

Generating Math Word Problems from Equations with Topic Controlling and Commonsense Enforcement

Tianyang Cao^{1,2}, Shuang Zeng^{1,2}, Songge Zhao¹, Mairgup Mansur³ and Baobao Chang^{1,2*}

¹ Key Laboratory of Computational Linguistics, Peking University, MOE, China

² School of Software and Microelectronics, Peking University, China

³ Sogou Technology Inc.

ctymy@pku.edu.cn, zengs@pku.edu.cn, zhaosongge@pku.edu.cn, maerhufu@sogou-inc.com, chbb@pku.edu.cn

Abstract

Recent years have seen significant advancement in text generation tasks with the help of neural language models. However, there exists a challenging task: generating math problem text based on mathematical equations, which has made little progress so far. In this paper, we present a novel equation-to-problem text generation model. In our model, 1) we propose a flexible scheme to effectively encode math equations, we then enhance the equation encoder by a Variational Autoencoder (VAE) 2) given a math equation, we perform topic selection, followed by which a dynamic topic memory mechanism is introduced to restrict the topic distribution of the generator 3) to avoid commonsense violation in traditional generation model, we pretrain word embedding with background knowledge graph (KG), and we link decoded words to related words in KG, targeted at injecting background knowledge into our model. We evaluate our model through both automatic metrics and human evaluation, experiments demonstrate our model outperforms baseline and previous models in both accuracy and richness of generated problem text.

Introduction

Text generation has been broadly researched as an important task in nature language processing. It aims to generate a text span which is fluent, readable and faithful to the original input. Current work in text generation is usually implemented with an end-to-end neural network, like sequence to sequence model (Bahdanau, Cho, and Bengio 2014), transformer architecture (Vaswani et al. 2017). Previous generation studies have mainly focused on the following tasks: 1) data-to-text generation (Bamman and Smith 2014; Gyawali and Gardent 2014; Wiseman, Shieber, and Rush 2017; Puduppully, Dong, and Lapata 2019a), 2) topic-to-text generation (Feng et al. 2018; Yang et al. 2019; Li et al. 2019), 3) dialogue response generation, including single-turn dialogue and multi-turn dialogue (Cai et al. 2018; Wu et al. 2018; Pan et al. 2019), 4) and some other generation-related tasks such as story endings generation (Luo et al. 2019; Guan et al. 2020).

Instead of these types of generation tasks being studied, in this paper we focus on a relatively new type of language generation task: generating math word problems from equations, which does not seem to have been fully studied by

the community. Successful math word problem generation has potential to automate the writing of math word problem and thus has practical value in computer aided instruction. Examples of math word problems are shown in Table 1, we aim to generate math problem texts corresponding to given equations. Different from traditional language task, generating math word problem however poses at least three special challenges:

(1) Encoding math equations is different from encoding general texts. Math equations are comprised of numbers, variables and operations, which requires a different method to encode math tokens. Sometimes we hope different numbers share the same embedding, other times we hope the number embedding have the ability to represent its special information.

(2) Lacking coherence is one of main problems that text generation tasks usually suffer from (Cui et al. 2020). However, this problem is even worse in the task of generating math word problems, since it covers a broad variety of topics and suffers from the problem of topic drifting. Table 2 shows three bad cases generated by a seq2seq baseline model and the first case reveals the problem of topic drifting where the first generated sentence is about price of goods and the second sentence changes to substance mixture.

(3) Even if the generated math word problem deals with the same topic, commonsense violation still occurs, as is presented in the second and third question in Table 2, to be specific, we can't say "hypotenuse of an square" or "dimension of a number", because it doesn't conform to commonsense.

To tackle these challenges, we propose a novel model for generating math word problems from equations. In our proposed model, 1) we implement an equation encoder which enables our model to comprehensively encode number-aware and number-unaware information, 2) we enhance the equation encoder by a Variational Autoencoder to obtain problem text aware representation, followed by which we perform topic selection and topic controlling to avoid topic drifting. 3) we design a pretraining step that encourages the model to learn commonsense relations contained in concept network, besides, we equip our generator with a commonsense enhancement module, which is helpful in alleviating commonsense violation.

*Corresponding author.

Equation 1: $equ : 100 + 20 = x \text{ } equ : x/3 = y$ Problem 1: The teacher said if you multiply my age by 3, then subtract 20, the result is 100. How old is the teacher?
Equation 2: $equ : 1 - 1/4 - 2/5 = x$ Problem 2: If 1 / 4 of a pie is eaten and later 2 / 5, what is the remaining fraction ?

Table 1: Examples sampled from the 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 an square is 7.9 meters long,...
Bad Case 3: what is the dimensions of the original number,...

Table 2: Bad cases generated by baseline model

The contributions of this paper can be summarized as follows:

- We propose an equation encoding approach which integrates both number value aware and unaware information, and together with the enhancement of an autoencoding mechanism, flexible and text-sensitive equation representation can be achieved.
- We propose a topic memory mechanism, with which topic information could be maintained and be used to adjust the topic of the generated problem text and thus to maintain the topic consistency.
- We propose to enrich the word representation by aggregating commonsense embedding of words, by which influence on the decoding process could be exerted to alleviate commonsense violation.

In order to verify the effectiveness of our proposals, we construct a dataset by adapting math problem questions and answering records from Yahoo! *. Experiments on this dataset show our model is effective, it outperforms both the baseline model and the previous models. Particularly, coherence in term of topic and commonsense consistency is accordingly improved.

Related Work

Math Word Problem (MWP) solving is one of related tasks to our work, which aims to extract a math equation from the math problem text or automatically solve the problem. MWP solving can be treated as a reverse task of our work. There are currently two research lines for MWP, one is to directly generate the equation, e.g., Liu et al. (2019) models the equation as an abstract syntax tree (ABS) and generates the expression tree from top to down. Wang, Liu, and Shi (2017) first introduces seq2seq in MWP solving, based on

*Our code and dataset will be released upon the acceptance of the paper

which Chiang and Chen (2019) introduces copy mechanism to improve generation quality. Another is learning the structure by statistic methods. Zou and Lu (2019) constructs a template of math-text trees and utilizes conditional random fields to learn the joint representation of the problem text and its expression tree. Roy and Roth (2017) designs separate models for different equation tasks.

Our work is also related to work on data-to-text generation, which transforms non-linguistic input into texts. Thanks to the wide application of neural language model and the availability of large-scale datasets, deep-learning approaches have brought great promise in data-to-text generation tasks, e.g., Wiseman, Shieber, and Rush (2017); Bosse-lut et al. (2017); Puduppully, Dong, and Lapata (2019a,b); Gong et al. (2019) intend to produce sport reports on ROTOWIRE dataset, and Chisholm, Radford, and Hachey (2017); Lebre, Grangier, and Auli (2016) focus on biography generation on WIKIBIO dataset. (Wiseman, Shieber, and Rush 2017). Recently, explicit content selection and text planning have been introduced to determine which to say and how to say (Puduppully, Dong, and Lapata 2019a; Perezbeltrachini and Lapata 2018). To better capture entity information appearing in structured data, entity memory has also been successfully applied in this task (Puduppully, Dong, and Lapata 2019b).

Most related to our work is Zhou and Huang (2019), formatted as generating problem text conditioned on equation templates and keywords, where templates are induced by replacing numbers in equations with a special token and keywords are directly extracted in the golden problem text via TF-IDF. Their model is based on a sequence-to-sequence model and integrates features of templates and keywords in the decoding phase. Their model, however, requires keywords from the golden answer as input when testing, which is inappropriate in real scenarios.

Problem Formulation

The input to our model consists of a sequence of tokens representing equations, we denote them as $\mathbf{x} = x_1x_2...x_n$, where n is the equation length measured by token numbers. Note that \mathbf{x} is the concatenation of all equations if the input contains several equations, and a delimiter *equ* is used to separate different equations. Each math token belongs to one of the following three types: math operation (e.g., +, -, *, ...), number (e.g., 0.2, 1, 5, ...), variable (e.g., x, y, z, ...). Output of our model is the problem text: $\mathbf{y} = y_1y_2...y_L$, whose solution should be identical to that of input equations. L is the problem text length. Our model aims to estimate the following conditional probability:

$$P(\mathbf{y}|\mathbf{x}) = \prod_{t=1}^L P(y_t|\mathbf{y}_{<t}, \mathbf{x}) \quad (1)$$

Model

Our model is shown in Figure 1. we start with a variational encoder-decoder model that consists of an equation encoder

and a problem decoder as illustrated in Figure 1. The reasons we use the variational architecture are two folds. Firstly, variational architecture introduces continuity in encoding space. Secondly, it allows us to further introduce an extra variational autoencoder to enhance the interaction between equations and problem texts, and text-sensitive representation could be produced. We achieve this by a KL divergence regularization term to make the two distributions of encoding variables (denoted by z and z') close enough.

To tackle the problem of topic drifting, we introduce a topic selector and a topic memory component. the topic selector chooses specific topic based on the latent representation and topic memory information of the chosen topic is updated and used to control the decoding process to favor the topic-consistent text. To alleviate the commonsense violation, we introduce a pretraining step to produce commonsense embeddings for words, and based on which commonsense information of a decoded word could be aggregated and used to influence the following choice at each step of decoding.

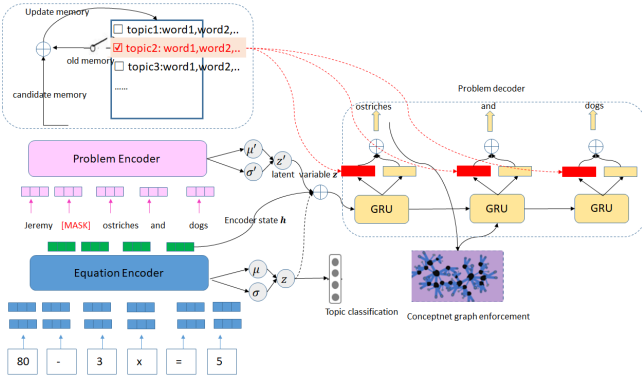


Figure 1: The overview of our proposed model. We omit the pretraining step for simplicity. In our variational autoencoder enhanced model, the problem encoder serves as prior network and the equation encoder serves as posterior network. Topic type predicted by the hidden equation representation z is used to select corresponding row in topic memory.

The Variational Encoder-Decoder Model

As we mentioned before, we choose the variational encoder-decoder model as the basic model to generate the MWP text from equations.

Equation Encoder: The input to our model is a sequence of math tokens $x_1 x_2 \dots x_n$, our input encoder encodes each token to one fixed hidden vector. As mentioned earlier, math equations encoding is different from encoding other natural languages, we should distinguish numbers, variables and operations to assign them different encoding. Sometimes the concrete number matters, while other times it doesn't matter.

We then consider the encoding scheme in Figure 2, it exploits Bidirectional Gated Recurrent Unit (BiGRU) as the basic module. BiGRU consumes token embedding of the equation sequences and the hidden states are computed by: $\overleftarrow{h}_i = GRU(emb(x_i), \overleftarrow{h}_{i-1})$, $\overrightarrow{h}_i =$

$GRU(emb(x_i), \overrightarrow{h}_{i+1})$. Combining forward and backward state yields $h_i = \overleftarrow{h}_i + \overrightarrow{h}_i$.

Intuitively, we expect the encoder to learn the balance between number aware representation and number unaware representation. Therefore, we firstly replace all numbers in the equation to a fixed mask [M] to obtain a number unaware token sequence. We separately feed the original sequence and the number unaware sequence into two different GRUs, then encoded hidden states are denoted as $\{h_{a,1}, h_{a,2}, \dots, h_{a,n}\}$ and $\{h_{b,1}, h_{b,2}, \dots, h_{b,n}\}$, respectively. Here we utilize Gated Linear Unit (GLU) (Dauphin et al. 2016) to compute final encoded hidden state:

$$h_k = MLP(h_{a,k}) \otimes \sigma(MLP(h_{b,k})) \quad 1 \leq k \leq n \quad (2)$$

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

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

z can then be sampled by using reparameterization trick: $z = \mu + r \odot \sigma$, where 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 $s_0 = MLP([h_n; z; h_n \otimes z])$. We denote the hidden state of the decoder at t th step as s_t and context vector obtained by attentions over the input equation as c_t . Assume the decoder generates word w_{t-1} in step $t-1$, decoding process can be formulated by :

$$s'_t = f(s_t) \quad s_t = GRU(s_{t-1}, g(e_{w_{t-1}})) \quad (4)$$

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

where $W^{vs} \in \mathbb{R}^{d \times d}$, $W^o \in \mathbb{R}^{d \times |V|}$, $|V|$, \hat{p} , 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.

In addition, most numbers appearing in math word problems are directly copied from source equations, we further adopt copy mechanism (See, Liu, and Manning 2017) to copy numbers from equations.

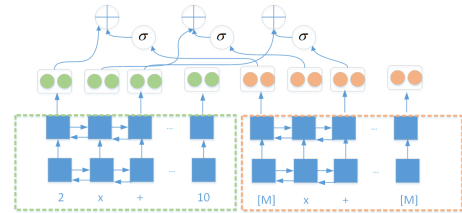


Figure 2: Framework of Equation Encoder

Enhancing Equation Encoder by Variational Autoencoder

Hidden equation representation z calculated by (3) fails to capture interaction between equations and problem text. We thus introduce a VAE to furtherly restrict z into similar vector space of given problem 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 problem text and works as a prior network. It takes the corrupted version of problem text y as input to guarantee robustness when testing, i.e., we randomly mask and delete some words in the original problem text. We implement the problem encoder module based on convolutional neural network (CNN) and F different convolutional kernels can be used to extract hierarchical features. Specifically, we define the k th convolutional kernel as $W^k \in \mathbb{R}^{d_k}$, performing convolution operation on problem text sequence $[y_1, y_2, \dots, y_L]$ yields:

$$h_k^q = \text{MaxPool}(f_{\text{conv}}([y_i; y_{i+1}; \dots; y_{i+l_k-1}])) \quad (6)$$

Combining results of different kernels, output of the problem encoder is calculated by:

$$q = \tanh(W^q [h_1^q; h_2^q; \dots; h_F^q]) \quad (7)$$

where parameter matrix $W^q \in \mathbb{R}^{F \times d}$.

Similar to (3), we perform linear transformation to q and obtain mean and variance of z 's prior distribution:

$$[\mu', \sigma'] = \text{MLP}(q) \quad z' | y \sim \mathcal{N}(\mu', \sigma'^2 \mathbf{I}) \quad (8)$$

We denote the problem decoder parameterized by θ_D as $p_D(y|x, z, \hat{p}; \theta_D)$, during training, 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:

$$\begin{aligned} \mathcal{L}_{VAE} = & -KL(p(z|y)||p(z|x)) \\ & + \mathbb{E}_{z \sim \mathcal{N}(\mu', \sigma'^2 \mathbf{I})} p_D(y|x, z, \hat{p}; \theta_D) \end{aligned} \quad (9)$$

Besides, we use KL cost annealing to avoid KL-vanishing phenomenon (Bowman et al. 2016). During inference, z is approximated by posterior network.

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 and then incorporate related topic words under this type into the problem decoder.

Topic Selection: To leverage topic background to the hidden equation representation z , we apply an unsupervised document topic model—Latent Dirichlet Allocation (LDA) (Blei, Ng, and Jordan 2003) 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 z :

$$\hat{p} = \arg \max \text{softmax}(W_z z + b_z) \quad (10)$$

Topic Controlling: Topic controlling renders our generator to interact with topic word distribution. With the help of LDA, a topic memory $C \in \mathbb{R}^{|P| \times K \times d}$ is constructed for storing pretrained embedding of topic keywords, where $|P|$ is the total topic number. K means each row of C contains information of top- K words of one topic and d is the vector dimension. With most probably topic type \hat{p} predicted in (10), then the concatenation of s_t and c_t is used as a query to the \hat{p} th row of topic memory and update s_t with the weighted sum of topic embedding in C :

$$\text{score}(t, j) = \frac{\exp([s_t; c_t] W^t C_{\hat{p}, j})}{\sum_{j=1}^K \exp([s_t; c_t] W^t C_{\hat{p}, j})} \quad 1 \leq j \leq K \quad (11)$$

and $f(s_t)$ in (5) is realized by:

$$f(s_t) = s_t + V \sum_{j=1}^K \text{score}(t, j) C_{\hat{p}, j} \quad (12)$$

where $W^t \in \mathbb{R}^{2d \times d}$ and $V \in \mathbb{R}^{d \times d}$ serves for linear projection. Furthermore, memory contexts are initialized by the pretrained word representation, but during the generating process, it should be dynamically updated with the produced sequence to keep recording new information, thus the topic memory can provide better guidance for 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 s'_t and $C_{\hat{p}, j}$:

$$u = \sigma(W^u [s'_t; C_{\hat{p}, j}]) \quad (13)$$

$$\tilde{C}_{\hat{p}, j} = \tanh(W^c [s'_t; C_{\hat{p}, j}]) \quad (14)$$

$$C_{\hat{p}, j} = u \otimes \tilde{C}_{\hat{p}, j} + (1 - u) \otimes C_{\hat{p}, j} \quad (15)$$

where $W^u, W^c \in \mathbb{R}^{d \times d}$.

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: In this part, we directly enrich information of our generator by pretraining word-level representation in an external commonsense KB. Note that word embedding pretraining is an off line step and is based on Graph Attention Network(GAT)(Velićković et al. 2017).

The commonsense KB contains multiple relationship in daily life, for example, a pair of related concepts (*average, Distinct From, large*) reveals "average" is distinct from

”large”, (*table, HasProperty, height*) means a table has the property ”height”. We can express the commonsense KB as a list of triples: $\{h_i, r_i, t_i\}_{i=1}^N$, where h_i, r_i, t_i represent head entity, relationship and tail entity, respectively. We denote the initial embedding of the i th node in KB as e_i and the initial embedding of the edge between node i and j as r_{ij} . For node i , its neighbour node set is defined as $\mathcal{N}(i)$, we then perform graph self attention (Velickovic et al. 2017) to capture the local information of each node’s neighbour nodes and assign a distribution over $\mathcal{N}(i)$:

$$\begin{aligned} e'_i &= W_e e_i \quad r'_{ij} = W_r r_{ij} \\ c_{ij} &= \text{LeakyReLU}(w_a^T [e_i; e_j; r'_{ij}]) \\ c'_i &= \text{softmax}(c_i) \\ e_i^* &= W^* \sum_{j \in \mathcal{N}(i)} c'_{ij} e'_j + b^* \end{aligned} \quad (16)$$

where W_e, W_r, w_a, W^*, b are all trainable parameters and $[a; b]$ denotes matrix concatenation operation.

To optimize node representation, we then follow TransE (Bordes et al. 2013), where the relationship between two entities can be seen as translation from the head entity to the tail entity, i.e., $\|e_i^* + e_j^* - r_{ij}\|$ can be minimized if and only if there exists relationship r_{ij} between node i and j . Therefore, we utilize margin loss to optimize our pretraining model in knowledge graph G :

$$L_p = \sum_{(h,r,t) \in G} \sum_{(\bar{h},\bar{r},\bar{t}) \notin G} \max(0, \lambda + F(h, r, t) - F(\bar{h}, \bar{r}, \bar{t})) \quad (17)$$

where $F(h, r, t) \triangleq \|e_h + r_{ht} - e_t\|_F^2$. $(\bar{h}, \bar{r}, \bar{t})$ is a negative triple and can be produced by randomly replacing head entity, relationship or tail entity in the positive triple. λ is a margin.

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 can then find adjacent nodes of w_{t-1} through Breadth First Search (BFS). For simplicity, we only consider first order neighbour nodes and second order neighbour nodes, as is shown in Figure 3. Let e_{ij} denote the path representation from node i to node j if i and j are directly connected:

$$e_{ij} = \phi(W^g [e_i; e_j]) \quad (18)$$

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

$$e_{ij} = \alpha \phi(W^g [e_i; e_j]) + (1 - \alpha) \sigma(e_{ik} \otimes (U e_{kj})) \quad (19)$$

where $U \in \mathbb{R}^{d \times d}$, $\phi(\cdot)$ is a nonlinear function, in this paper we use $\tanh(\cdot)$. $\sigma(\cdot)$ is Sigmoid function. $\alpha \in [0, 1]$ is

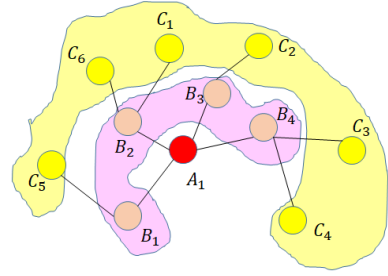


Figure 3: Illustration of search adjacent nodes. Nodes with pink background are first order neighbours of A_1 , which are directly connected to A_1 . Nodes with yellow background are second order neighbours, which connect to A_1 via an intermediate node.

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 $e_{w_{t-1},j}$ when j goes through $\mathcal{N}_1(w_{t-1}) \cup \mathcal{N}_2(w_{t-1})$:

$$\beta_{t-1,j} \propto \exp(e_{w_{t-1}} W^b e_{w_{t-1},j}) \quad (20)$$

$$g_{t-1} = \sum_{j \in \mathcal{N}_1(w_{t-1}) \cup \mathcal{N}_2(w_{t-1})} \beta_{t-1,j} e_{w_{t-1},j} \quad (21)$$

Followed by (21), to better reflect the effect of concept knowledge to word choice, we combine $e_{w_{t-1}}$ with g_{t-1} to realize $g(e_{w_{t-1}})$ in (5):

$$g(e_{w_{t-1}}) = GRU(e_{w_{t-1}}, H g_{t-1}) \quad (22)$$

Training Objective

We aggregate 1): VAE loss mentioned in (9) 2) auxiliary topic prediction loss $\mathcal{L}_{topic} = \mathbb{E}_{z \sim \mathcal{N}(\mu', \sigma'^2 \mathbf{I})} p(\hat{p}|z, \mathbf{x})$ to obtain total loss:

$$\mathcal{L}_{total} = \mathcal{L}_{VAE} + \mu \mathcal{L}_{topic} \quad (23)$$

where μ is a hyperparameter.

Experiments

Datasets

We crawl and collect data from Yahoo, there are 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 pair whose problem text length is longer than 45 tokens, besides, we replace those words appearing less than 2 times to $\langle \text{UNK} \rangle$.

Model Settings

The batch size for training is 32. We employ ConceptNet5[†] to construct KB, it has 34 types of relationship in total. 2 layer graph attention network is implemented for word

[†]<https://github.com/commonsense/conceptnet5>

	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 3: Statistic of datasets

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 4. 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. Margin in (17) is set to $\lambda = 1$. μ in (23) is set to 0.5. Weight coefficient in (19) is set to $\alpha = 0.7$. We use Adam optimizer (Kingma and Ba 2014) to train our model, the learning rate is set to 0.0005.

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...

Table 4: Topic classes statistics and representative words sampled from each topic

Baselines

In this section we describe our baselines. Since math word problem generation has never been researched before and our model framework is more close to Seq2seq model, we adopt Seq2seq and its variants for comparison.

- **Seq2seq**(Bahdanau, Cho, and Bengio 2014): Seq2seq is first proposed for machine translation task. In this paper, we implement Seq2seq with attention mechanism and copy mechanism.
- **SeqGAN**(Yu et al. 2016): SeqGAN fuses the advantage of reinforcement learning (RL) and Generative Adversarial Network (GAN). It achieves improvements over strong baselines in both text generation and music generation task.
- **MAGNET**(Zhou and Huang 2019): MAGNET generates math word problems from equation templates and keywords. However, their model requires 10 keywords directly extracted from the golden problem text as an extra input, which is inappropriate in real scenarios; their

dataset and code are both unavailable. We thus implement MAGNET in our dataset. To make fair comparison, we also append 10 keywords obtained by the same way to the input sequence of our model when performing **separate** comparison with MAGNET.

Automatic Evaluation

We report automatic evaluation in five aspects: BLEU (up to bigrams) (Papineni et al. 2002), ROUGH-L (Lin 2004), 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 5. In Table 5 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) 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 MAGNET with our model including the same keywords as an extra input in Table 6, which demonstrates our model can achieve performance gain with the same input.

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
Our model	0.0433	0.2415	20.84	53.81	55.14
w/o Pt & 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: Automatic results in test dataset with BLEU, ROUGH-L(ROU), Dist-1(D1), Dist-2(D2) and Number Recall(NR). Pt, CE, TM and V denote pretraining, commonsense enhancement, topic memory and VAE, respectively.

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

Table 6: Comparison between our model with keywords(KW) and MAGNET in automatic results

Human Evaluation

Automatic metrics such as BLEU and ROUGH 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. We randomly select 50 generated sentences and score them in

three grades. The scores are projected to 1~3, where higher score implies better performance (for solvability we use percentage). We report the average scores in Table 7.

- **Fluency(Flu)**: Fluency mainly judges whether the problem text is fluent, i.e., whether the generated problem text has some grammar errors. Fluency also reflects whether there exists some expression conflict with our commonsense.
- **Coherence(Coh)**: Coherence weights if the problem text is consistent in text level.
- **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 equations and solve them according to the generated problem text.
- **Solvability-2(S2)**: Solvability-2 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.

Table 7 (*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 7 (*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.

	Flu	Coh	S1(%)	S2(%)
Our model	2.42	2.41	35	55
Seq2seq	2.27	2.09	23	34
SeqGAN	2.25	1.97	20	40
MAGNET	2.40	2.60	44	76
Our model(KW)	2.56	2.76	56	74

Table 7: Human evaluation results: comparison between the proposed model and baseline models.

Case Study

Table 8 shows some math word problems generated by different models. It’s easy to show problem text generated by Seq2seq suffers from lack of coherence, for example, in case2 and case3, the baseline result talks about different topics in the same sentence. As a comparison, our generator talks about 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, problem text generated by our model is more in line with commonsense, such as "the hypotenuse of a right triangle", which accords to our design goal.

case 1	Equation	$4 * (x - y) = 800$ <i>equ</i> : $2 * (x + y) = 800$
	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 .
	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.
case 2	SeqGAN	The perimeter of a rectangle is 800 inches. Find the length.
	Equation	<i>equ</i> : $0.1 * x + 0.05 * y = 1.95$ <i>equ</i> : $y = x + 3$
	Reference	Ken has \$ 1.95 in nickels and dimes . There are 3 more nickels than dimes . How many of each does he have ?
	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 .
case 3	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.
	Equation	<i>equ</i> : $0.3 * (10 - x) + 1.0 * x = 0.5 * 10$
	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 ?
case 3	Ours	How many liters of a 30% acid solution must be added to 10 liters of 20% solution on the mixture to make 10 liters 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.
case 3	SeqGAN	A boat travels 10 miles per hour. How much would a 20% acid solution to be worth in 100 account

Table 8: Three examples of math word problems generated by different models.

Conclusion

In this paper, we propose a problem text-guided math word problem generation model with topic controlling and commonsense enforcement. Experiments demonstrate our model can generate more fluent, coherence and logical problems. However, as a completely new task, math word problem generation still leaves much space for improvement. In the future, we tend to explore how to induce math word problem templates from training dataset, based on which we can generate math word problems with higher quality.

References

- Bahdanau, D.; Cho, K.; and Bengio, Y. 2014. Neural Machine Translation by Jointly Learning to Align and Translate. *arXiv: Computation and Language* .
- Bamman, D.; and Smith, N. A. 2014. Unsupervised

- discovery of biographical structure from text. *Transactions of the Association for Computational Linguistics* 2: 363–376. URL https://www.mitpressjournals.org/doi/pdf/10.1162/tacL_a.00189.
- Blei, D. M.; Ng, A. Y.; and Jordan, M. I. 2003. Latent dirichlet allocation. *Journal of Machine Learning Research* 3: 993–1022.
- Bordes, A.; Usunier, N.; Garciaduran, A.; Weston, J.; and Yakhnenko, O. 2013. Translating Embeddings for Modeling Multi-relational Data 2787–2795.
- Bosselut, A.; Levy, O.; Holtzman, A.; Ennis, C.; Fox, D.; and Choi, Y. 2017. Simulating Action Dynamics with Neural Process Networks. *arXiv: Computation and Language*.
- Bowman, S. R.; Vilnis, L.; Vinyals, O.; Dai, A. M.; Jozefowicz, R.; and Bengio, S. 2016. Generating Sentences from a Continuous Space 10–21.
- Cai, D.; Wang, Y.; Bi, W.; Tu, Z.; Liu, X.; Lam, W.; and Shi, S. 2018. Skeleton-to-Response: Dialogue Generation Guided by Retrieval Memory. *arXiv: Computation and Language*.
- Chiang, T.; and Chen, Y. 2019. Semantically-Aligned Equation Generation for Solving and Reasoning Math Word Problems 2656–2668.
- Chisholm, A.; Radford, W.; and Hachey, B. 2017. Learning to generate one-sentence biographies from Wikidata 1: 633–642.
- Cui, L.; Wu, Y.; Liu, S.; Zhang, Y.; and Zhou, M. 2020. MuTual: A Dataset for Multi-Turn Dialogue Reasoning.
- Dauphin, Y. N.; Fan, A.; Auli, M.; and Grangier, D. 2016. Language Modeling with Gated Convolutional Networks. *arXiv: Computation and Language*.
- Feng, X.; Liu, M.; Liu, J.; Qin, B.; Sun, Y.; and Liu, T. 2018. Topic-to-Essay Generation with Neural Networks 4078–4084.
- Gong, H.; Feng, X.; Qin, B.; and Liu, T. 2019. Table-to-Text Generation with Effective Hierarchical Encoder on Three Dimensions (Row, Column and Time). *arXiv: Computation and Language*.
- Guan, J.; Huang, F.; Zhao, Z.; Zhu, X.; and Huang, M. 2020. A Knowledge-Enhanced Pretraining Model for Commonsense Story Generation. *arXiv: Computation and Language*.
- Gyawali, B.; and Gardent, C. 2014. Surface Realisation from Knowledge-Bases 424–434.
- Kingma, D. P.; and Ba, J. 2014. Adam: A Method for Stochastic Optimization. *arXiv: Learning*.
- Lebret, R.; Grangier, D.; and Auli, M. 2016. Neural Text Generation from Structured Data with Application to the Biography Domain 1203–1213.
- Li, J.; Zhao, W. X.; Wen, J.; and Song, Y. 2019. Generating Long and Informative Reviews with Aspect-Aware Coarse-to-Fine Decoding 1969–1979.
- Lin, C. 2004. ROUGE: A Package for Automatic Evaluation of Summaries 74–81.
- Liu, Q.; Guan, W.; Li, S.; and Kawahara, D. 2019. Tree-structured Decoding for Solving Math Word Problems 2370–2379.
- Luo, F.; Dai, D.; Yang, P.; Liu, T.; Chang, B.; Sui, Z.; and Sun, X. 2019. Learning to Control the Fine-grained Sentiment for Story Ending Generation 6020–6026.
- Pan, B.; Li, H.; Yao, Z.; Cai, D.; and Sun, H. 2019. Reinforced Dynamic Reasoning for Conversational Question Generation 2114–2124.
- Papineni, K.; Roukos, S.; Ward, T.; and Zhu, W. 2002. Bleu: a Method for Automatic Evaluation of Machine Translation 311–318.
- Perebeltrachini, L.; and Lapata, M. 2018. BOOTSTRAPPING GENERATORS FROM NOISY DATA 1: 1516–1527.
- Puduppully, R.; Dong, L.; and Lapata, M. 2019a. Data-to-Text Generation with Content Selection and Planning 33(01): 6908–6915.
- Puduppully, R.; Dong, L.; and Lapata, M. 2019b. Data-to-text Generation with Entity Modeling 2023–2035.
- Roy, S.; and Roth, D. 2017. Mapping to Declarative Knowledge for Word Problem Solving. *arXiv: Computation and Language*.
- See, A.; Liu, P. J.; and Manning, C. D. 2017. Get To The Point: Summarization with Pointer-Generator Networks 1: 1073–1083.
- Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; and Polosukhin, I. 2017. Attention is All you Need 5998–6008.
- Velickovic, P.; Cucurull, G.; Casanova, A.; Romero, A.; Lio, P.; and Bengio, Y. 2017. Graph Attention Networks. *arXiv: Machine Learning*.
- Wang, Y.; Liu, X.; and Shi, S. 2017. Deep Neural Solver for Math Word Problems 845–854.
- Wiseman, S.; Shieber, S. M.; and Rush, A. M. 2017. Challenges in Data-to-Document Generation 2253–2263.
- Wu, Y.; Wei, F.; Huang, S.; Wang, Y.; Li, Z.; and Zhou, M. 2018. Response Generation by Context-aware Prototype Editing. *arXiv: Computation and Language*.
- Yang, P.; Li, L.; Luo, F.; Liu, T.; and Sun, X. 2019. Enhancing Topic-to-Essay Generation with External Commonsense Knowledge. 2002–2012. doi:10.18653/v1/P19-1193.
- Yu, L.; Zhang, W.; Wang, J.; and Yu, Y. 2016. SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient. *arXiv: Learning*.
- Zhou, Q.; and Huang, D. 2019. Towards Generating Math Word Problems from Equations and Topics. 494–503.
- Zou, Y.; and Lu, W. 2019. Text2Math: End-to-end Parsing Text into Math Expressions 5326–5336.