Generating Math Word Problems from Equations with Topic Consistency Maintaining and Commonsense Enforcement

Tianyang Cao[1,2], Shuang Zeng[1,2], Songge Zhao[1], Mairgup Mansur[3], and Baobao Chang[1,2(✉)]

[1] Key Laboratory of Computational Linguistics, Peking University, MOE, Beijing, China
{ctymy,zengs,zhaosongge}@pku.edu.cn
[2] School of Software and Microelectronics, Peking University, Beijing, China
chbb@pku.edu.cn
[3] Sogou Technology Inc., Beijing, China
maerhufu@sogou-inc.com

Abstract. Data-to-text generation task aims at generating text from structured data. In this work, we focus on a relatively new and challenging equation-to-text generation task – generating math word problems from equations and propose a novel equation-to-problem text generation model. Our model first utilizes a template-aware equation encoder and a Variational AutoEncoder (VAE) model to bridge the gap between abstract math tokens and text. We then introduce a topic selector and a topic controller to prevent topic drifting problems. To avoid the commonsense violation issues, we design a pre-training stage together with a commonsense enforcement mechanism. We construct a dataset to evaluate our model through both automatic metrics and human evaluation. Experiments show that our model significantly outperforms baseline models. Further analysis shows our model is effective in tackling topic drifting and commonsense violation problems.

Keywords: Math word problem generation · Topic controlling · Commonsense enforcement

1 Introduction

Text generation, aiming to automatically generate fluent, readable and faithful natural language text from different types of input, has become an increasingly popular topic in NLP community.

Many recent text generation approaches [3,6,7,10,13,14] focus on data-to-text generation task, which generates textual output from structured data such as tables of records or knowledge graphs (KGs). However in this paper, we focus on a relatively new type of data-to-text generation task: generating math word

I. Farkaš et al. (Eds.): ICANN 2021, LNCS 12893, pp. 66–79, 2021.
https://doi.org/10.1007/978-3-030-86365-4_6

Equations: *y = 3 * x ; y – 20 = 100* **Problem:** The teacher said if you multiply my age by ***3***, then subtract ***20***, the result is ***100***. ***How old is the teacher***?
Equations: *1 – 1/4 – 2/5 = x* **Problem:** If ***1/4*** of a pie is eaten and later ***2/5***, ***what is the remaining fraction***?

Fig. 1. Two examples selected from the MWP generation dataset.

Bad Case 1: The adult hat is $50 for sale. How much of each kind should be used to make a mixture that is 70% protein …
Bad Case 2: The hypotenuse of a square is 7.9 meters long, ...
Bad Case 3: What is the dimensions of the original number, ...

Fig. 2. Three bad cases generated by baseline (Seq2Seq) model.

problems (MWP) from equations [15], which seems has not been fully studied by NLP community. Successful math word problems generation has the potential to automate the writing of mathematics questions. Thus it can alleviate the burden of school teachers and further help improve the teaching efficiency (Fig. 1).

Our target is to design a model to generate math problem text from the given equations and the generated math problem could be solved by the equations. Different from the traditional text generation task, there are three major challenges in effective MWP generation from equations:

(1) Encoding math equations is much more difficult than encoding plain text, tables or KGs. A math equation consists of different type of tokens, such as number, variable and operator. They express different meanings and are very abstract for generating text. So a model should use different methods to encode them and need to bridge the gap between the abstract math tokens and natural language text.
(2) Recent language modeling advancements indeed make generated text more fluent, but still lacking of coherence, especially in the aspect of topic drifting, has always been a non-trivial problem that traditional text generation models usually suffer from [4]. And we find this problem is even worse in MWP generation since target math word problems in MWP dataset covers a broad variety of topics. Figure 2 shows three bad cases generated by a Seq2Seq model. The first case reveals the topic drifting problem where the topic of the first generated sentence is the *price of goods* but the topic of the second sentence is changed to *substance mixture*. So how to maintain the topic consistency in generated text to avoid topic drifting is very challenging. (3) The task requires generated problem text to be in line with the commonsense which is very hard for existing architecture. As shown in the last two cases in Fig. 2, we cannot say "hypotenuse of a square" or "dimension of a number", because they aren't in accordance with the commonsense. So we should design an effective architecture to avoid commonsense violation problem.

To tackle these challenges, we propose a novel architecture for generating MWP from equations. First, to effectively encode different kinds of math tokens in the given equations, we propose a template-aware equation encoder that considers both template information and equation information. We further utilize

a problem-aware Variational AutoEncoder (VAE) with a Kullback-Leibler distance loss to bridge the gap between abstract math tokens and problem text. Then we propose a topic selection mechanism that selects a fixed topic for each generated text and a topic controlling mechanism that controls the topic at every decoding step to avoid topic drifting problem. To cope with the possible commonsense violation issue of generated text, we design a pre-training stage as well as a commonsense enforcement module to encourage our model to generate math problem text that is in line with the commonsense.

Our contributions can be summarized as follows:

- We propose an effective way of encoding different math tokens and a problem-aware VAE to bridge the gap between abstract math tokens and generated text.
- We utilize a topic selection and a topic controlling mechanism, so topic consistency of generated math problem text could be maintained.
- We design a pre-training stage and a commonsense enforcement module to alleviate commonsense violation.

In order to verify the effectiveness of our model, we construct a dataset by obtaining math word problems and their corresponding equations from *Yahoo!*[1]. Experimental results on this dataset show our model significantly outperforms previous models. Further analysis and human evaluation show our model is effective in tackling the three challenges mentioned before, especially the topic drifting and commonsense violation problems.

2 Task Definition

The input of MWP generation task is a set of equations $\{E_1, E_2, ..., E_{|E|}\}$, each equation can be denoted as a sequence of math tokens: $E_k = x_1 x_2 ... x_{|E_k|}$, where $|E_k|$ is the length of k-th equation measured by the number of math tokens. Each math token belongs to one of the following three types: math operator (e.g., $+, -, *, \div, =, ...$), number (e.g., $0.2, 1, 30, ...$), variable (e.g., $x, y, z, ...$). The output of the task is the MWP text: $\boldsymbol{y} = y_1 y_2 ... y_L$, which could be solved by the input equations. L is the length of problem text. Our model aims to estimate the following conditional probability depending on equations and previously generated words $\boldsymbol{y}_{<t}$:

$$P(\boldsymbol{y}|\boldsymbol{x}) = \prod_{t=1}^{L} P(y_t|\boldsymbol{y}_{<t}, E_1, E_2, ...) \tag{1}$$

The difficulty of the input equations in this task is not beyond middle school level, only involving algebra operation in elementary mathematics: "$+, -, *, /, \wedge, ...$".

[1] https://github.com/caotianyang/math2textcs1.

3 Model

The overall architecture of our model is shown in Fig. 3. We start with a variational encoder-decoder model as our base model which consists of a template-aware equation encoder and a math word problem generation decoder. Since the math tokens in the original input equation are very abstract and lack enough context information for generating text, we introduce a problem-aware Variational AutoEncoder to encourage the equation encoder to produce text-sensitive representation that is more suitable for decoding problem text.

To tackle the problem of topic drifting, we introduce a topic selector with a topic controller. The topic selector chooses a specific topic based on the latent representation of equations. The dynamic topic memory is used to control the decoding process to favor the topic-consistent text. To alleviate the commonsense violation, we introduce a pre-training step to produce commonsense embedding for words and use a commonsense enforcement module to aggregate commonsense information which will influence the following choice at each step of decoding.

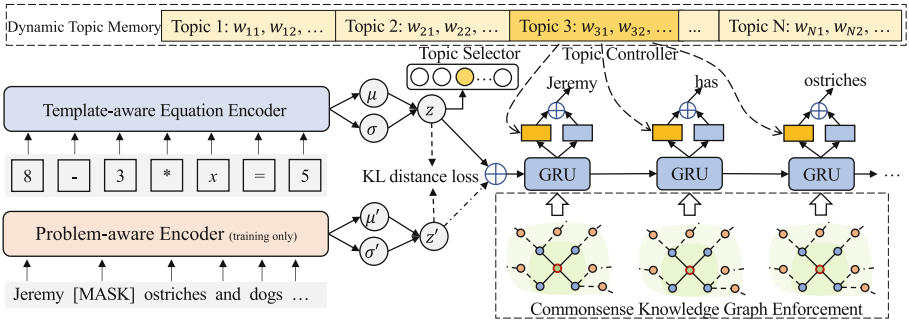

Fig. 3. The overview of our proposed model. We omit the pretraining step for simplicity. In our variational autoencoder enhanced model, the problem encoder serves as prior network and the equation encoder serves as posterior network. Topic type predicted by the hidden equation representation z is used to select the corresponding row in topic memory. Next, the MWP decoder resorts to both the dynamic topic memory and the Commonsense KG reasoning to generate MWP text.

3.1 Variational Encoder-Decoder Module

As we mentioned before, we choose the variational encoder-decoder model as the basic model to generate the MWP text from equations. The backbone of our model consists of a template-aware equation encoder and a problem text decoder.

Template-Aware Equation Encoder: The input to our model is a sequence of math tokens $x_1 x_2 ... x_n$, our input encoder encodes each token to one fixed

hidden vector. Math equations encoding is different from encoding other natural languages, we should distinguish numbers, variables and operations to assign them different encoding.

We exploit BiGRU as the basic module, it consumes token embedding of the equation sequences and the hidden states are computed by: $\overleftarrow{\boldsymbol{h}}_i = GRU(emb(x_i), \overleftarrow{\boldsymbol{h}}_{i-1})$, $\overrightarrow{\boldsymbol{h}}_i = GRU(emb(x_i), \overrightarrow{\boldsymbol{h}}_{i-1})$. $emb(x_i) = \boldsymbol{E}_{token}(x_i) + \boldsymbol{E}_{type}(x_i)$ is the sum of corresponding token embedding and type embedding. Combining forward and backward state yields $\boldsymbol{h}_i = \overleftarrow{\boldsymbol{h}}_i + \overrightarrow{\boldsymbol{h}}_i$.

To improve the generalization capacity of the equation encoder, we further incorporate a soft gate controlled by equation template. The equation template is constructed by replacing all numbers in the equation to a fixed mask [M]. We separately feed the original sequence and the template sequence into two different GRUs, then encoded hidden states are denoted as $\{\boldsymbol{h}_{a,1}, \boldsymbol{h}_{a,2}, ..., \boldsymbol{h}_{a,n}\}$ and $\{\boldsymbol{h}_{b,1}, \boldsymbol{h}_{b,2}, ..., \boldsymbol{h}_{b,n}\}$, respectively. Here we utilize Gated Linear Unit (GLU) [5] to compute final encoded hidden state:

$$\boldsymbol{h}_k = MLP_1(\boldsymbol{h}_{a,k}) \odot \sigma(MLP_2(\boldsymbol{h}_{b,k})) \; 1 \leq k \leq n \tag{2}$$

where $\sigma(\cdot)$ is a sigmoid function and $MLP(\cdot)$ is a linear layer. $\odot$ indicates element-wise multiplication. $\boldsymbol{h}_{b,k}$ can be understood as a weight matrix to select salient information in $\boldsymbol{h}_{a,k}$. We perform linear transformation to $\boldsymbol{h}_n$ and approximate mean and variance of z's posterior distribution by assuming the hidden equation representation z follows multivariate Gaussian distribution.

$$[\boldsymbol{\mu}, \boldsymbol{\sigma}] = MLP(\boldsymbol{h}_n) \quad z|\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2\mathbf{I}) \tag{3}$$

$\mathbf{I}$ is an identity matrix and z can then be sampled by using reparameterization trick: $\boldsymbol{z} = \boldsymbol{\mu} + \boldsymbol{r} \odot \boldsymbol{\sigma}$, where $\boldsymbol{r}$ is a standard Gaussian distribution variable.

Problem Decoder: For generating problem text, we use GRU based decoder. We first initialize the decoder state by $\boldsymbol{s}_0 = MLP([\boldsymbol{h}_n; \boldsymbol{z}; \boldsymbol{h}_n \odot \boldsymbol{z}])$. We denote the hidden state of the decoder at tth step as $\boldsymbol{s}_t$ and context vector obtained by attentions over the input equation as $\boldsymbol{c}_t$. Assume the decoder generates word w_{t-1} in step $t-1$, decoding process can be formulated by:

$$\boldsymbol{s}'_t = f(\boldsymbol{s}_t) \quad \boldsymbol{s}_t = GRU(\boldsymbol{s}_{t-1}, g(\boldsymbol{e}_{w_{t-1}})) \tag{4}$$

$$p_D(y_t|y_{<t}, \boldsymbol{x}, z, \hat{p}; \theta_D) = softmax(\boldsymbol{W}^o tanh(\boldsymbol{W}^{vs}[\boldsymbol{s}'_t; \boldsymbol{c}_t])) \tag{5}$$

where $\boldsymbol{W}^{vs} \in \mathbb{R}^{d \times d}$, $\boldsymbol{W}^o \in \mathbb{R}^{d \times |V|}$. $|V|$, $\hat{p}$ and d is the vocabulary size, topic category and embedding size, respectively. $f(\cdot)$ and $g(\cdot)$ is designed for leveraging topic restriction and commonsense restriction, respectively, which will be explained later. We further adopt copy mechanism [11] to copy numbers from equations.

3.2 Enhancing Equation Encoder by Variational Autoencoder

Hidden equation representation $\boldsymbol{z}$ derived by (3) fails to capture interaction between equations and MWP text. We thus introduce a problem-aware VAE

to further restrict $\boldsymbol{z}$ into similar vector space of MWP text to obtain problem text aware representation. In this paper, the VAE is comprised of the problem encoder and the problem decoder. As the problem text is known when training, posterior distribution of z generated by the equation encoder is conditioned on prior distribution generated by the problem encoder.

The problem encoder summarizes the MWP text to a vector $\boldsymbol{q}$ and works as a prior network. It takes the corrupted version of problem text $\boldsymbol{y}$ as input to guarantee robustness when testing, i.e., we randomly mask and delete some words in the original MWP text. We implement the problem encoder module based on convolutional neural network (CNN) with F different convolutional kernels to extract multi-scale features:

$$\boldsymbol{h}_k^q = MaxPool(f_{conv}(\left[\boldsymbol{y}_i; \boldsymbol{y}_{i+1}; ...; \boldsymbol{y}_{i+l_k-1}\right])) \tag{6}$$

$$\boldsymbol{q} = tanh(\boldsymbol{W}^q \left[\boldsymbol{h}_1^q; \boldsymbol{h}_2^q; ...; \boldsymbol{h}_F^q\right]) \tag{7}$$

where $\boldsymbol{W}^k \in \mathbb{R}^{dl_k}$ is the kth convolutional kernel and parameter matrix $\boldsymbol{W}^q \in \mathbb{R}^{Fd \times d}$. Similar to (3), we perform linear transformation to $\boldsymbol{q}$ and obtain mean and variance of z's prior distribution: $[\boldsymbol{\mu}', \boldsymbol{\sigma}'] = MLP(\boldsymbol{q})$ $z'|\boldsymbol{y} \sim \mathcal{N}(\boldsymbol{\mu}', \boldsymbol{\sigma}'^2\mathbf{I})$.

We denote the problem decoder parameterized by θ_D as $p_D(\boldsymbol{y}|\boldsymbol{x}, z, \hat{p}; \theta_D)$, during training, $\boldsymbol{z}$ is obtained by prior network. We aim to minimize Kullback-Leibler distance (KL loss) between prior distribution and posterior distribution. Loss function of our Variational Encoder-Decoder framework can then be computed by combining KL loss and generator decoding loss:

$$\begin{aligned}\mathcal{L}_{VAE} = &-KL(p(z|\boldsymbol{y})||p(z|\boldsymbol{x})) \\ &+ \mathbb{E}_{z\sim\mathcal{N}(\boldsymbol{\mu}', \boldsymbol{\sigma}'^2\mathbf{I})} p_D(\boldsymbol{y}|\boldsymbol{x}, z, \hat{p}; \theta_D)\end{aligned} \tag{8}$$

Besides, we use KL cost annealing to avoid KL-vanishing phenomenon [2]. During inference, $\boldsymbol{z}$ is approximated by posterior network.

3.3 Topic Selection and Controlling

Generally speaking, given an input equation, for example, $0.5 * x + 0.3 * y = 10$, our model should first select a certain type of topic and then incorporate related topic words under this type into the problem decoder.

Topic Selection: To leverage topic background to the hidden equation representation $\boldsymbol{z}$, we apply an unsupervised document topic model– Latent Dirichlet Allocation (LDA) [1] to assign a topic type for each math problem text. We treat each math question as a document, each document is associated with a topic distribution over all topics, meanwhile each topic contains several words with the highest probability in this topic. We then estimate the problem topic type through $\boldsymbol{z}$:

$$\hat{p} = \arg\max softmax(\boldsymbol{W}_z \boldsymbol{z} + \boldsymbol{b}_z) \tag{9}$$

Topic Controling: Topic controling renders our generator to interact with topic word distribution. With the help of LDA, a topic memory $\boldsymbol{C} \in \mathbb{R}^{|P| \times K \times d}$ is

constructed for storing pretrained embedding of topic keywords, where $|P|$ is the total topic number. K means each row of $\boldsymbol{C}$ contains information of top-K words of one topic and d is the vector dimension. With the most probably topic type $\hat{p}$ predicted in (9), the concatenation of $\boldsymbol{s}_t$ and $\boldsymbol{c}_t$ is used as a query to the $\hat{p}$th row of topic memory and update $\boldsymbol{s}_t$ with the weighted sum of topic embedding in $\boldsymbol{C}$:

$$score(t,j) = \frac{\exp([\boldsymbol{s}_t;\boldsymbol{c}_t]\,\boldsymbol{W}^t\boldsymbol{C}_{\hat{p},j})}{\sum_{j=1}^{K}\exp([\boldsymbol{s}_t;\boldsymbol{c}_t]\,\boldsymbol{W}^t\boldsymbol{C}_{\hat{p},j})}\ 1 \le j \le K \tag{10}$$

and $f(\boldsymbol{s}_t)$ in (4) is realized by:

$$f(\boldsymbol{s}_t) = \boldsymbol{s}_t + \boldsymbol{V}\sum_{j=1}^{K} score(t,j)\boldsymbol{C}_{\hat{p},j} \tag{11}$$

where $\boldsymbol{W}^t \in \mathbb{R}^{2d\times d}$ and $\boldsymbol{V} \in \mathbb{R}^{d\times d}$ serves for linear projection. Futhermore, memory contexts are initialized by the pretrained word representation, but during the generating process, it should be dynamicly updated with the produced sequence to keep recording new information, thus the topic memory can provide better guidance for the generator. We achieve this goal by computing a weight vector with a gated mechanism to weight in what degree the topic memory should be updated, then we obtain candidate state based on $\boldsymbol{s}'_t$ and $\boldsymbol{C}_{\hat{p},j}$, where $\boldsymbol{W}^u, \boldsymbol{W}^c \in \mathbb{R}^{d\times d}$:

$$\boldsymbol{u} = \sigma(\boldsymbol{W}^u\,[\boldsymbol{s}'_t;\boldsymbol{C}_{\hat{p},j}]) \tag{12}$$

$$\tilde{\boldsymbol{C}}_{\hat{p},j} = tanh(\boldsymbol{W}^c\,[\boldsymbol{s}'_t;\boldsymbol{C}_{\hat{p},j}]) \tag{13}$$

$$\boldsymbol{C}_{\hat{p},j} = \boldsymbol{u}\otimes\tilde{\boldsymbol{C}}_{\hat{p},j} + (\boldsymbol{1}-\boldsymbol{u})\otimes\boldsymbol{C}_{\hat{p},j} \tag{14}$$

3.4 Commonsense Enforcement

We argue it's beneficial to make our network leverage context-related concepts. We thus implement commonsense enforcement in two aspects: word knowledge pretraining and commonsense aware generator.

Word Embedding Pretraining For Commonsense Enforcement: We directly enrich information of our generator by pretraining word-level representation in an external commonesense KB. Note that word embedding pretraining is an off line step and is based on Graph Attention Network (GAT) [12]. For detail, see the Appendix.

Commonsense Aware Generator: In decoding phase, we merge neighbour nodes information in commonsense KB of generated words in the previous step to inject commonsense knowledge into our generator. For example, if "original cost" has been generated, we hope next sequence is "of the stock", other than "of the volume", for stock has the property "cost". Assume the decoder generates word w_{t-1} in step $t-1$, we extract a sub-graph within two-hop paths starting

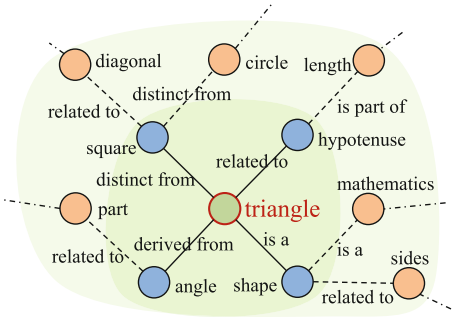

Fig. 4. Illustration of searching adjacent nodes. For word "triangle", first-order neighbors in knowledge graph are colored in blue while second-order neighbors are colored in orange. (Color figure online)

from w_{t-1} by Breadth First Search (BFS), as is shown in Fig. 4. Let $\boldsymbol{e}_{ij}$ denote the path representation from node i to node j if i and j are directly connected:

$$\boldsymbol{e}_{ij} = \phi(\boldsymbol{W}^g [\boldsymbol{e}_i; \boldsymbol{e}_j]) \tag{15}$$

where $\boldsymbol{W}^g \in \mathbb{R}^{2d \times d}$. If i and j are connected via intermediate node k, we aggregate the shortest path representation from i to j to obtain $\boldsymbol{e}_{ij}$:

$$\boldsymbol{e}_{ij} = \alpha\phi(\boldsymbol{W}^g [\boldsymbol{e}_i; \boldsymbol{e}_j]) + (1-\alpha)\sigma(\boldsymbol{e}_{ik} \otimes (\boldsymbol{U}\boldsymbol{e}_{kj})) \tag{16}$$

where $\boldsymbol{U} \in \mathbb{R}^{d \times d}$, $\phi(\cdot)$ is a nonlinear function, in this paper we use $tanh(\cdot)$. $\sigma(\cdot)$ is Sigmoid function. $\alpha \in [0, 1]$ is a scalar to control the contribution of direct and indirect information. Denote first order neighbour set and second order neighbour set of w_{t-1} as $\mathcal{N}_1(w_{t-1})$ and $\mathcal{N}_2(w_{t-1})$, respectively. We use an attention mechanism to tend to all possible paths, i.e., we calculate the aggregate summary of $\boldsymbol{e}_{w_{t-1},j}$ when j goes through $\mathcal{N}_1(w_{t-1}) \cup \mathcal{N}_2(w_{t-1})$:

$$\beta_{t-1,j} \propto \exp(\boldsymbol{e}_{w_{t-1}} \boldsymbol{W}^b \boldsymbol{e}_{w_{t-1},j}) \tag{17}$$

$$\boldsymbol{g}_{t-1} = \sum_{j \in \mathcal{N}_1(w_{t-1}) \cup \mathcal{N}_2(w_{t-1})} \beta_{t-1,j} \boldsymbol{e}_{w_{t-1},j} \tag{18}$$

Followed by (18), to better reflect the effect of concept knowledge to word choice, we combine $\boldsymbol{e}_{w_{t-1}}$ with $\boldsymbol{g}_{t-1}$ to realize $g(\boldsymbol{e}_{w_{t-1}})$ in (4):

$$g(\boldsymbol{e}_{w_{t-1}}) = GRU(\boldsymbol{e}_{w_{t-1}}, \boldsymbol{H}\boldsymbol{g}_{t-1}) \tag{19}$$

3.5 Training Objective

We aggregate 1): VAE loss mentioned in (8) 2) auxiliary topic prediction loss $\mathcal{L}_{topic} = \mathbb{E}_{z \sim \mathcal{N}(\boldsymbol{\mu}', \boldsymbol{\sigma}'^2 \mathbf{I})} p(\hat{p}|z, \boldsymbol{x})$ to obtain total loss:

$$\mathcal{L}_{total} = \mathcal{L}_{VAE} + \mu\mathcal{L}_{topic} \tag{20}$$

where μ is a hyperparameter.

4 Experiments

4.1 Datasets

Dolphin-18K [16] is the largest MWP dataset with various types of MWP text, while only a part of it (3154) are released. We then reuse the python script provided by [16] to crawl and collect data from Yahoo, which extends Dolphin-18K to 9643 samples in total. Statistic information of our data is listed in Table 3. We conduct some data preprocessing by deleting those equation-problem text pairs whose problem text length is longer than 45 tokens, besides, we replace those words appearing less than 2 times to ⟨UNK⟩.

4.2 Motivation of Creating New Dataset

MWP solving datasets currently used include Alg514 [18], Dolphin1878 [19], DRAW-1K [20], Dolphin18K [16]. Table 3 gives the statistic of these datasets. Alg514, Dolphin1878, DRAW-1K are all public available, while neural generation models for generative tasks are usually data-hungry thus equation-MWP pairs in those datasets are insufficient. Though Dolphin18K is a large scale dataset, only a part of it (3154) are released. Moreover, existing datasets only include a certain type of MWP text, e.g., MWP text for linear equations, which restricts their practical application. We then reuse the python script provided by [16] and acquire 14943 equation-MWP text pairs in total from Yahoo !. Generally, the public available datasets can be treated as the subset of our dataset. Next, we conduct data preprocess as follows, which is beneficial to train the generation model:

- We normalize the equations by replacing all the equation variables in each sample to $x, y, z, ...$ in order, e.g., $u + v + r = 100, u - r = 10$ is replaced to $x + y + z = 100,\ x - z = 10$.
- We manually correct the wrong spelling words in MWP text (Table 1).

Table 1. Statistics of several existing MWP solving datasets. Avg EL, Avg Ops refer to average equation length and average numbers of operators in equations, respectively. $*$ indicates only 3154 equation-MWP pairs of Dolphin18K are available.

Dataset	Size	Problem type	Avg EL	Avg Ops
Alg514	514	Algebra, linear	9.67	5.69
Dolphin1878	1878	Number word problems	8.18	4.97
DRAW-1K	1000	Algebra, linear, one-variable	9.99	5.85
Dolphin18K	18460*	Algebra, linear, multi-variable	9.19	4.96
Our Dataset	14943	Algebra, linear/nonlinear, multi-variable	16.64	6.41

4.3 Model Settings

The batch size for training is 32. We employ ConceptNet5[2] to construct KB, it has 34 types of relationship in total. 2 layer graph attention network is implemented for word knowledge pretraining step. The embedding size and all hidden state size of GRU are set to 256. In problem encoder three different convolutional kernels are used and their kernel sizes are 2,3,4, respectively. To be fair, we use 1-layer GRU for both our model and baseline. For LDA we divide all samples into 9 topic types and their amount and representative words are reported in Table 2. Each problem is associated with a topic distribution over 9 topics. The topic with the highest probability is adopted as the golden category. Meanwhile each topic contains several words and we choose top 30 words to construct topic memory for each topic. μ in (20) is set to 0.5. Weight coefficient in (16) is set to $\alpha = 0.7$. We use Adam optimizer [17] to train our model, the learning rate is set to 0.0005.

Table 2. Topic classes statistics and representative words sampled from each topic

Topic type	Train	Representative words
1	810	Do, people, divided, men, mean...
2	758	Length, width, rectangle, area, inches...
3	1573	Probability, quarter, dimes, coins, marbles...
4	557	Sum, difference, larger, smaller, less...
5	1087	Solution, gallon, mixture, grams, water...
6	633	Interest, year, invested, dollars, rate...
7	663	Angles, degrees, percent, digit, increased...
8	879	Sold, ticket, prices, children, adult...
9	754	Speed, minutes, travels, took, plane...

4.4 Automatic Evaluation

We report automatic evaluation in five aspects: BLEU (up to bigrams) [9], ROUGE-L [8], Dist-1, Dist-2, which indicates the proportion of different unigrams (bigrams) in all unigrams (bigrams), Number recall, which is used to measure how many numbers in problem text are correctly copied. Results are reported in Table 4. In Table 4 we also present results of ablation study. We can observe 1) our model yields higher performance in all metrics compared with baselines, especially in Dist-1 and Dist-2, which proves our model can generate more diversity math word problems. We consider this is because baseline models have no guidance in topic words and knowledge, thus they tend to generate the simplest question type like "one number is twice the second number...".

[2] https://github.com/commonsense/conceptnet5.

2) taking out topic control or commonsense enhancement will both decrease evaluation scores, which verifies their effectiveness. For example, removing commonsense enhancement declines BLEU score by 24.4%, while removing VAE & topic memory declines BLEU score by 35.5%.

We also separately compare MAGENT with our model including the same keywords as an extra input in Table 5, which demonstrates our model can still achieve performance gain with the same input.

Table 3. Statistic of datasets.

	Train	Dev	Test
Size	7714	964	965
Equation Length (average)	16.69	16.23	16.63
Problem Length (average)	28.90	29.64	28.74
Tokens	7445	3065	2875

Table 4. Automatic results in test dataset with BLEU, ROUGE-L (ROU), Dist-1 (D1), Dist-2 (D2) and Number Recall (NR). TP, TM and V denote the equation template, topic memory and VAE, respectively. CE includes both the pretraining step and the commonsense enforcement for the decoder.

Model	BLEU	ROU	D1 (%)	D2 (%)	NR (%)
Seq2seq	0.0259	0.2025	14.56	34.99	47.60
SeqGAN	0.0262	0.1922	12.96	30.02	44.00
DeepGCN	0.0304	0.2094	16.81	45.17	49.21
Transformer	0.0277	0.2036	16.69	37.57	50.89
Our model	**0.0433**	**0.2415**	**20.84**	53.81	55.14
w/o TP	0.0385	0.2377	18.88	**57.41**	**55.84**
w/o CE	0.0327	0.2273	18.75	51.19	54.42
w/o TM	0.0345	0.2256	20.00	55.00	54.31
w/o V & TM	0.0280	0.2141	18.79	50.05	53.82

Table 5. Comparison between our model with keywords (KW) and MAGNET in automatic results

Model	BLEU	ROU	D1 (%)	D2 (%)	NR (%)
MAGNET	0.0976	0.3793	**21.72**	57.22	42.62
Our model (KW)	**0.1152**	**0.4006**	18.81	**58.85**	**51.50**

4.5 Human Evaluation

Automatic metrics such as BLEU and ROUGE only focus on n-gram similarity, but fail to measure true generation quality (i.e., if topic drifting occurs). We invite three human annotators to judge generation quality in four aspects. 1) **Fluency (Flu)**: it mainly judges whether the problem text is fluent, i.e., whether the generated problem text has some grammar errors. 2) **Coherence (Coh)**: it weights if the problem text is coherent in text-level; 3) **Solvability-1 (S1)**: as our target is a math word problem, we should pay attention to whether the problem text can be solved, i.e., in what percentage we can set up the same (or equivalent) equations and solve them according to the generated problem text; 4) **Solvability-2 (S2)** is a more relaxed criterion compared with Solvability-1, it only requires the text produced is a valid math problem and could be solved regardless what equations could be set. We randomly select 50 generated MWP texts and score them in five grades. The scores are projected to 1–5, where higher score implies better performance (for solvability we use percentage). We report the average scores in Table 6.

Table 6 (*upper*) confirms our proposed model receives significant higher score in coherence and solvability, we assume this is because our model restricts the problem text into a certain topic and provides related words for reference.

In Table 6 (*bottom*) we report comparison between our model with keywords and MAGNET. Human scores reflect that our method achieves 12% relative improvement over MAGNET in Solvability-1. Especially, with keywords fed into the model, the problem of topic drifting is no longer notable for both our model and MAGNET.

Table 6. Human evaluation results: comparison between the proposed model and baseline models.

	Flu	Coh	S1 (%)	S2 (%)
Our model	**4.03**	**4.02**	**35**	**55**
Seq2seq	3.78	3.48	23	34
SeqGAN	3.75	3.28	20	40
DeepGCN	3.61	3.55	29	52
Transformer	3.80	3.53	20	45
MAGNET	4.00	4.33	44	**76**
Our model (KW)	**4.27**	**4.60**	**56**	74

4.6 Case Study

Table 7 shows some math word problems generated by different models. It's easy to show problem text generated by Seq2seq suffers from lack of coherence, e.g., in the above case, the baseline result talks about different topics in the same sentence. As a comparison, our generator discusses the same topic and generates

words around this topic. What's more, the topic of problem text generated by our proposed model is highly consistent with reference answer, which verifies the effectiveness of the proposed model.

We can also observe commonsense violation appears in baseline results, for example, "chemist has a perimeter" and "geometric is 4 more than" are obviously illogical. Relatively speaking, MWP text generated by our model, is more in line with commonsense, such as "the hypotenuse of a right triangle". These results reflect that our model can benefit from both the topic consistency maintaining and commonsense enforcement mechanism.

Table 7. Three examples of math word problems generated by different models. Transformer is abbreviate to Trans. Topic words in the left column indicate the overlap between selected topic words and the generated MWP text, which is also highlighted in the right column. CG reflects the reasoning procedure adopted by the decoder.

Equation: $equ : 4 * (x - y) = 800$ $equ : 2 * (x + y) = 800$ **Reference**: An airplane travels 800 miles against the wind in 4 hrs and makes the return trip with the same wind in 2 hrs . Find the speed of the wind. **Topic Words**: travels, miles, speed **CG**: travel $\xrightarrow{RelatedTo}$ trip travel $\xrightarrow{RelatedTo}$ take	**Ours**: A plane travels 800 miles in 4 hours . the return trip against the same wind took 4.5 hours. Find the speed of the current wind. **Seq2seq**: A chemist has a perimeter of 80 cm, the area of the rectangle is 800 m. what is the length of the rectangle. **SeqGAN**: The perimeter of a rectangle is 800 inches. Find the length. **DeepGCN**: A man has 800 more than four times as old as his son . If the current is 800 m. Find the speed of the plane. **Trans**: The sum of two numbers is 800. The sum of their squares is 25% . Find the numbers.
Equation: $equ : 0.1 * x + 0.05 * y = 1.95$ $equ : y = x + 3$ **Reference**: Ken has $ 1.95 in nickels and dimes . There are 3 more nickels than dimes. How many of each does he have? **Topic Words**: nickels, dims, ticket **CG**: nickel $\xrightarrow{IsA}$ coin $\xrightarrow{RelatedTo}$ dime	**Ours**: Arnold has some nickels and dimes . If he made a number of $ 3 .50 . If she had 3 more nickels than dimes. Find the amount of each ticket . **Seq2seq**: A total of 1.95 seats is in the ratio of 10%. If the total area of the coins is 1.95. Find **SeqGAN**: A carpet is 3 times as many more than the other. The total value is 3. **DeepGCN**: Dan invested $ 1.95 . part a t 10% annual interest. the total value of the cost of a 10% salt solution is added to **Trans**: A computer has a mixture of $ 3 . 000 and a mixture contains 5% profit of 5% . How much does each have.
Equation: $equ : 0.1 * x + 0.05 * y = 1.95$ $equ : y = x + 3$ **Reference**: A car radiator contains 10 liters of 30% antifreeze solution. How many liters will have to be replaced with the pure antifreeze if the resulting solution is to be 50% antifreeze? **Topic Words**: acid, solution **CG**: acid $\xrightarrow{IsA}$ liquid $\xrightarrow{RelatedTo}$ solution	**Ours**: How many liters of a 30% acid solution must be added to 10 liters of 20% solution on the mixture to make l0 litters of a 50% solution. **Seq2seq**: A raditor contains 50% nitric acid. If we take the same place to be 10. What is the total number of water. **SeqGAN**: A boat travels 10 miles per hour. How much would a 20% acid solution to be worth in 100 account **DeepGCN**: A <UNK> radiator is $ 10 . 00 a t a certain number. and the rest a t the same time . **Trans**: A car traveled in a car point is going a t a constant speed. If the car going a car point in the car . about 0 mph . how many miles apart.

5 Conclusion

We propose a novel model and a dataset for generating MWP from equations. Our model can effectively encode different types of math tokens in equations and reduce the gap between abstract math tokens and generated natural language text. It is also very useful in tackling the topic drifting and commonsense violation problems. Experiments on our dataset show our model significantly outperforms baseline models.

References

1. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent Dirichlet allocation. J. Mach. Learn. Res. (2003)
2. Bowman, S.R., Vilnis, L., Vinyals, O., Dai, A.M., Jozefowicz, R., Bengio, S.: Generating sentences from a continuous space. In: SIGNLL. ACL (2016)
3. Chen, S., Wang, J., Feng, X., Jiang, F., Qin, B., Lin, C.Y.: Enhancing neural data-to-text generation models with external background knowledge. In: EMNLP-IJCNLP. Association for Computational Linguistics (2019)
4. Cui, L., Wu, Y., Liu, S., Zhang, Y., Zhou, M.: Mutual: a dataset for multi-turn dialogue reasoning (2020)
5. Dauphin, Y.N., Fan, A., Auli, M., Grangier, D.: Language modeling with gated convolutional networks. arXiv, Computation and Language (2016)
6. Gong, L., Crego, J., Senellart, J.: Enhanced transformer model for data-to-text generation. In: Proceedings of the 3rd Workshop on Neural Generation and Translation. Association for Computational Linguistics (2019)
7. Gyawali, B., Gardent, C.: Surface realisation from knowledge-bases. In: ACL. The Association for Computer Linguistics (2014)
8. Lin, C.: Rouge: a package for automatic evaluation of summaries, pp. 74–81 (2004)
9. Papineni, K., Roukos, S., Ward, T., Zhu, W.: Bleu: a method for automatic evaluation of machine translation. In: ACL. ACL (2002)
10. Puduppully, R., Dong, L., Lapata, M.: Data-to-text generation with content selection and planning. In: AAAI. AAAI Press (2019)
11. See, A., Liu, P.J., Manning, C.D.: Get to the point: summarization with pointer-generator networks. In: SIGNLL. Association for Computational Linguistics (2017)
12. Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., Bengio, Y.: Graph attention networks. arXiv, Machine Learning (2017)
13. Wiseman, S., Shieber, S.M., Rush, A.M.: Challenges in data-to-document generation. In: EMNLP. Association for Computational Linguistics (2017)
14. Zhao, C., Walker, M., Chaturvedi, S.: Bridging the structural gap between coding and decoding for data-to-text generation. In: ACL. Association for Computational Linguistics (2020)
15. Zhou, Q., Huang, D.: Towards generating math word problems from equations and topics. In: INLG. Association for Computational Linguistics (2019)
16. Huang, D., Shi, S., Lin, C.-Y., Yin, J., Ma, W.-Y.: How well do computers solve math word problems? Large-scale dataset construction and evaluation. In: ACL. Association for Computational Linguistics (2016)
17. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. arXiv, Learning (2014)
18. Kushman, N., Artzi, Y., Zettlemoyer, L., Barzilay, R.: Learning to automatically solve algebra word problems. In: ACL, pp. 271–281. Association for Computational Linguistics (2014)
19. Shi, S., Wang, Y., Lin, C.-Y., Liu, X., Rui, Y.: Automatically solving number word problems by semantic parsing and reasoning. In: EMNLP. Association for Computational Linguistics (2015)
20. Upadhyay, S., Chang, M.-W.: Annotating derivations: a new evaluation strategy and dataset for algebra word problems. In: EACL. Association for Computational Linguistics, April 2017