Generating Equation by Utilizing Operators : GEO Model

Kyung Seo Ki* **Donggeon Lee*** **Bugeun Kim** **Gahgene Gweon**
Department of Transdisciplinary Studies
Seoul National University
Seoul, Republic of Korea
{kskee88,lsw5835,cd4209,ggweon}@snu.ac.kr

Abstract

Math word problem solving is an emerging research topic in Natural Language Processing. Recently, to address the math word problem solving task, researchers have applied the encoder-decoder architecture, which is mainly used in machine translation tasks. The state-of-the-art neural models use hand-crafted features and are based on generation methods. In this paper, we propose the GEO (Generation of Equations by utilizing Operators) model that does not use hand-crafted features and addresses two issues that are present in existing neural models: 1. missing domain-specific knowledge features and 2. losing encoder-level knowledge. To address missing domain-specific feature issue, we designed two auxiliary tasks: operation group difference prediction and implicit pair prediction. To address losing encoder-level knowledge issue, we added an Operation Feature Feed Forward (OP3F) layer. Experimental results showed that the GEO model outperformed existing state-of-the-art models on two datasets, 85.1% in MAWPS, and 62.5% in DRAW-1K, and reached comparable performance of 82.1% in ALG514 dataset.

1 Introduction

Math word problem solving is an emerging research topic in Natural Language Processing. The problem solving task involves dealing with numbers or numeric context information, which is useful in understanding language. Recently, to address the math word problem solving task, researchers applied neural models with the encoder-decoder architecture, which is mainly used in machine translation tasks. Such models generate equations as outputs using given natural language sentence sequences as inputs (Wang et al., 2019; Meng and Rumshisky, 2019). The state-of-the-art neural models use hand-crafted features and are based on generation methods. In the primary benchmark datasets, such hand-crafted feature-based models have recently reported performance levels of about 80%, while neural models remain at 60-70% (Wang et al., 2017). Due to the relatively low performance of neural models, models that utilize hand-crafted features still retain state-of-the-art (SOTA) performance in many datasets (Upadhyay et al., 2016a). However, defining and building hand-crafted features is costly. In this paper, we propose the GEO (Generation of Equations by utilizing Operators) model that does not use hand-crafted features and addresses two issues that are present in existing neural models: 1. missing domain-specific knowledge features and 2. losing encoder-level knowledge.

The first issue of missing domain-specific knowledge feature occurs when neural models don't make enough use of domain-specific information of the target domain. For our case of math word problem solving, domain-specific knowledge is mathematical information. Considering how humans extract and process mathematical information from a word problem when building an equation for the problem, people identify main mathematical elements, such as numbers and operations (Usiskin, 1988). Existing neural models so far assumes that such main mathematical elements would be automatically identified in the model training process. Yet, explicitly utilizing information about numbers and operations can be used to achieve better model performance. Therefore, in the GEO model, we will introduce two auxiliary tasks that enable the encoder to learn information about numbers and operators explicitly.

* Equal contribution.

Proceedings of the 28th International Conference on Computational Linguistics, pages 426–436
Barcelona, Spain (Online), December 8-13, 2020

The second issue of losing encoder-level knowledge occurs when decoder-learned hidden states are not utilized in the generation process in an encoder-decoder model. In the case of math problems, when information is lost in the generation process, the decoder may generate an irrelevant number that does not appear in the problem. To solve this issue, Huang et al. (2018) introduced a method of copying numbers in the questions. However, this method has a limitation in that it is difficult to apply the method when an answer equation contains a number that does not appear explicitly in the problem. Therefore, to address this problem, we will introduce an additional layer, Operation Feature Feed Forward (OP3F) layer that combines the encoder hidden states and the decoder hidden states in the GEO model. The OP3F layer will allow the encoder-decoder model to reflect the information learned from the encoder during the generation process.

The GEO model is a neural model that is capable of sufficiently learning mathematical information without using hand-crafted features. Our experimental results show that the GEO model outperforms the existing neural models. In addition, the GEO model yields a comparable performance as the hand-crafted feature-based model. The contributions of our research are summarized as follows:

- We propose a novel model that achieves comparable (82.1% in ALG514) and higher (62.5% in DRAW-1K) problem solving accuracy compared to existing hand-crafted feature-based models and neural models.

- The proposed GEO model can overcome missing domain-specific knowledge features and losing encoder-level knowledge of the existing encoder-decoder based neural model in the domain of math word problem solving.

The following sections are constructed as follows: First, in Section 2, we introduce the previous attempts for math word problem solving. Next, in Section 3, we explain the GEO model's architecture and its components. In Section 4 and 5, we report the experimental setup and results along with a discussion of our results. Lastly, in Section 6, we conclude our study and present future research directions.

2 Related Work

Our goal is to improve the performance of the neural model to be similar or better compared to the hand-crafted feature (HCF)-based model by addressing the two problems of the existing neural model, missing domain-specific knowledge, and losing encoder-level knowledge. In this section, we present prior models that were used in solving math word problems and explore directions of improvement for existing models.

In math word problem solving tasks, earlier machine learning studies utilized HCFs that reflect the domain-specific knowledge to extract mathematical information. Kushman's study (Kushman et al., 2014), which was the first attempt in solving the math problems using machine learning techniques, showed that math word problems could be solved by using expert-defined HCFs. Following Kushman's research, studies reported that improved performance can be obtained by defining and utilizing features related to domain-specific knowledge (Upadhyay et al., 2016a; Huang et al., 2018). In particular, the MixedSP (Upadhyay et al., 2016a), which utilized context-related/question-related HCFs, has been reported as SOTA of 83.0% in the benchmark dataset of ALG514.

Recently, with the emergence of neural models, researchers solved the math problem task by using neural models with only automatically extracted features, which is cost efficient compared to expert-designed HCFs (Wang et al., 2017; Robaidek et al., 2018). Yet, such models did not show comparable performance to the HCF-based models. For example, Wang et al. (2017)'s Deep Neural Solver (DNS), which utilizes word embedding and Gated Recurrent Unit (GRU) network for automatic feature extraction, reported performance of 70.1% in the ALG514 dataset, which is lower than the performance of an HCF-based model with a performance of 79.7%. The CASS model (Huang et al., 2018) also showed similar trends in performance between pure neural models and HCF-based models. CASS reported 44.8% performance for ALG514 in the pure neural model, whereas the model reported an 82.5% performance for the same model with HCF-based domain-specific features. A possible explanation for why the neural

models yield lower performance than the HCF-based models can be that the neural models could not sufficiently utilize the domain-specific knowledge feature, which the HCF-based models could usefully utilize. Considering that the performance improvement occurred when HCFs were additionally given to the same model in several studies, it could be said that the model did not lack the learning ability but could not extract the domain-specific knowledge features. Thus, if a technique can be devised in which the neural model can automatically extract such domain-specific knowledge features, there is still a possibility that the neural model will perform better or comparable to the existing HCF-based models.

The fine-tuning technique of the language model, which has been actively applied to neural models in recent years, offers useful implications for finding ways to automatically extract domain-specific knowledge features. The fine-tuning technique is a technique that adjusts the language model, which is a pretrained representation using a large amount of corpus to address a specific task. The Fine-tuning technique has been reported to yield good performance in various natural language processing tasks (Devlin et al., 2018; Liu, 2019; Clark et al., 2020). In fact, some studies have shown that applying the technique contributes to performance improvement in the math word solving task. For example, Ki et al. (2020) reported the highest performance in the Korean language for two datasets of CC and IL using BERT (Devlin et al., 2018), a widely used language model.

However, it is insufficient to conclude that fine-tuning the pretrained language model would be useful for extracting domain-specific knowledge features. This is because there is a study showing that the pretrained language model has decreased the performance (Robaidek et al., 2018), and there has been a report that the pretrained language model did not have much effect in cases with a little amount of data (Andor et al., 2019). Therefore, one should examine whether the impact of fine-tuning a language model can be used to extract the domain-specific knowledge feature in the math word problem solving task. In the GEO model, two auxiliary tasks are designed to learn domain-specific knowledge features in the fine-tuning process of the language model.

Although neural models can properly learn domain-specific knowledge features, the learned features must be utilized without loss in the model's architecture, in order to contribute to performance improvement. However, in the encoder-decoder structure which is recently used to solve math word problems, if the length of the equation is complex, the information learned by the model through the encoder could be lost in the decoder's equation generation process. To avoid this losing encoder-level knowledge issue, a study such as Wang et al. (2019) applied techniques such as Recursive Neural Network or attention method, which are known to respond relatively well to long sequences. However, considering that there are reports of degrading performance for long equation generation in this study as well, it is expected that further exploration to solve this problem is required. So in the GEO model, we try to devise a new layer for encoder-decoder architecture that addresses the above issue.

3 Generating Equation by Utilizing Operators (GEO) Model

We propose the GEO model, which is a variant of the encoder-decoder model. Figure 1 demonstrates the GEO model's architecture. The input for the GEO is the preprocessed math word problem and the output is an answer equation which is in the form of the Index for Numeric Constants (INC) postfix equation, which can be used to generate the solution to the given word problem. The GEO model has two components which does not exist in a pure encoder-decoder model. First, the GEO has auxiliary task layers component, which addresses the missing domain-specific knowledge features. Second, the GEO utilizes an Operation Feature Feed Forward (OP3F) layer component that applies encoder-level knowledge to decoder hidden states. In the following subsections, we will explain the data preprocessing step along with the main components of the GEO, which are the base model, auxiliary task layers, and the OP3F layer.

3.1 Data preprocessing

We processed the original dataset which consists of questions and equations to use as the GEO model's input. Figure 2 shows the example of how the original format of the math word problem dataset is converted to the processed format. The processed format used by the GEO model consists of INCs, an

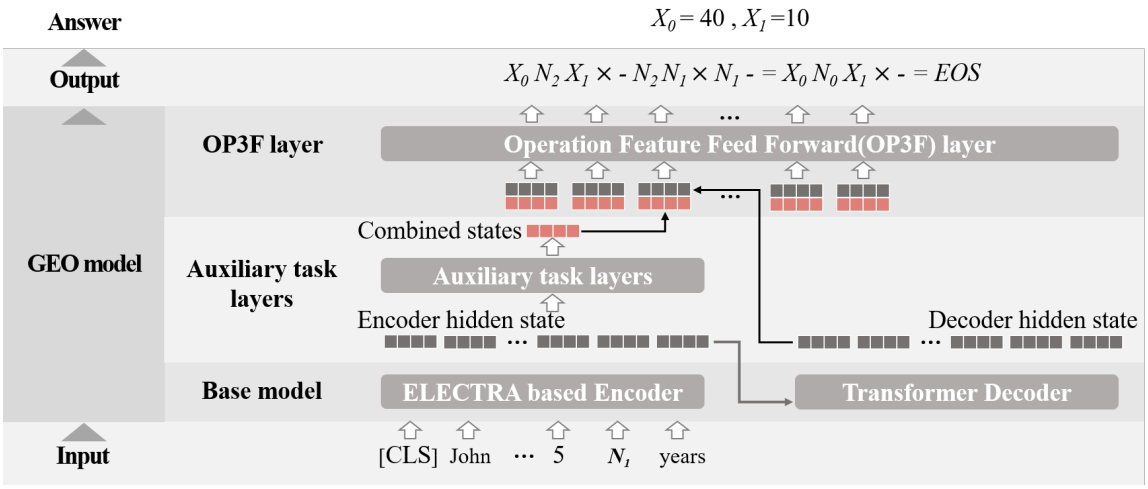

Figure 1: The architecture of GEO model

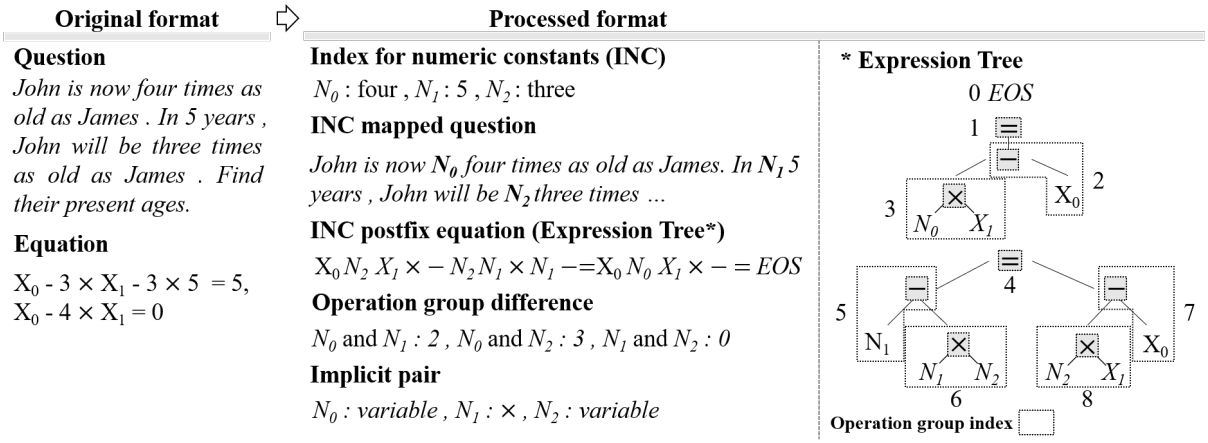

Figure 2: Preprocessing for GEO model's input

INC mapped question, an INC postfix equation, Operation Group (OG) indices, OG differences, and implicit pairs. INC is a normalized form of a number, N_i which indexes the numbers that appear in the question in the order they appear. By introducing the normalized form of numbers, we enabled the model to understand the abstract meaning of numeric tokens, which helps to generate the equation. Then, the INC is added next to the number appearing in the question to form the 'INC mapped question.' The INC postfix equation is made by replacing the number presented in the equation of the original format with the INC in a postfix form.

Next, we added two preprocesses, operation group difference and implicit pair, which will be used for two auxiliary tasks. First, operation group difference is calculating the difference between operation groups (OG), which is a sub-tree in an expression tree of an equation. Using an expression tree, the OG assigning process for the postfix equation begins with assigning a group from the right end of the postfix equation. Thus, OG index 0 is assigned to the end-of-sentence(EOS) token, and subsequent OG indices are given to operators in order that appears in the expression tree. For the operands, OG allocation is performed by designating the OG of the nearest non-terminal node of each operand. After assigning OGs, the difference between OG indices can be calculated.

Second, the implicit pair is designated for each INC token in a given problem. The implicit pair links a variable or operator that is associated with each INC token in an INC postfix equation. To make an implicit pair, first we examine a variable located right before and after the INC token in the INC postfix equation. If variable does not exist, we examine an operator which belongs to the same operation group

with the INC. If an INC exists multiple times in the INC postfix equation, we selected the first left occurrence of the INC token for model learning efficiency. For example, to assign the implicit pair for N_1, we refer to the first left nearest INC N_1, and assign the operator $\times$ as an implicit pair since the $\times$ belongs to the same group.

3.2 Base model

Our base model for comparing GEO model's performance is an encoder-decoder model which uses Electra (Clark et al., 2020) as an encoder and Transformer (Vaswani et al., 2017) as a decoder. The Electra model is a Transformer-based language model, which has a similar structure to BERT, which is known to increase cost efficiency by using a discriminative model for pretraining. We selected Electra considering the advantages of performance since Electra has achieved SOTA performance in various natural language processing tasks with fewer parameters than XLNet (Yang et al., 2019) and RoBERTa (Liu et al., 2019). Meanwhile, We selected the Transformer decoder for our encoder-decoder architecture based on the prior study that reports the highest performance in summarization tasks using the Transformer decoder along with the BERT encoder (Liu, 2019).

3.3 Two auxiliary tasks: operation group difference prediction and implicit pair prediction

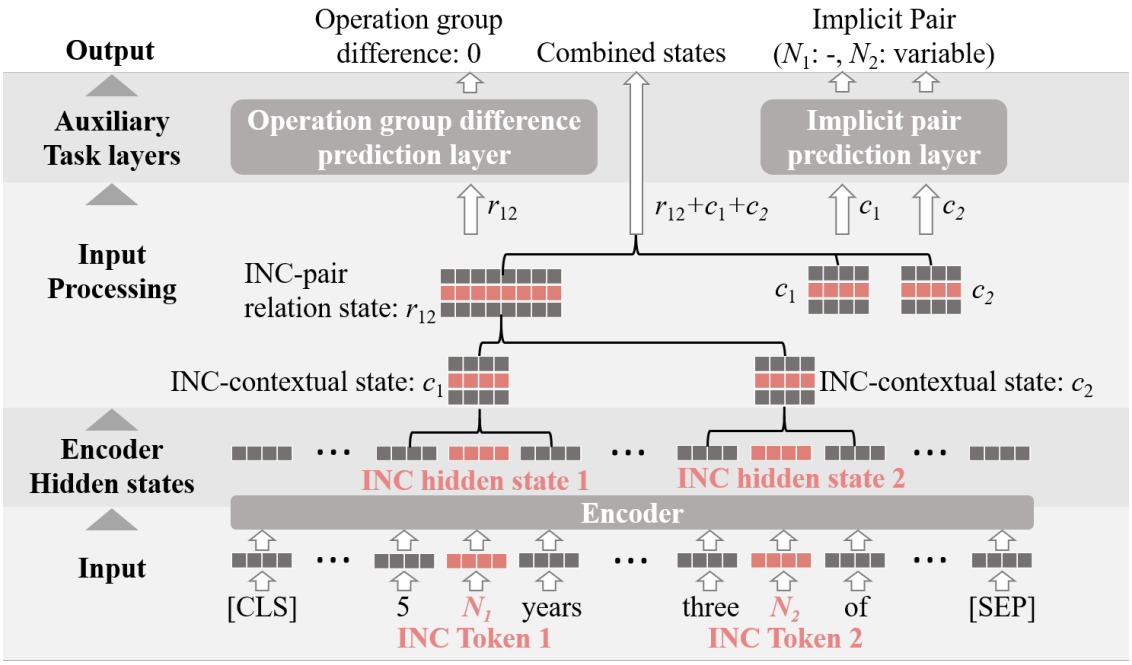

Figure 3: Auxiliary tasks : Operation group difference prediction and Implicit pair prediction

To address the issue of missing domain-specific knowledge feature in neural models, we added two auxiliary tasks, *operation group difference prediction* and *implicit pair prediction* to the base model's encoder learning process. Specifically, in math word problem solving, the missing domain-specific features are numbers and operation information. Figure 3 shows the overall pipeline for the two auxiliary tasks. As input to the two auxiliary tasks, INC-contextual state and INC-pair relation state are used. The 'INC-contextual' state consists of encoder hidden states that are obtained by applying windowing to a pair of two INC tokens. By applying windowing, the model can reflect the contextual information that can be derived from other tokens adjacent to the INC token. The INC-pair relation state is a concatenated state of two INC-contextual states. Below we detail the purpose and the learning process of each auxiliary task.

The first auxiliary task is operation group (OG) difference prediction. The purpose of OG difference prediction is to distinguish the INC token locations within an INC postfix equation. To calculate OG

difference prediction, the model takes the difference between operation group indices of two given INC tokens. A total of nC_2 OG differences are calculated, where n is the number of INCs, which are extracted from a math word problem. When multiple INC differences are calculated due to multiple occurrence of INC token in the problem, the minimum value is used to identify the relative difference in the operation in which the INC is located. For example, in Figure 2, to calculate the OG difference between N_1 and N_2, a total of four computations are possible: 5&6, 6&6, 6&8, 5&8. Among these four values, the model will use the minimum value of 0, which is the difference between 6&6. The equation formula of operation group difference prediction is presented below.

$$\mathrm{r}_{ij} = \mathrm{Concat}\left(\mathrm{e}_{i-\mathrm{w}:i+\mathrm{w}}, \mathrm{e}_{j-\mathrm{n}:j+\mathrm{n}}\right), \quad (1)$$

$$\mathrm{OperationGroupDiff}_{ij} = \arg\max\left(\mathrm{Softmax}\left(\mathrm{FF}_{\mathrm{GroupDiff}}\left(\mathrm{r}_{ij}\right)\right)\right), \quad (2)$$

where $\mathrm{e}_{i-w:i+w}$ indicates encoder hidden states from the $(i-w)$th token to the $(i+w)$th token, w is a window size, c_i denotes a INC contextual state of INC token for each token i, and r_{ij} denotes an INC pair relation state between the INC token i and j. Also, in the equation, $\mathrm{Concat}(\cdot)$, $\mathrm{FF}(\cdot)$, and $\mathrm{Softmax}(\cdot)$ are functions indicating concatenation, linear feed-forward, and softmax operation, respectively.

The second auxiliary task is implicit pair prediction. The implicit pairs are intended to capture relations which do not appear explicitly in the question, such as variable and operator, but are required in generating the equation. The procedure of implicit pair prediction is as follows. First, all of the INC are required to be bound with specific implicit pairs, so we set all of INC to have one implicit pair. The order of implicit pairs to be predicted are set according to priority: {variable, $-$, $\div$, $\times$, $+$, $=$}. A variable has the highest priority because in expression trees, variables are likely to have the same hierarchy as INC. The operators are then prioritized by order-sensitivity: order-sensitive operators such as $-$, $\div$, and order-insensitive operators such as $+$, $\times$. The equation formula of the implicit pair prediction is as follows.

$$\mathrm{c}_i = \mathrm{Concat}\left(\mathrm{e}_{i-\mathrm{w}:i+\mathrm{w}}\right) \quad (3)$$

$$\mathrm{Implicit}_i = \arg\max\left(\mathrm{Softmax}\left(\mathrm{FF}_{Implicit}\left(\mathrm{c}_i\right)\right)\right) \quad (4)$$

3.4 Operation Feature Feed Forward (OP3F) layer

In addition to the two auxiliary tasks, we designed the Operation Feature Feed Forward(OP3F) layer, an additional layer that can address the problem of losing encoder-level knowledge in the decoder. As shown in Figure 1, the GEO model uses the OP3F layer by generating equations using a concatenation of the encoder's combined states and decoder hidden states, which can add encoder-level knowledge to decoder hidden states. The combined states from the encoder are made by concatenating the mean of relation states with the mean of contextual states. The combined state is passed to the feed-forward layer and transformed into the same size as the decoder hidden states. After that, the combined state is concatenated with the decoder hidden state and passed to the OP3F layer to generate a new token. The formulas related to the OP3F layer are described as follows:

$$\mathrm{h}_{combined} = \mathrm{FF}_{dim}\left(\mathrm{Concat}\left(\mathbb{E}\left[r_{ij} : j > i\right], \mathbb{E}\left[\mathrm{c}_i\right]\right)\right) \quad (5)$$

$$\mathrm{t}_{i+1} = \arg\max_t\left(\mathrm{Softmax}\left(\mathrm{FF}_{opf}\left(\mathrm{Concat}\left(\mathrm{h}_{combined}, \mathrm{d}\right)\right)\right)\right). \quad (6)$$

4 Experimental Setup

To evaluate and understand the GEO's performance, we conducted two experiments using three benchmark datasets: ALG514, DRAW-1K, and MAWPS. Experiment 1 was designed to compare the problem-solving accuracy of the GEO against the SOTA models that either uses HCF or pure neural model. Experiment 2 was designed to ablate how each component of the GEO contributes to performance.

	ALG514	DRAW-1K	MAWPS
Dataset size			
Problems	514	1,000	2,373
Splits	5-fold	Train.600,Dev.200,Test.200	5-fold
Complexity of generating equations (per problem)			
Unknown	1.82	1.75	1.00
Expression tokens	13.08	14.16	6.20

Table 1: Characteristics of experimental datasets

4.1 Dataset and evaluation metric

In Table 1, three benchmark datasets were used for experiments. The datasets are ALG514 (Kushman et al., 2014), DRAW-1K (Upadhyay and Chang, 2016), MAWPS (Koncel-Kedziorski et al., 2016). We selected the above datasets according to size and complexity, since size and complexity can be used as the level of difficulty in the equation generation. Complexity is proportional to the number of unknowns and generated tokens. In Table 1, ALG514 and DRAW-1K are more difficult than MAWPS, because of the size and the number of unknown and equation tokens. We used answer accuracy as a metric to evaluate the exact performance of the GEO. After generating the INC mapped postfix equation using the GEO, `SymPy` (Meurer et al., 2017) solves the equation and gets the answer accuracy. For DRAW-1K, which has training and development sets, we report the accuracy of the development and the test set. For ALG514 and MAWPS, we report the average accuracy using 5-fold cross-validation.

4.2 Experiment 1: evaluating performance of the GEO

Two types of SOTA models, models using HCFs and pure neural models, were compared to the GEO model to evaluate the GEO model's performance. A brief description and performance of each model are as follows. HCF-based model is MixedSP (Upadhyay et al., 2016b) which reported 83.0%(ALG514) and 59.5%(DRAW-1K) using numeric quantity based HCF. Pure neural models are DNS (Wang et al., 2017), Robaidek et al. (2018), and Lee and Gweon (2020). DNS reported 70.0%(ALG514) by using a neural-based template retrieval model, which ensembled with neural model and TF-IDF. Robaidek et al. (2018) reported 53.5%(DRAW-1K) using classification method based on self-attention. Lee and Gweon (2020) reported 78.8%(MAWPS) using BERT-based multi-task learning method.

4.3 Experiment 2: understanding the impact of auxiliary tasks and OP3F layer

We set up experiment 2 to understand each component of the GEO appropriate for performance improvement. Thus, we used three ablation models: base model (Electra + transformer decoder), Aux model (base model + auxiliary tasks), GEO (base model + auxiliary tasks + OP3F layer). By comparing each model's performance, we wanted to validate each components' contribution to performance. Also, we wanted to get empirical evidence as to whether this improvement follows the hypotheses we set.

4.4 Implementation

We used `pyTorch` 1.5 (Paszke et al., 2019) to build GEO model. We used two Electra models, `large(L)` and `base(B)`. Both models are provided in `transformers` library(Wolf et al., 2019). Primarily we used `(B)` for training, and after confirming the performance, we applied `(L)` for two datasets to achieve performance improvement. Learning rate for encoders were 10^{-4} (ALG514$^{(B)}$, DRAW-1K$^{(B)}$, MAPWS$^{(B)}$), $5 \cdot 10^{-5}$ (ALG514$^{(L)}$, DRAW-1K$^{(L)}$), and learning rate for decoders were $2 \cdot 10^{-4}$ (ALG514, DRAW-1K, MAPWS$^{(B)}$). Batch sizes were 16 (ALG514$^{(L)}$, DRAW-1K$^{(L)}$), and 24 (ALG514$^{(B)}$, DRAW-1K$^{(B)}$, MAWPS$^{(B)}$). Warmup step ratios were 0.1 (ALG514$^{(B)}$, DRAW-1K$^{(B)}$, MAWPS$^{(B)}$), 0.03 (encoder of ALG514$^{(L)}$, DRAW-1K$^{(L)}$), and 0.1 (decoder of ALG514$^{(L)}$, DRAW-1K$^{(L)}$). Label smoothing cross entropy loss (Szegedy et al., 2016) was used to prevent overconfidence in the equation generation. Calculating the auxiliary task loss, we divided the loss by the number of each task and multiplied the batch size. Applying the OP3F layer, we used two linear feed-forward

layers to reduce learning parameters. For other hyperparameters, we used decoder layer 3, decoder heads 8, decoder feed-forward layer 2048, beam size 3 and window size were 3. All experiments of `(L)` were conducted on Google colab, while `(B)` were conducted in a local PC with 64GB RAM and RTX2070 Super and RTX2080 Ti.

5 Results & Discussion

5.1 Experiment 1

Model	ALG514	DRAW-1K		MAWPS
	(5-fold)	(Dev.)	(Test)	(5-fold)
Hand-crafted SOTA	**83.0**[MixedSP]	-	**59.5**[MixedSP]	-
Pure neural SOTA	70.1[DNS]	-	53.5[Robaidek]	**78.8**[T-MTDNN]
GEO	82.1 (L)	59.5 (L)	62.5 (L)	85.1 (B)

Table 2: Accuracy of the GEO and existing models. (L) and (B) denotes `electra-large` and -`base`.

As shown in Table 2, the GEO obtained comparable results to the existing SOTA in three datasets. In particular, it is noteworthy that the GEO showed significant performance improvement even in small-sized and highly complex datasets such as ALG514 and DRAW-1K. Specifically, the GEO showed an increased rate of about 12.0% in ALG514 compared to the existing neural model, and the increase rate was about 9% in DRAW-1K. These results show that GEO is especially effective when the model is required to generate a difficult equation.

5.2 Experiment 2

Model	ALG514	DRAW-1K		MAWPS
	(5-fold)	(Dev.)	(Test)	(5-fold)
Base	44.1±4.1	37.0	32.0	81.6±2.1
+Aux	64.4±2.8	52.0	47.0	84.7±1.3
+OP3F (GEO)	73.3±1.7	55.0	55.0	85.1±1.1

Table 3: Accuracy(%) of the GEO (`electra-base`) and its ablated models

As shown in Table 3, the base model showed the lowest performance. Next, when auxiliary tasks were added, the performance improved significantly in all three datasets. The perfomance improved by about 20.3% in ALG514, 15% in DRAW-1K, and 3.1% in MAWPS. Finally, when the OP3F layer was applied, a significant performance improvement of 8.9% (ALG514), 8% (DRAW-1K), and 0.4% (MAWPS) was observed. A small increase in MAWPS seems to be due to a ceiling effect since MAWPS has low complexity than the other two datasets. The experimental results confirmed that the GEO outperformed the prior studies or achieved comparable results. Notably, results shown in small datasets such as ALG514, suggest that a neural model without HCFs can achieve the SOTA level performance. Furthermore, via the ablation study, we suggest that the improvement is achieved by solving the two issues we identified: missing domain-specific knowledge features and losing encoder-level knowledge. An error analysis for the ablation study provides empirical evidence for our suggestion. Table 4 is a set of equations generated by three ablation models. We found two cases that enable understanding of the effect of each component of the model. Case 1 shows that auxiliary tasks can handle domain-specific knowledge, while Case 2 shows that the OP3F layer can prevent losing encoder-level knowledge.

In Case 1 of Table 4, the base model did not grasp the relationship of INCs such as N_1, N_2, and N_3, in a given math word problems. Thus we speculate that the base model generated tokens without considering location of INCs. Also, the base model obtained only 32% of test results in the DRAW dataset, which is about 21% less than the previous neural model SOTA. This result shows a similar tendency to the study of Robaidek (Robaidek et al., 2018). According to his study, when learning after adding ELMO to the

Case 1. base model did not grasp the relationship between INCs	
Question	A cashier has $30N_0$ bills, all of which are $10N_1$ or $20N_2$ bills. The total value of the money is $330N_3$. How many of each type does the cashier have ?
Answer eqution	$N_1 \times X_0 + N_2 \times X_1 = N_3, \quad X_0 + X_1 = N_0$
Base	$0.05 \times X_0 + N_1 \times X_1 = N_2, \quad X_0 + X_1 = N_0$ (**Incorrect**)
+Aux	$N_1 \times X_0 + N_2 \times X_1 = N_3, \quad X_0 + X_1 = N_0$ (Correct)
+OP3F(GEO)	$N_1 \times X_0 + N_2 \times X_1 = N_3, \quad X_0 + X_1 = N_0$ (Correct)
Case 2. OP3F layer is effective to use numeric information learned from the encoder.	
Question	A total of $\$50000N_0$ is invested in twoN_1 funds paying $8N_2\%$ and $8.5N_3\%$ simple interest. The annual interest is $\$4120N_4$. How much is invested in each fund?
Answer equation	$0.01 \times N_2 \times X_0 + 0.01 \times N_3 \times X_1 = N_4, \quad X_0 + X_1 = N_0$
Base	$0.01 \times N_2 \times X_0 + 0.01 \times N_3 \times X_1 = N_5, \quad X_0 + X_1 = N_0$ (**Incorrect**)
+Aux	$0.01 \times N_2 \times X_0 + 0.01 \times N_3 \times X_1 = N_5, \quad X_0 + X_1 = N_0$ (**Incorrect**)
+OP3F (GEO)	$0.01 \times N_2 \times X_0 + 0.01 \times N_3 \times X_1 = N_4, \quad X_0 + X_1 = N_0$ (Correct)

Table 4: Sample incorrect problems (`electra-large`) from DRAW-1K development set

classification model based on self-attention, the answer accuracy fell from 53% to 45.5% in DRAW-1K. In other words, the base model results seem to empirically support that the pretrained language model has a limit in addressing the problem of missing domain-specific knowledge features.

Case 2 of Table 4 presents evidence that the GEO model's OP3F layer utilizes the numeric information learned from the encoder directly in the generation process of the decoder. Both the base model and the Aux model generated a N_5 token, which is irrelevant to the question. However, the GEO model did not make the N_5 token. These results suggest that GEO's methodology could be useful for other generation-based tasks that may cause similar problems. Therefore, we will verify if the GEO's methodology could be applied to semantic parsing and sequential coding that require sequential decoding similar to math word problem solving using small datasets.

6 Conclusion

In this paper, we propose a novel encoder-decoder model, the GEO model for math word problem solving. The GEO model's performance accuracy outperformed the existing state-of-the-art models in two out of the three datasets. Also, our ablation study showed that two components of the GEO model, two auxiliary tasks and the OP3F layer, address the raised two issues: missing domain-specific knowledge features and missing encoder-level knowledge. As future work, we plan to apply our auxiliary tasks and OP3F layer to another task, especially including number information and abstraction is needed.

Acknowledgements

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. 2020R1C1C1010162).

References

Daniel Andor, Luheng He, Kenton Lee, and Emily Pitler. 2019. Giving bert a calculator: Finding operations and arguments with reading comprehension. *arXiv preprint arXiv:1909.00109*.

Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. 2020. Electra: Pre-training text encoders as discriminators rather than generators. *arXiv preprint arXiv:2003.10555*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Danqing Huang, Jing Liu, Chin-Yew Lin, and Jian Yin. 2018. Neural math word problem solver with reinforcement learning. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 213–223.

Kyung Seo Ki, Donggeon Lee, and Gahgene Gweon. 2020. Kotab: Korean template-based arithmetic solver with bert. In *2020 IEEE International Conference on Big Data and Smart Computing (BigComp)*, pages 279–282. IEEE.

Rik Koncel-Kedziorski, Subhro Roy, Aida Amini, Nate Kushman, and Hannaneh Hajishirzi. 2016. MAWPS: A math word problem repository. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1152–1157, San Diego, California, June. Association for Computational Linguistics.

Nate Kushman, Yoav Artzi, Luke Zettlemoyer, and Regina Barzilay. 2014. Learning to automatically solve algebra word problems. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 271–281.

D. Lee and G. Gweon. 2020. Solving arithmetic word problems with a templatebased multi-task deep neural network (t-mtdnn). In *2020 IEEE International Conference on Big Data and Smart Computing (BigComp)*, pages 271–274.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Yang Liu. 2019. Fine-tune bert for extractive summarization. *arXiv preprint arXiv:1903.10318*.

Yuanliang Meng and Anna Rumshisky. 2019. Solving math word problems with double-decoder transformer. *arXiv preprint arXiv:1908.10924*.

Aaron Meurer, Christopher P. Smith, Mateusz Paprocki, Ondřej Čertík, Sergey B. Kirpichev, Matthew Rocklin, AMiT Kumar, Sergiu Ivanov, Jason K. Moore, Sartaj Singh, Thilina Rathnayake, Sean Vig, Brian E. Granger, Richard P. Muller, Francesco Bonazzi, Harsh Gupta, Shivam Vats, Fredrik Johansson, Fabian Pedregosa, Matthew J. Curry, Andy R. Terrel, Štěpán Roučka, Ashutosh Saboo, Isuru Fernando, Sumith Kulal, Robert Cimrman, and Anthony Scopatz. 2017. Sympy: symbolic computing in python. *PeerJ Computer Science*, 3:e103, January.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Benjamin Robaidek, Rik Koncel-Kedziorski, and Hannaneh Hajishirzi. 2018. Data-driven methods for solving algebra word problems. *arXiv preprint arXiv:1804.10718*.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826.

Shyam Upadhyay and Ming-Wei Chang. 2016. Annotating derivations: A new evaluation strategy and dataset for algebra word problems. *CoRR*, abs/1609.07197.

Shyam Upadhyay, Ming-Wei Chang, Kai-Wei Chang, and Wen-tau Yih. 2016a. Learning from explicit and implicit supervision jointly for algebra word problems. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 297–306.

Shyam Upadhyay, Ming-Wei Chang, Kai-Wei Chang, and Wen-tau Yih. 2016b. Learning from explicit and implicit supervision jointly for algebra word problems. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 297–306, Austin, Texas, November. Association for Computational Linguistics.

Zalman Usiskin. 1988. Conceptions of school algebra and uses of variables. *The ideas of algebra, K-12*, 8:19.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Yan Wang, Xiaojiang Liu, and Shuming Shi. 2017. Deep neural solver for math word problems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 845–854.

Lei Wang, Dongxiang Zhang, Jipeng Zhang, Xing Xu, Lianli Gao, Bing Tian Dai, and Heng Tao Shen. 2019. Template-based math word problem solvers with recursive neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7144–7151.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in neural information processing systems*, pages 5753–5763.