5-fold cross validation results of ablation tests.

word-based models, the following ablation tests are performed on the word-based approach.

**Word-Based - Gate** It uses  $r_t$  instead of  $r_t^{sa}$  and  $r_t^{opr}$  as the input of both StackActionSelector and OperandSelector.

**Word-Based - Gate - Attention** Considering that the prior generation-based model (seq2seq) did not use any attention mechanism, we compare the models with and without the attention mechanism. Removing attention means excluding  $q_{t-1}$  in (4.11), so the input of both operator and operand selector becomes  $r_t = [h_t^D; s_t]$ . The result implies that our model is not better than previous models solely because of the attention.

**Word-Based - Gate - Attention - Stack** To check the effectiveness of the stack status ( $s_t$  in (4.11)), the experiments of removing the stack status from the input of both operator and operand selectors ( $r_t = h_t^D$ ) are conducted. The results well justify our idea of choosing operators based on semantic meanings of operands.

**Word-Based - Semantic Transformer** To validate the effectiveness of the idea that views an operator as a semantic transformer, we modify the semantic transformer function of the operator  $\diamond$  into  $f_\diamond(e_1, e_2) = e_\diamond$ , where  $e_\diamond$  is a learnable parameter and is different for different operators. Therefore,  $e_\diamond$  acts like the embedding of the operator  $\diamond$ , and the decoding process is more similar to a general seq2seq model. The results show that the semantic transformer in the original model encodes not only the last operator applied on the operands but other information that helps the selectors.

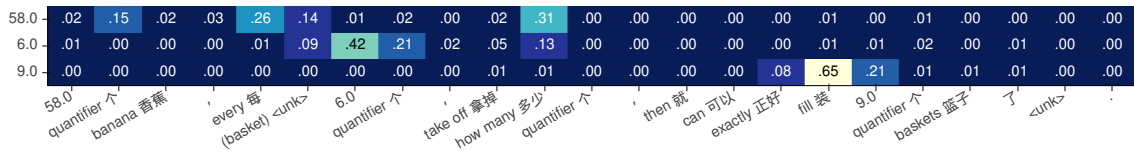


Figure 4.3: The self-attention map visualization of operands’ semantic expressions for the problem “There are 58 bananas. Each basket can contain 6 bananas. How many bananas are needed to be token off such that exactly 9 baskets are filled?”.

**Word-Based - Semantic Representation** To explicitly evaluate the effectiveness of operands’ semantic representations, we rewrite semantic representation of the  $i$ -th operand in the problem texts from (4.2) to  $e_i^c = b_i^c$ , where  $b_i^c$  is a parameter. Thus for every problem, the representation of the  $i$ -th operand is identical, even though their meanings in different problems may be different. This modification assumes that no semantic information is captured by  $b_i^c$ , which can merely represent a symbolic placeholder in an equation. Because the semantic transformer is to transform the semantic representations, applying this component is meaningless. Here the semantic transformer is also replaced with  $f_{\diamond}(e_1, e_2) = e_{\diamond}$  as the setting of the previous ablation test. The results show that the model without using semantic representations of operands causes a significant accuracy drop of 3.5%. The main contribution of this paper about modeling semantic meanings of symbols is validated and well demonstrated here.

## 4.7 Qualitative Analysis

To further analyze whether the proposed model can provide interpretation and reasoning, we visualize the learned semantic representations of constants to check where the important cues are,

### 4.7.1 Constant Embedding Analysis

To better understand the information encoded in the semantic representations of constants in the problem, a self-attention is performed when their semantic representations are ex-

tracted by the encoder. Namely, we rewrite (4.2) as

$$e_i^c = \text{Attention}(h_{p_i}^E, \{h_t^E\}_{t=1}^m). \quad (4.17)$$



Then we check the trained self-attention map ( $\alpha$  in the attention function) on the validation dataset.

For some problems, the self-attention that generates semantic representations of constants in the problem concentrates on the number’s quantifier or unit, and sometimes it also focuses on informative verbs, such as “*gain*”, “*get*”, “*fill*”, etc., in the sentence. For example, Figure 4.3 shows the attention weights for an example math word problem, where lighter colors indicate higher weights. The numbers “58” and “6” focus more on the quantifier-related words (e.g. “*every*” and “*how many*”), while “9” pays higher attention to the verb “*fill*”. The results are consistent with those hand-craft features for solving math word problems proposed by the prior research [29, 31, 30]. Hence, we demonstrate that the automatically learned semantic representations indeed capture critical information that facilitates solving math word problems without providing human-crafted knowledge.

### 4.7.2 Decoding Process Visualization

We visualize the attention map ( $q_t$  in (4.6)) to see how the attention helps the decoding process. An example is shown in the top of Figure 4.4, where most attention focuses on the end of the sentence. Unlike the machine translation task, the attention shows the word-level alignment between source and target languages, solving math word problems requires high-level understanding due to the task complexity.

To further analyze the effectiveness of the proposed gating mechanisms for stack action and operand selection, the activation of gates  $g^{sa}$ ,  $g^{opd}$  at each step of the decoding process is shown in the bottom of Figure 4.4. It shows that most of time, the gate activation is high, demonstrating that the proposed gating mechanisms play an important role during decoding. We also observe a common phenomenon that the activation  $g_2^{sa}$ , which controls how much attention the stack action selector puts on the stack state when deciding an stack

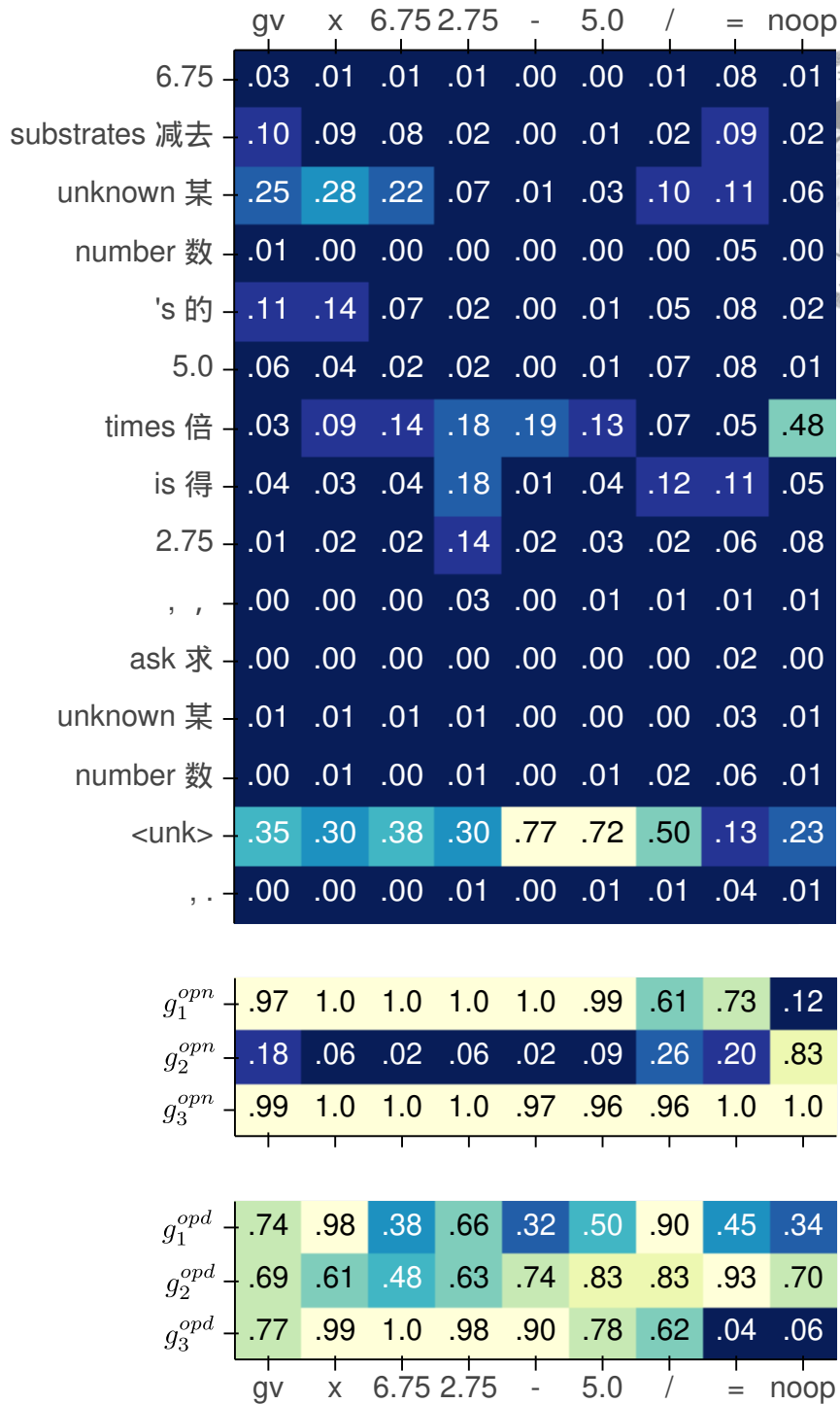


Figure 4.4: Word attention and gate activation ( $g^{sa}$  and  $g^{opd}$ ) visualization when generating stack actions for the problem “6.75 deducting 5 times of an unknown number is 2.75. What is the unknown number?”, where the associated equation is  $x = (6.75 - 2.75) \div 5$ . Note that  $g^{opd}$  is meaningful only when the  $t$ -th stack action is push\_op.

action, is usually low until the last “operator application” stack action. For example, in the example of Figure 4.4,  $g_2^{sa}$  is less than 0.20 till the last argument selection stack action, and activates when deciding the *division operator application* ( $\div$ ) and the *equal application*

---

**Problem & Results**

---

红花有 60 朵，黄花比红花多  $\frac{1}{6}$  朵，黄花有多少朵。(There are 60 red flowers. Yellow flowers are more than red ones by  $\frac{1}{6}$ . How many yellow flowers are there?)

Generated Equation:  $60 + \frac{1}{6}$

Correct Answer: 70

火车 48 小时行驶 5920 千米，汽车 25 小时行驶 2250 千米，汽车平均每小时比火车每小时慢多少千米？(The train travels 5920 kilometers in 48 hours, and the car travels 2250 kilometers in 25 hours. How many kilometers per hour is the car slower than the train?)

Generated Equation:  $2250 \div 25 - 5920 \div 48$

Correct Answer:  $33\frac{1}{3}$

小红前面 5 人，后面 7 人，一共有多少人？(There are 5 people in front of Little Red and 7 people behind. How many persons are there in total?)

Generated Equation:  $5 + 7$

Correct Answer: 13

---

Table 4.3: Randomly sampled incorrect predictions.

(=). It may result from the higher-level semantics of the operand ( $6.75 - 2.75$ ) on the stack when selecting the stack action *division operator application* ( $\div$ ). In terms of the activation of  $g^{opd}$ , we find that three features are important in most cases, demonstrating the effectiveness of the proposed mechanisms.

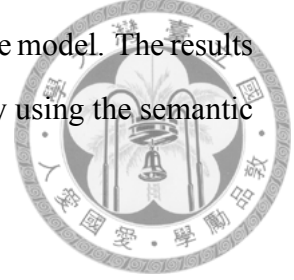
### 4.7.3 Error Analysis

We randomly sample some results predicted incorrectly by our model shown in Table 4.3. In the first example, the error is due to the language ambiguity, and such ambiguity cannot be resolved without considering the exact value of the number. From the second example, although our model identifies the problem as a comparison problem successfully, it handles the order of the operands incorrectly. For the third problem, it cannot be solved by using only the surface meaning but requires some common sense. Therefore, above phenomena show the difficulty of solving math word problems and the large room for improvement.

### 4.7.4 Discussion

In this part, an end-to-end neural math solver that incorporates semantic representations of numbers is proposed in this work. The experiments show that the proposed model

outperforms the strong baseline model by almost 10% accuracy on the benchmark dataset, and empirically demonstrate the effectiveness of each component in the model. The results proves that, the reasoning capability of machines can be improved by using the semantic of the symbols.








---

**Algorithm 1** Training and Inference
 

---

```

function SolveProblem(problem_text)
   $v \leftarrow \text{ExtractConstants}(\text{problem\_text})$  ▷  $v$  is a list of constants in the problem.
   $h^E, h_0^D, c_0^D, E \leftarrow \text{Encoder}(\text{problem\_text})$ 
   $S \leftarrow \text{Stack}()$ 
   $\text{ret}, \text{loss}, t, \text{equations} \leftarrow \text{padding}, 0, 1, \{\}$ 
  while not solvable(equations) do
     $h_t^D \leftarrow \text{LSTM}(h_{t-1}^D, c_{t-1}, \text{ret})$ 
     $s_t \leftarrow S.\text{get\_top2}()$ 
     $h^E \leftarrow \text{Attention}(h_{t-1}^D, h^E)$ 
     $r_t \leftarrow [h_t^D, s_t, h^E]$ 
     $p_{sa} \leftarrow \text{StackActionSelector}(r_t)$ 
     $p_{opd} \leftarrow \text{OperandSelector}(r_t)$ 
    if training then ▷ Target equation  $y$  is available when training.
       $Y_t \leftarrow y_t$ 
      if  $y_t$  is operand then
         $\text{loss} \leftarrow \text{loss} + L_1(\text{push}) + L_2(y_t)$ 
      else
         $\text{loss} \leftarrow \text{loss} + L_1(y_t)$ 
      end if
    else
       $Y_t \leftarrow \text{StackActionSelector}(r_t^{sa})$ 
      if  $Y_t = \text{push}$  then
         $Z_t \leftarrow \text{OperandSelector}(r_t^{opd})$ 
      end if
    end if
    if  $Y_t = \text{gen\_var}$  then
       $e^x \leftarrow \text{Attention}(h_t^D, h^E)$ 
       $\text{ret} \leftarrow e^x$ 
    else if  $Y_t = \text{push}$  then
       $S.\text{push}(v_{Z_t}, e_{Z_t})$ 
       $\text{ret} \leftarrow e_{Z_t}$ 
    else if  $Y_t \in \{+, -, \times, \div\}$  then
       $(v_a, e_a), (v_b, e_b) = S.\text{pop}(), S.\text{pop}()$ 
       $S.\text{push}(v_a Y_t v_b, f_{Y_t}(e_a, e_b))$ 
       $\text{ret} \leftarrow f_{Y_t}(e_a, e_b)$ 
    else if  $Y_t = \text{equal}$  then
       $(v_a, e_a), (v_b, e_b) = S.\text{pop}(), S.\text{pop}()$ 
       $\text{equations} = \text{equations} \cup "v_a = v_b"$ 
       $\text{ret} \leftarrow S.\text{top}()$ 
    end if
  end while
  return solve(equations)
end function

```

---





## Chapter 5

# Conversation Modeling

To investigate the conversation understanding capability of machines, the conversational question answering task is focused here. Two main datasets for conversational question answering, QuAC [5] and CoQA [6] are focused in this work. In the two datasets, questions and answers are collected in a conversational manner, where each conversation includes two participants: a student who asks question about a given passage, and a teacher who answers the question according to the passage. The teacher in both sets may reply “*no answer*” if the answer cannot be found in the given passage. When evaluating the model for both tasks, the content as well as the position of answers to the previous question is available to the model. In the statistics of two datasets, answers to the consecutive questions tend to be close to each other depicted in Figure 5.1.

Even though both datasets are collected for conversational question answering, they have several different properties.

- Answer format

Answers in QuAC are always the text spans in the given passage, while answers in CoQA are free texts similar to some spans in the passage. Answers in QuAC is generally longer than answers in CoQA shown in Figure 5.2, where the distribution implies that QuAC is more realistic than CoQA. Answering “*yes*” or “*no*” is also allowed in CoQA. Note that the evidence span (span in the passage that supports the answer) is provided in CoQA, so the previous answers’ position information is

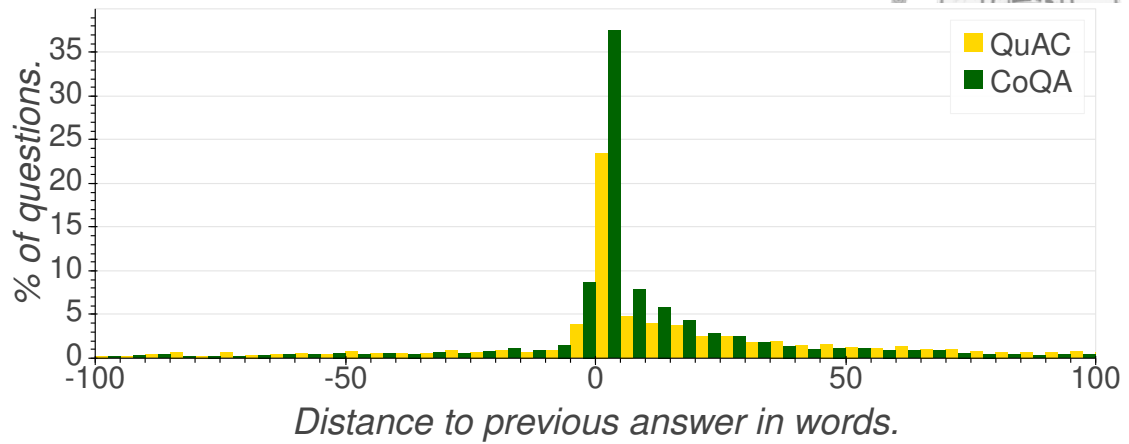
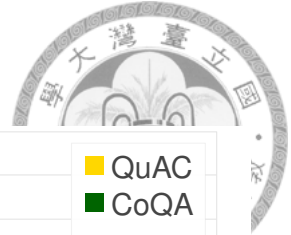


Figure 5.1: The histogram of distance between answers to consecutive questions of a conversation in words. The bin size is 5 words. The distance is counted as zero if the two answer spans are overlapped, and is positive if the current is after the previous answer, negative otherwise.

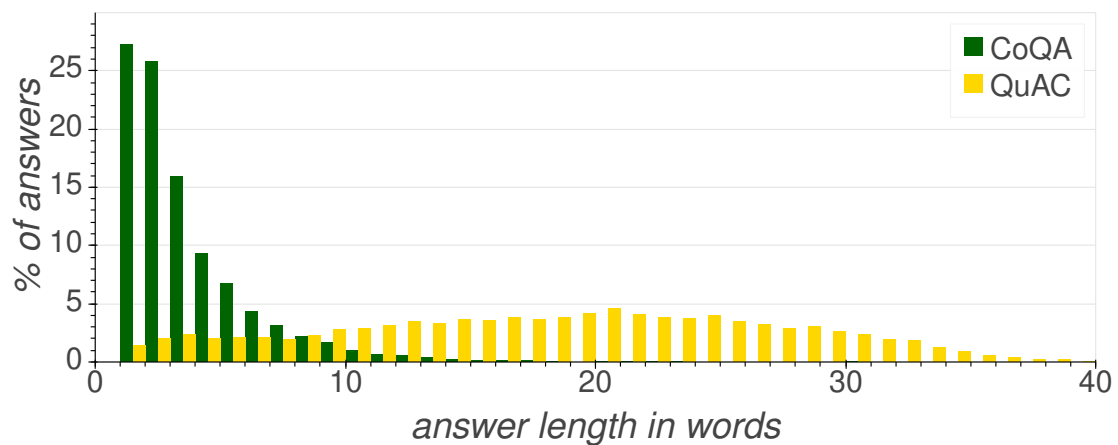


Figure 5.2: Histogram of answer length distribution in QuAC and CoQA. The bin size is 1 word.

still available.

- Dataset collection process

The Amazon mechanical turkers who generated the CoQA dataset have full access<sup>\*</sup> to the passage. On the other hand, turkers who generated questions in QuAC cannot see the passages. The latter setting may be more suitable for practical applications, because real users want to seek for information using questions when not reading passages.



## 5.1 Models

We consider models including FlowQA, BERT, and SDNet, as they are the only publicly available models till now. For FlowQA and SDNet, we use the code released by the authors. Some modifications are made for the following experiments<sup>1</sup>. Each models of each setting are trained with 3 different random seeds, and the resulted mean and standard deviation value are reported for reliability.

All models in the experiments are mainly based on those designed for single-turn reading comprehension tasks, so this section focuses on describing the modification for each method in order to handle understanding in conversational question answering. Below three models are detailed.

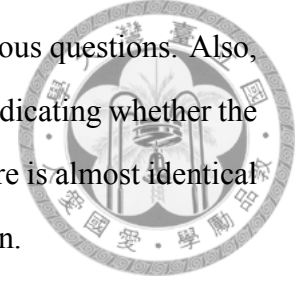
### 5.1.1 FlowQA

FlowQA [7] is the model specifically designed for conversational question answering, which contains a mechanism that can incorporate intermediate representations generated during the process of answering previous questions. This model significantly improved conversational machine comprehension tasks for both QuAC and CoQA data. In FlowQA, the main mechanism designed for the conversational structure is the Integration-Flow layer. In the model, there is a question-aware context representation  $C_i$  for each question  $Q_i$  in the history, and the Integration process simply applies BiLSTM to each  $C_i$

---

<sup>1</sup>For reproducibility, we will release all code, script, and experiment settings.

independently. After that, the Flow process applies BiLSTM to the same word across different context representations in order to capture knowledge of previous questions. Also, an *one-bit* feature is added to the input context word representation indicating whether the word appears in previous answers. The rest of the reasoning procedure is almost identical to FusionNet [55] that focuses on single-turn machine comprehension.



### 5.1.2 BERT

In the current state-of-the-art question answering models, most models leverage the benefits from BERT [56] to advance the task. We apply BERT on QuAC by converting the task into a single-turn machine reading comprehension task such as SQuAD [2]. We prepend previous  $N$  questions to the current question  $Q_k$ , so it becomes  $\hat{Q}_k = \{Q_{k-N}, \dots, Q_{k-1}, Q_k\}$ . At the embedding layer, beside the original word embedding, segment embedding and position embedding, an additional embedding is also added to represent whether the word appears in previous answer spans. Then we follow the procedure of applying BERT to SQuAD that concatenates the extended and the context to form the input and uses the context output representations from BERT to predict the start and the end of the answer span [56].

### 5.1.3 SDNet

To incorporate the information from dialogue history, SDNet prepends not only previous questions but also previous answers to the current question  $Q_k$ , i.e.,

$$\hat{Q}_k = Q_{k-N}, A_{k-N}, \dots, Q_{k-1}, A_{k-1}, Q_k$$

, and no more additional effort is put to deal with the conversational structure. For the input representation for both context and question words, they used BERT as contextualized embeddings along with GloVe. The rest of the model architecture for reasoning is highly inspired by FusionNet [55].

## 5.2 How Well the Performance Reflects Content Comprehension?



This section attempts at investigating how well the performance reflects the capability of comprehension on QuAC and CoQA? Both datasets claim rich linguistic phenomena unique to conversations, where QuAC claims to have 61% of questions including coreference referring to entities in the given passage, 44% of coreference referring to entities in previous history, and 11% of questions that ask for more information in the conversation. Also, CoQA claims to have 49.7% of questions with explicit coreference to conversations, and 19.8% with implicit ones. Given such high ratio of questions related to conversations, it is natural to expect that *higher performance implies better understanding in conversational question answering*. Especially, the understanding should be based on the content of the conversation, in which those special linguistic phenomena is embodied. To inspect the expectation, we design experiments based on the premise: If comprehension on the content of conversation is reflected well by the performance, then model trained without the access to conversation content should not achieve high performance.

### 5.2.1 Experimental Settings

We compare models trained and tested with three different settings:

- Original: The model has free access to the previous conversation history, as the setting proposed by the models.
- - text: The model has no access to the content of the answer to the previous question, but has access to their position in the provided context. The previous questions are not used either.
- - conversation: The model has *no any* access to the previous conversation history.

In the - text setting, the answer span in the passage is masked with zeros. As questions are to seek for information, the answer content should be highly informative. Therefore,

Dataset	Model	F1 (stdev)	$\Delta$ compared to full model
QuAC	FlowQA	64.4 (.30)	0.0
	FlowQA - text	62.4 (.20)	-2.0
	FlowQA - conversation	54.1 (.13)	-10.3
	BERT	63.6 (.16)	0.0
	BERT - text	62.2 (.48)	-1.4
	BERT - conversation	55.3 (.03)	-8.3
CoQA	FlowQA	76.9 (.22)	0.0
	FlowQA - text	71.5 (.24)	-5.4
	FlowQA - conversation	63.4 (.11)	-13.5
	SDNet + position	76.4 (.31)	0.0
	SDNet + position - text	74.0 (.19)	-2.4
	SDNet - conversation	68.3 (.45)	-8.1

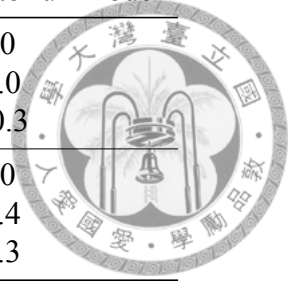


Table 5.1: Model performance on the validation set of QuAC and CoQA. Note that the original SDNet model does not utilize the position information, so *SDNet + position - text* is a modified SDNet with additional one dimension feature indicating the previous answer as the input.

the information loss by answer masking cannot be easily compensated by the surrounding words. Especially, for QuAC, since answer spans are typically as long as sentences, masked language models like BERT can *in no way* recover the masked part. Thus, in this setting, the only information about previous answers is their positions in the passage.

### 5.2.2 Discussion

We compare the results of models in Table 5.1. For both QuAC and CoQA, models trained without access to conversation content can achieve performance significantly better than models trained without access to any conversation history. Especially for QuAC, - content models consistently outperform - conversation models by up to absolute 7% F1 score. As for CoQA, though not as consistently, but similar comparison can also be observed.

The above results indicate that better understanding in the dataset is not well reflected by the performance on these two datasets. Undoubtedly, better comprehension should be based on semantic understanding specific to the textual content. However, even no content is provided to - text models, - text models can still achieve performance higher than - conversation models. It indicates that *higher performance* does not necessarily imply



*better content understanding*. Therefore, future models may need to further verify whether they indeed focus on semantic understanding instead of utilizing the position information only.



### 5.3 Do Models Understand Conversation Content?

The results in §5.2 do not answer the question that if the full models understand content of conversations well. As shown in §5.2, the full models can achieve better performance than models use only the answers' position information. However, §5.2 also shows the usefulness of the previous position information. Since the full model has access to both the position and content of the previous answers, it is not clear whether the better performance is contributed by understanding conversation content. To answer this question, we analyze the trained models by a series of testing settings.

#### 5.3.1 Repeat Attack

We propose *repeat attack* that increases the distance between answers in the context. To do so, a text span is repeated between the answer spans in the passage. For QuAC, as most answers are sentences, we repeat the answer sentence for each answer spans in the passage. For CoQA, since its answer span is generally much shorter than QuAC, we repeat sentences that contain the answer span. The attack examples for both datasets are shown in Table 5.3 and Table 5.4. By repeating part of the passage, the meaning delivered should remain the same. When evaluating the models, the previous answer positions provided to models contain only the text as in original answer. Due to the repeated text, the distance between consecutive answer spans is lengthen by this attack. The distribution of the distance is visualized in Figure 5.5, which is much smoother than the one before the attack (Figure 5.1).

We use *repeat attack* to investigate models' understanding of conversation content. It is motivated by the high ratio of answers close to answers to the previous questions (as shown in Figure 5.1). It is possible that positions of previous answers leak the position

---

### Repeat Attack Example in QuAC

---

**Passage:** In 2004, Oldman returned to prominence when he landed a significant role in the Harry Potter film series, playing Harry Potter's godfather Sirius Black.<sup>ans1</sup> *In 2004, Old man returned to prominence when he landed a significant role in the Harry Potter film series, playing Harry Potter's godfather Sirius Black.* The following year, he starred as James Gordon in Christopher Nolan's commercially and critically successful Batman Begins.<sup>ans2</sup> starred as James Gordon in Christopher Nolan's commercially and critically successful Batman Begins, ...

**Question 1:** What was his resurgence or comeback role?

**Answer 1:** In 2004, Oldman returned to prominence when he landed a significant role in the Harry Potter film series, playing Harry Potter's godfather Sirius Black.

**Question 2:** Are there any other interesting aspects about this article?

**Answer 2:** The following year, he starred as James Gordon in Christopher Nolan's commercially and critically successful Batman Begins,

**Origin FlowQA prediction:** (F1 0.83) The following year, he starred as James Gordon in Christopher Nolan's commercially and critically successful Batman Begins,

**Origin BERT prediction:** (F1 0.83) The following year, he starred as James Gordon in Christopher Nolan's commercially and critically successful Batman Begins

**Attacked FlowQA prediction:** (F1 0.20) In 2004, Oldman returned to prominence when he landed a significant role in the Harry Potter film series, playing Harry Potter's godfather Sirius Black.

**Attacked BERT prediction:** (F1 0.21) In 2004, Oldman returned to prominence when he landed a significant role in the Harry Potter film series, playing Harry Potter's godfather Sirius

Figure 5.3: An example of repeat attack on QuAC and the model results. The red italic text is the inserted attack.

---

### Repeat Attack Example in CoQA

---

**Passage:** Once upon a time, in a barn<sup>ans2</sup> near a farm house, there lived a little white<sup>ans1</sup> kitten named Cotton. *Once upon a time, in a barn near a farm house, there lived a little white kitten named Cotton.* Cotton lived high up in a nice warm place above the barn where all of the farmer's horses slept. But Cotton wasn't alone<sup>ans3</sup> in her little home above the barn, oh no. ...

**Question 1:** What color was Cotton?

**Answer 1:** white

**Question 2:** Where did she live?

**Answer 2:** in a barn

**Question 3:** Did she live alone?

**Answer 3:** Cotton wasn't alone

Figure 5.4: An example of repeat attack on CoQA. The red italic text is the inserted attack. In this example, the attack increases the distance between answer 2 and 3.

information of the current answer. The model may thus learn to take as the answer candidates the sentences close to the previous answer span. On the other hand, if a model does

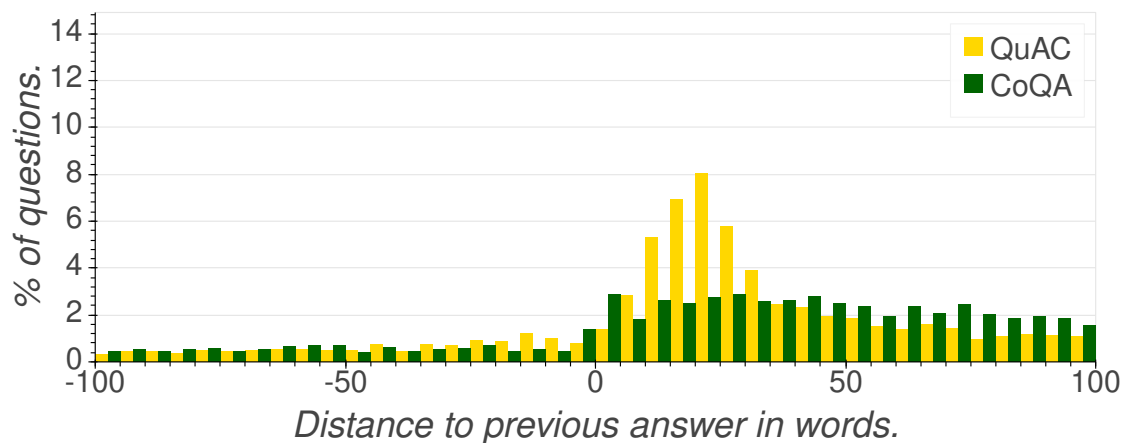


Figure 5.5: Histogram of distance between answers to consecutive questions in words after the attack. The bin size is 5.

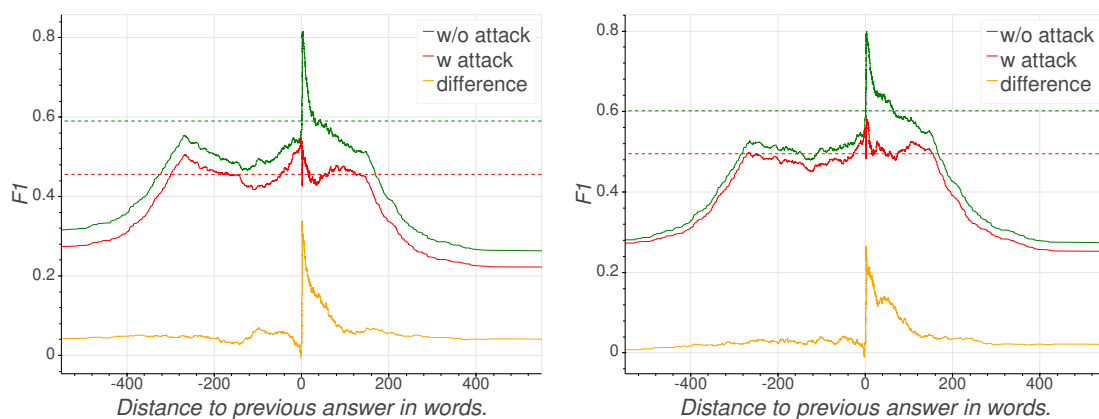


Figure 5.6: F1 change between before and after repeat attack in terms of the answer’s distance to its previous answers on QuAC (left: FlowQA; right: BERT). The x-coordinate is the answer’s distance to the previous answer in words, and the meaning of zero, positive and negative is same as in Figure 5.1. The y-coordinate is the average F1 score. “no answer” sample are ignored here. The dotted line is the average F1 score of all samples.

answer the question by understanding answer content, then the model should be robust to this attack. Therefore, by using the attacked data to test models that are trained on the normal data, the models’ capability of understanding can be well investigated.

For QuAC, the results shown in Table 5.2 indicates that both FlowQA and BERT are sensitive to the distance between consecutive answers. Their performance drops significantly when applying the repeat attack. The F1 score under attack is roughly the F1 score of models trained without any conversational information. Furthermore, we conduct the same experiments on FlowQA and BERT trained without using position information of the previous answers, and find that although they perform much worse than the full models,

they are more robust against the attack.

To better investigate how the answer position is related to the model robustness, we plot the relation between F1 scores before/after attack in terms of the distance to previous answer in Figure 5.6. We find that both BERT and FlowQA predict answers more accurately when the current answer has short distance to the previous answer (blue and green lines). Also, our attack is more effective when the distance is short (yellow line). These findings imply that models trained on QuAC with the position information may rely less on the answer content but rely too much on the answer position. In addition, models trained without the position information may rely more on the semantic information in the conversations.

We also investigate the performance affected by the attack based on if the question is a followup of the previous question, which is annotated in QuAC dataset. The followup questions are questions more likely to depend on the previous question, and therefore are expected to require understanding of conversation content. However, results shown in table 5.3 indicate that they are more vulnerable to our attack.

QuAC models' reliance on position information can be further shown by the qualitative error analysis on the attacked validation set. We inspect the questions that the models originally predict the answer with a high F1 score, but predict the answer with a low F1 scores when applying attack. One sample is shown in Table 5.3, where two models under attack directly predict the sentence after the previous answer span regardless of the content. On other cases, we find FlowQA under attack makes mistakes by selecting the next span more consistently. In contrast, BERT under attack is prone to predict a much implausible answer or reply "*no answer*" (an example is shown in figure 5.7.). The reason may be that BERT is specialized to search answers in the region near the the previous answer, so consequently, when the next sentence after the previous answer is incorrect, it will fail to answer the question correctly.

On the other hand, it is less clear if the drop of performances on CoQA dataset is due to insufficient of conversation content understanding. Unlike on QuAC, - position models are not more robust than the original models as obviously as on QuAC. It can be

---

## QuAC

**Passage:** ... Gardner has been quoted as saying that **he regarded parapsychology and other research into the paranormal as tantamount to "tempting God" and seeking "signs and wonders"**.<sup>ans4</sup> **he regarded parapsychology and other research into the paranormal as tantamount to "tempting God" and seeking "signs and wonders"**. He stated that while he would expect tests on the efficacy of prayers to be **negative, he would not rule out a priori the possibility that as yet unknown paranormal forces may allow prayers to influence the physical world**.<sup>ans5</sup> negative, he would not rule out a priori the possibility that as yet unknown paranormal forces may allow prayers to influence the physical world. ...

...

**Question 4:** What conclusion can he draw from there believes?

**Answer 4:** **he regarded parapsychology and other research into the paranormal as tantamount to "tempting God" and seeking "signs and wonders"**.

**Question 5:** Did he make other statement related to this?

**Answer 5:** **negative, he would not rule out a priori the possibility that as yet unknown paranormal forces may allow prayers to influence the physical world.**

**Origin FlowQA prediction:** (F1 0.94) He stated that while he would expect tests on the efficacy of prayers to be negative, he would not rule out a priori the possibility

**Origin BERT prediction:** (F1 1.0) He stated that while he would expect tests on the efficacy of prayers to be negative,

**Attacked FlowQA prediction:** (F1 0.21) he regarded parapsychology and other research into the paranormal as tantamount to "tempting God" and seeking "signs and wonders".

**Attacked BERT prediction:** (F1 0.0) *no answer*

---

Figure 5.7: An example of repeat attack on QuAC and the model prediction which BERT predicts *no answer*.

observed in figure 5.8 that questions in CoQA seem to be equally susceptible to the attack regardless of the distance to previous answers. Conversation content understanding of the models trained on CoQA may require further investigation.

### 5.3.2 Predict without Previous Answer Text

To investigate if the content of previous answers is used by the models trained with position information, we measure the performance on the validation set predicted without the content of previous answers. To remove the content information of previous answers, we mask the previous answer spans with zeros. Different to previous - text settings, models here has access to previous questions. Particularly, for FlowQA, the content information

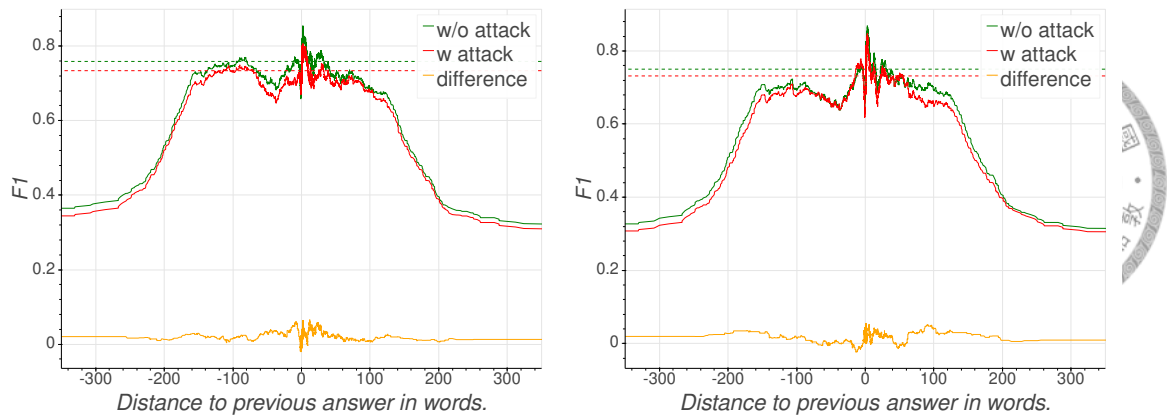


Figure 5.8: Relation between the F1 score before/after repeat attack and the answer’s distance to its previous answer on CoQA. (Left: FlowQA, Right: SDNet)

Dataset	Model	Repeat Attack		$\Delta$
		w/o	w/	
QuAC	FlowQA	64.4 (0.30)	53.3 (0.75)	-11.1
	BERT	63.6 (0.16)	55.3 (0.74)	-8.3
	FlowQA - position	59.3 (0.37)	56.9 (0.37)	-2.4
	BERT - position	58.0 (0.18)	56.5 (0.22)	-1.5
CoQA	FlowQA	76.9 (0.22)	72.0 (0.20)	-4.9
	SDNet	76.4 (0.31)	71.8 (0.48)	-4.6
	FlowQA - position	76.8 (0.42)	72.4 (0.24)	-4.4
	SDNet - position	75.7 (0.76)	70.9 (0.10)	-4.8

Table 5.2: F1 score on the validation of QuAC and CoQA with or without attack. ”- position” indicates training without the position information of answers to previous questions.

of the previous answer may be flowed along with the flow structure, and the RNN memory of the previous answer spans is also reset to zeros.

The more performance drops when masking previous answers implies that the model relies more on the content information of those answers. According to the results in Table 5.4, it seems that all models more or less rely on the text of previous answers. Meanwhile, as expected, the models except SDNet - position trained without the position information almost drop to the performance of ones trained without any conversation history information. SDNet - position is an exception because answers to previous questions are prepended to the question when training and testing, so masking answers in the passage does not remove all the content of previous answers. However, it is surprising that FlowQA on QuAC can still keep the performance up to 60% F1, implying that FlowQA

Model		Followup	No Followup
FlowQA	no attack	64.0 (0.59)	64.8 (0.11)
	attack	49.5 (0.59)	57.6 (0.94)
	$\Delta$	14.6 (1.17)	7.2 (1.00)
BERT	no attack	63.4 (0.32)	64.0 (0.57)
	attack	52.4 (1.37)	58.5 (0.18)
	$\Delta$	11.0 (1.12)	5.5 (0.41)

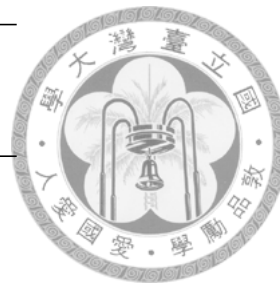


Table 5.3: Impact on F1 score by the attack.

Dataset	Model	Ans. Mask		$\Delta$
		w/o	w/	
QuAC	FlowQA	64.4 (0.30)	60.5 (0.54)	-3.9
	BERT	63.6 (0.16)	52.6 (1.76)	-11.0
	FlowQA - position	59.3 (0.37)	55.0 (1.80)	-4.3
	BERT - position	58.0 (0.18)	50.1 (0.45)	-7.9
CoQA	FlowQA	76.9 (0.22)	71.2 (0.41)	-5.7
	SDNet	76.4 (0.31)	73.5 (0.51)	-2.9
	FlowQA - position	76.8 (0.42)	68.2 (0.28)	-8.6
	SDNet - position	75.7 (0.76)	75.6 (0.69)	-0.1

Table 5.4: F1 results on the validation sets of QuAC and CoQA. Models are inferred without/with applying masks on the previous answers. Models without suffix are trained with full access to conversation, while “- position” indicates that the models are trained without the previous answer position information. Note that in both training settings, masks are not applied.

may rely on position information much more than the semantic information.

### 5.3.3 Predict without Previous Answer Position

To directly test to what extent the models rely on position information of previous answers, we conduct the experiments of predicting answers without position information. The results are shown in Table 5.5, where both results of FlowQA and BERT on QuAC drop significantly if not using position information. Among them, FlowQA drops even more, regardless of the flow structure that models the dialog flow. The performance is even lower than the models trained without using any conversation history. It indicates that although the models on QuAC may rely on the semantics of previous answers, the position information is indeed exploited. On the other hand, it is interesting to see that FlowQA trained on CoQA rely little position information of the previous answers.

Dataset	Model	Position Info.	
		w/	w/o
QuAC	FlowQA	64.4 (0.30)	48.1 (0.72)
	BERT	63.6 (0.16)	54.9 (0.08)
CoQA	FlowQA	76.9 (0.22)	76.7 (0.36)
	SDNet	76.4 (0.31)	73.7 (0.18)



Table 5.5: F1 score on the validation sets of QuAC and CoQA. Models are inferred with/without access to position information, but trained with access to position information.

### 5.3.4 Implication of Above Experiments

Our results show that the models trained on QuAC have high tendency to rely heavily on the position of previous answers. The proposed attack and experiment settings can serve as a diagnosis tool in the future.

## 5.4 Dataset and Model Analysis

To further investigate the performance difference between the models trained on QuAC and CoQA, two questions are focused here.

**Why do CoQA models rely less on position information?** It is unclear why the models trained on CoQA rely less on the position information of the previous answer as shown in the previous sections. Figure 5.2 shows that answers in CoQA are much shorter than in QuAC, so if we normalize the distance to the previous answer related to the length of the answers, the average distance to the previous answer in CoQA would be much longer than QuAC. Furthermore, short answers may also imply that an answer can be identified as the sentence containing the answer. To verify the suggestion, we randomly shuffle the sentences in the passage for each CoQA example. An example of shuffled passage can be found in table 5.7. By doing this, the cross-sentence information and the order information should be removed from the passage. Then we use CoQA models trained without position information to predict the answers. The results in Table 5.6 show that, roughly speaking, up to 70% questions can still be answer correctly. It thus supports our suggestion, and



Dataset	Model	Shuffle		$\Delta$
		w/o	w/	
CoQA	FlowQA - position	76.8 (0.42)	71.7 (0.25)	-5.1
	SDNet - position	75.7 (0.76)	70.1 (0.59)	-5.6

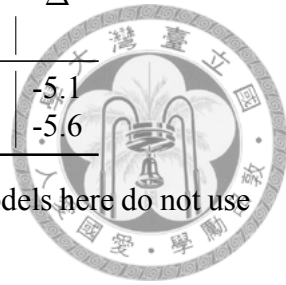


Table 5.6: F1 score of models on shuffled validation set of CoQA. Models here do not use the position information of the previous answers.

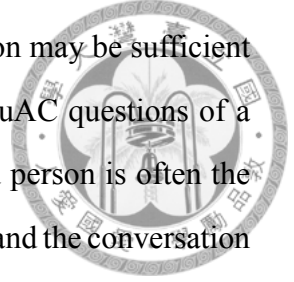
partly explains why models trained on CoQA rely less on the position information.

**Passage:** What are you doing, Cotton?!” The rest of her sisters were all orange with beautiful white tiger stripes like Cotton’s mommy. Then Cotton thought, ”I change my mind. All of her sisters were cute and fluffy, like Cotton. But Cotton wasn’t alone in her little home above the barn, oh no. Cotton lived high up in a nice warm place above the barn where all of the farmer’s horses slept. ”I only wanted to be more like you”. When her mommy and sisters found her they started laughing. ”Don’t ever do that again, Cotton!” So one day, when Cotton found a can of the old farmer’s orange paint, she used it to paint herself like them. And with that, Cotton’s mommy picked her up and dropped her into a big bucket of water. She shared her hay bed with her mommy and 5 other sisters. Her sisters licked her face until Cotton’s fur was all all dry. Next time you might mess up that pretty white fur of yours and we wouldn’t want that!” We would never want you to be any other way”. Being different made Cotton quite sad. She often wished she looked like the rest of her family. When Cotton came out she was herself again. But she was the only white one in the bunch. Cotton’s mommy rubbed her face on Cotton’s and said ”Oh Cotton, but your fur is so pretty and special, like you. they all cried. ” I like being special”. Once upon a time, in a barn near a farm house, there lived a little white kitten named Cotton.

Table 5.7: An example of random shuffled CoQA passage.

**Why do QuAC models rely more position information?** We provide a few possible explanation why the models trained on QuAC rely much on the position information. According to the analysis on QuAC [5], 11% of the questions is of the type “*Anything else?*”. It is common in the general article where the information of the same type is written in near contexts. Because the question “*Anything else?*” is asked to seek for more information similar to the previous answer, it is very likely that the position of the previous answer provides a strong hint for the current answer. Though there is a high percentage of the questions containing pronouns in QuAC, they do not necessarily force the model to learn coreference resolution either. For the pronouns in questions, they often refer to

entities in the previous answer. Because entities in consecutive sentences seldom change much, simply looking for the answer near the previous answer location may be sufficient to answer the question. Especially, we find personal pronouns in QuAC questions of a conversation often refer to only one same person. Also, the referred person is often the main role in the passage. This further removes the necessity to understand the conversation history.



### **5.4.1 Discussion**

In this part, conversation content understanding of different models learned from different datasets is investigated. The experiments shows concerns 1) Performance on QuAC and CoQA does not well reflect model's comprehension on conversation content. 2) The model trained on QuAC does not necessarily learn conversation comprehension. 3) In CoQA, cross-sentence information is not that important for current model. Given the concerns pointed out in this work, future researchers should be able to avoid some possible hazards in this direction.



## Chapter 6

# Conclusion and Future Work

In this work we investigate current machines' capability of understanding natural language in two aspects. Firstly, the reasoning capability is studied. The results show the success of the model inspired by the human reasoning process. Secondly, the capability of conversation content understanding is investigated. A series of experiments are designed to scrutinize the content understanding capability in the previous proposed models, identifying some potential hazards which future researchers should be aware of.


There are potential directions to further investigate the arithmetic reasoning and conversation modeling capability of machines. For the arithmetic reasoning part, some more efforts may be need to better scrutinize the generalization ability of the models. To do so, some composite questions that require capability more than pattern matching may be created. For the conversation modeling part, inventing a more realistic data collection process may be an important future direction. Both the QuAC and CoQA datasets are created in an unrealistic setting and thus may contain artifacts. It is also unclear how to design a model that learns conversation comprehension naturally until now. The future work can focus more on those directions.



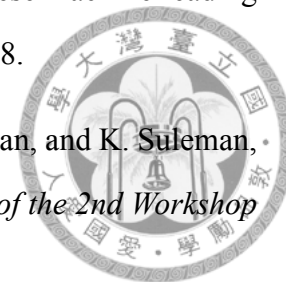


# Bibliography

- [1] S. Mandal and S. K. Naskar, “Solving arithmetic mathematical word problems: A review and recent advancements,” in *Information Technology and Applied Mathematics*, pp. 95–114, Springer, 2019.
- [2] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, “Squad: 100,000+ questions for machine comprehension of text,” in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 2383–2392, 2016.
- [3] T. Nguyen, M. Rosenberg, X. Song, J. Gao, S. Tiwary, R. Majumder, and L. Deng, “Ms marco: A human generated machine reading comprehension dataset,” *arXiv preprint arXiv:1611.09268*, 2016.
- [4] P. Rajpurkar, R. Jia, and P. Liang, “Know what you don’ t know: Unanswerable questions for squad,” in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 784–789, 2018.
- [5] E. Choi, H. He, M. Iyyer, M. Yatskar, W.-t. Yih, Y. Choi, P. Liang, and L. Zettlemoyer, “Quac: Question answering in context,” in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2174–2184, 2018.
- [6] S. Reddy, D. Chen, and C. D. Manning, “Coqa: A conversational question answering challenge,” *Transactions of the Association for Computational Linguistics*, vol. 7, pp. 249–266, 2019.

- 
- [7] H.-Y. Huang, E. Choi, and W. tau Yih, “FlowQA: Grasping flow in history for conversational machine comprehension,” in *International Conference on Learning Representations*, 2019.
- [8] C. Zhu, M. Zeng, and X. Huang, “Sdnet: Contextualized attention-based deep network for conversational question answering,” *arXiv preprint arXiv:1812.03593*, 2018.
- [9] Y.-T. Yeh and Y.-N. Chen, “Flowdelta: Modeling flow information gain in reasoning for conversational machine comprehension,” in *Proceedings of the 2nd Workshop on Machine Reading for Question Answering*, pp. 86–90, 2019.
- [10] M. Seo, A. Kembhavi, A. Farhadi, and H. Hajishirzi, “Bidirectional attention flow for machine comprehension,” in *Proceedings of ICLR*, 2016.
- [11] T.-R. Chiang and Y.-N. Chen, “Semantically-aligned equation generation for solving and reasoning math word problems,” in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 2656–2668, 2019.
- [12] T.-R. Chiang, H.-T. Ye, and Y.-N. Chen, “An empirical study of content understanding in conversational question answering,” *arXiv preprint arXiv:1909.10743*, 2019.
- [13] J. L. Elman, “Finding structure in time,” *Cognitive science*, vol. 14, no. 2, pp. 179–211, 1990.
- [14] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
- [15] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, “Attention is all you need,” in *Advances in Neural Information Processing Systems*, pp. 5998–6008, 2017.
- [16] J. L. Ba, J. R. Kiros, and G. E. Hinton, “Layer normalization,” *arXiv preprint arXiv:1607.06450*, 2016.

- [17] C. C. Shao, T. Liu, Y. Lai, Y. Tseng, and S. Tsai, “Drcd: a chinese machine reading comprehension dataset,” *arXiv preprint arXiv:1806.00920*, 2018.
- [18] A. Trischler, T. Wang, X. Yuan, J. Harris, A. Sordoni, P. Bachman, and K. Suleman, “Newsqa: A machine comprehension dataset,” in *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pp. 191–200, 2017.
- [19] G. Lai, Q. Xie, H. Liu, Y. Yang, and E. Hovy, “Race: Large-scale reading comprehension dataset from examinations,” in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 785–794, 2017.
- [20] D. Dua, Y. Wang, P. Dasigi, G. Stanovsky, S. Singh, and M. Gardner, “DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs,” in *Proc. of NAACL*, 2019.
- [21] A. Amini, S. Gabriel, S. Lin, R. Koncel-Kedziorski, Y. Choi, and H. Hajishirzi, “Mathqa: Towards interpretable math word problem solving with operation-based formalisms,” in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 2357–2367, 2019.
- [22] S. Wang and J. Jiang, “Machine comprehension using match-lstm and answer pointer,” 2016.
- [23] D. Weissenborn, G. Wiese, and L. Seiffe, “Making neural qa as simple as possible but not simpler,” in *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pp. 271–280, 2017.
- [24] Y. Shen, P.-S. Huang, J. Gao, and W. Chen, “Reasonet: Learning to stop reading in machine comprehension,” in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1047–1055, ACM, 2017.
- [25] M. Hu, Y. Peng, Z. Huang, X. Qiu, F. Wei, and M. Zhou, “Reinforced mnemonic reader for machine reading comprehension,” in *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pp. 4099–4106, AAAI Press, 2018.

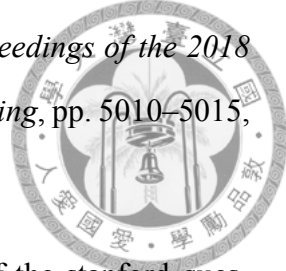


- [26] C. Xiong, V. Zhong, and R. Socher, “DCN+: Mixed objective and deep residual coattention for question answering,” in *International Conference on Learning Representations*, 2018.
- [27] X. Liu, Y. Shen, K. Duh, and J. Gao, “Stochastic answer networks for machine reading comprehension,” in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1694–1704, 2018.
- [28] N. Kushman, L. Zettlemoyer, R. Barzilay, and Y. Artzi, “Learning to automatically solve algebra word problems,” in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014*, pp. 271–281, 2014.
- [29] M. J. Hosseini, H. Hajishirzi, O. Etzioni, and N. Kushman, “Learning to solve arithmetic word problems with verb categorization,” in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pp. 523–533, 2014.
- [30] S. Roy, T. Vieira, and D. Roth, “Reasoning about quantities in natural language,” *TACL*, vol. 3, pp. 1–13, 2015.
- [31] S. Roy and D. Roth, “Solving general arithmetic word problems,” in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pp. 1743–1752, 2015.
- [32] 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.
- [33] S. Roy, S. Upadhyay, and D. Roth, “Equation parsing : Mapping sentences to grounded equations,” in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pp. 1088–1097, 2016.
- [34] S. Upadhyay, M. Chang, K. Chang, and W. Yih, “Learning from explicit and implicit supervision jointly for algebra word problems,” in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 297–306, 2016.



- [35] S. Upadhyay and M. Chang, “Annotating derivations: A new evaluation strategy and dataset for algebra word problems,” in *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, pp. 494–504, 2017.
- [36] S. Roy and D. Roth, “Mapping to declarative knowledge for word problem solving,” *TACL*, vol. 6, pp. 159–172, 2018.
- [37] L. Wang, D. Zhang, L. Gao, J. Song, L. Guo, and H. T. Shen, “MathDQN: Solving arithmetic word problems via deep reinforcement learning,” in *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.
- [38] P. Mehta, P. Mishra, V. Athavale, M. Shrivastava, and D. M. Sharma, “Deep neural network based system for solving arithmetic word problems,” in *Proceedings of the IJCNLP 2017*, pp. 65–68, 2017.
- [39] Y. Wang, X. Liu, and S. Shi, “Deep neural solver for math word problems,” in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 845–854, 2017.
- [40] W. Ling, D. Yogatama, C. Dyer, and P. Blunsom, “Program induction by rationale generation: Learning to solve and explain algebraic word problems,” in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017*, pp. 158–167, 2017.
- [41] S. Shi, Y. Wang, C. Lin, X. Liu, and Y. Rui, “Automatically solving number word problems by semantic parsing and reasoning,” in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015*, pp. 1132–1142, 2015.
- [42] R. Jia and P. Liang, “Adversarial examples for evaluating reading comprehension systems,” in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 2021–2031, 2017.

- [43] D. Kaushik and Z. C. Lipton, “How much reading does reading comprehension require? a critical investigation of popular benchmarks,” in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 5010–5015, 2018.
- [44] M.-A. Rondeau and T. J. Hazen, “Systematic error analysis of the stanford question answering dataset,” in *Proceedings of the Workshop on Machine Reading for Question Answering*, pp. 12–20, 2018.
- [45] M. Yatskar, “A qualitative comparison of coqa, squad 2.0 and quac,” in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 2318–2323, 2019.
- [46] Y. Nie, Y. Wang, and M. Bansal, “Analyzing compositionality-sensitivity of nli models,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 6867–6874, 2019.
- [47] C. Sankar, S. Subramanian, C. Pal, S. Chandar, and Y. Bengio, “Do neural dialog systems use the conversation history effectively? an empirical study,” in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, (Florence, Italy), pp. 32–37, Association for Computational Linguistics, July 2019.
- [48] R. Geirhos, P. Rubisch, C. Michaelis, M. Bethge, F. A. Wichmann, and W. Brendel, “Imagenet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness.,” in *International Conference on Learning Representations*, 2019.
- [49] W. Brendel and M. Bethge, “Approximating CNNs with bag-of-local-features models works surprisingly well on imagenet,” in *International Conference on Learning Representations*, 2019.



- [50] I. Sutskever, O. Vinyals, and Q. V. Le, “Sequence to sequence learning with neural networks,” in *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014*, pp. 3104–3112, 2014.
- [51] T. Luong, H. Pham, and C. D. Manning, “Effective approaches to attention-based neural machine translation,” in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 1412–1421, 2015.
- [52] A. Graves, G. Wayne, and I. Danihelka, “Neural turing machines,” *arXiv preprint arXiv:1410.5401*, 2014.
- [53] A. Meurer, C. P. Smith, M. Paprocki, O. Čertík, S. B. Kirpichev, M. Rocklin, A. Kumar, S. Ivanov, J. K. Moore, S. Singh, T. Rathnayake, S. Vig, B. E. Granger, R. P. Muller, F. Bonazzi, H. Gupta, S. Vats, F. Johansson, F. Pedregosa, M. J. Curry, A. R. Terrel, v. Roučka, A. Saboo, I. Fernando, S. Kulal, R. Cimrman, and A. Scopatz, “SymPy: symbolic computing in python,” *PeerJ Computer Science*, vol. 3, p. e103, Jan. 2017.
- [54] B. Robaidek, R. Koncel-Kedziorski, and H. Hajishirzi, “Data-driven methods for solving algebra word problems,” *CoRR*, vol. abs/1804.10718, 2018.
- [55] H.-Y. Huang, C. Zhu, Y. Shen, and W. Chen, “Fusionnet: Fusing via fully-aware attention with application to machine comprehension,” in *International Conference on Learning Representations*, 2018.
- [56] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding,” in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186, 2019.