Mathematical Word Problem Generation from Commonsense Knowledge Graph and Equations

Tianqiao Liu, Qiang Fang, Wenbiao Ding, Hang Li, Zhongqin Wu, Zitao Liu[*]

TAL Education Group, Beijing, China

{liutianqiao,fangqiang,dingwenbiao,lihang4}@tal.com
{wuzhongqin,liuzitao}@tal.com

Abstract

There is an increasing interest in the use of mathematical word problem (MWP) generation in educational assessment. Different from standard natural question generation, MWP generation needs to maintain the underlying mathematical operations between quantities and variables, while at the same time ensuring the relevance between the output and the given topic. To address above problem, we develop an end-to-end neural model to generate diverse MWPs in real-world scenarios from commonsense knowledge graph and equations. The proposed model (1) learns both representations from edge-enhanced Levi graphs of symbolic equations and commonsense knowledge; (2) automatically fuses equation and commonsense knowledge information via a self-planning module when generating the MWPs. Experiments on an educational gold-standard set and a large-scale generated MWP set show that our approach is superior on the MWP generation task, and it outperforms the SOTA models in terms of both automatic evaluation metrics, i.e., BLEU-4, ROUGE-L, Self-BLEU, and human evaluation metrics, i.e., equation relevance, topic relevance, and language coherence. To encourage reproducible results, we make our code and MWP dataset public available at https://github.com/tal-ai/MaKE_EMNLP2021.

1 Introduction

A mathematical word problem (MWP) is a coherent narrative that provides clues to the underlying correct mathematical equations and operations between variables and numerical quantities (Cetintas et al., 2010; Moyer et al., 1984). MWPs challenge a student from a wide range of skills such as literacy skills for understanding the question, analytical skills for recognizing the problem type and applying arithmetical operators (Rembert et al., 2019; Moon-Rembert and Gilbert, 2019). Table 1 shows

one such problem[1] where students are asked to infer the counts of chickens and rabbits.

Math Word Problem	Chickens and rabbits were in the yard. Together they had 27 heads and 86 legs. How many rabbits were in the yard?		
Equations	x+y=27 2x+4y=86	Solutions	x=11 y=16

Table 1: An illustrative example of an MWP.

In this paper, our objective is to automatically generate well-formed MWPs. Such automation will not only reduce the teachers' burden of manually designing MWPs, but provide students with a sufficiently large number of practice exercises, which help students avoid rote memorization (Williams, 2011; Wang and Su, 2016).

A large spectrum of models have been developed and successfully applied in a broad area of natural question generation (NQG) (Pan et al., 2019; Li et al., 2018; Liu et al., 2020; Sun et al., 2018; Zhang and Bansal, 2019; Kurdi et al., 2020; Guan et al., 2021a,b) and there has been a recent movement from the NQG community towards automatic generation of MWPs (Koncel-Kedziorski et al., 2016a; Polozov et al., 2015; Zhou and Huang, 2019). For example, Koncel-Kedziorski et al. (2016a) proposed a two-stage rewriting approach to edit existing human-authored MWPs. Polozov et al. (2015) conducted the MWP generation as a constrained synthesis of labeled logical graphs that represent abstract plots.

In general, there exists a large number of NQG models representing various text data and their syntax and semantics (Pan et al., 2019). However, automatic generation of MWPs still presents numerous challenges that come from special characteristics of real-world educational scenarios as follows: (1) MWP generation models need to not only generate fluent sentences but understand the mathematical variables, numerical quantities, op-

---

[*] The corresponding author: Zitao Liu

[1]One MWP example from https://www.hackmath.net/en/math-problem/56

erations, and their relations. Moreover, the models are supposed to be able to generalize to unseen equations. (2) Multiple studies have found that MWPs with real-life plots help conceptual knowledge understanding, discourse comprehension and children engagement (Carpenter et al., 1980; Rembert et al., 2019). (3) Computerized educational assessment systems require diverse MWP results even given similar input equations, which helps prevent students from rote memorization (Deane and Sheehan, 2003).

To overcome the above challenges, in this paper, we present a novel neural generation model **MaKE** (short for **Ma**thematical word problem generation from commonsense **K**nowledge and **E**quations), which aims to automatically generate coherent and diverse MWPs from given equations in students' real-life scenarios. More specifically, to fully understand the mathematical variables, numerical quantities, operations, and their relations, equations are transformed into an edge-enhanced Levi graph. We adopt the gated graph neural networks (GGNNs) to learn representative embeddings from the equation based symbolic Levi graph. Meanwhile, the same procedure is applied to the external commonsense based knowledge graph (CSKG), which helps generate topic-relevant and semantically valid sentences in real-life settings. We choose to use the conditional variational autoencoder (VAE) framework to generate MWPs from diversity promoting latent states. Furthermore, in the decoding stage, we develop a self-planning module to dynamically select and fuse information from both equations and commonsense knowledge, which improves syntax structure of generated MWP sentences. Overall this paper makes the following contributions:

- We propose a GGNN based conditional VAE model for MWP generation. To the best of our knowledge, we are the first to introduce the combinational architecture of GGNN and condition VAE for MWP generation.

- We design a novel self-planning decoding module to wisely fuse information from equations and commonsense knowledge with implicit schedule, which helps generate semantically valid MWPs.

- The proposed model achieves the SOTA scores and outperforms existing methods by a significant margin on real-world educational MWP datasets from both automatic machinery and human evaluation metrics.

2 Related Work

2.1 Natural Question Generation

Previous research has directly approached the task of automatically generating questions for many useful applications such as augmenting data for the QA tasks (Li et al., 2018; Sun et al., 2018; Zhang and Bansal, 2019), helping semantic parsing (Guo et al., 2018) and machine reading comprehension (Yu et al., 2020; Yuan et al., 2017), improving conversation quality (Mostafazadeh et al., 2016; Dong et al., 2019), and providing student exercises for education purposes (Koncel-Kedziorski et al., 2016a).

Various NQG methods are developed which can be divided into two categories: heuristic based approaches and neural network based approaches (Pan et al., 2019; Kurdi et al., 2020). The former generates questions in two stages: it first obtains intermediate symbolic representations and then constructs the natural language questions by either re-arranging the surface form of the input sentence or generating with pre-defined question templates. The latter neural approaches view the NQG task as a sequence-to-sequence (seq2seq) learning problem and jointly learn generation process in an end-to-end manner (Yao et al., 2018; Zhou et al., 2018).

2.2 Math Word Problem Generation

Different from standard NQG tasks, generating MWPs not only needs the syntax, semantics and coherence of the output narratives, but requires understandings of the underlying symbolic representations and the arithmetic relationship between quantities. In general, MWP generation approaches can be divided into three categories: (1) template based approaches; (2) rewriting based approaches; and (3) neural network based approaches.

Template based approaches usually fall into a similar two-stage process: they first generalize an existing problem into a template or a skeleton, and then generate the MWP sentences from the templates (Williams, 2011; Polozov et al., 2015; Bekele, 2020). Deane and Sheehan (2003) used semantic frames to capture both scene stereotypical expectations and semantic relationships among words and utilized a variant of second-order predicate logic to generate MWPs. Wang and Su (2016) leveraged the binary expression tree to represent the story of the MWP narrative and composed the natural language story recursively via a bottom-up tree traversal. Template based approaches heavily rely on the tedious and limited hand-crafted tem-

plates, leading to very similar generated results. This cannot meet the demand of a large number of high-quality and diverse MWPs.

Rewriting based approaches target the MWP generation problem by editing existing human-written MWP sentences to change their theme without changing the underlying story (Koncel-Kedziorski et al., 2016a; Moon-Rembert and Gilbert, 2019). For example, Koncel-Kedziorski et al. (2016a) proposed a rewriting algorithm to construct new texts by substituting thematically appropriate words and phrases. Rewriting based approaches are more flexible compared with templates based approaches. However, there are several drawbacks that prevent them from providing the large number of MWPs. First, the generation process is based on existing MWPs, which significantly limits the generation ability. Second, students easily fall into rote memorization since it is too trivial to notice that the underlying mathematical equations are still unchanged.

Recent attempts have been focused on exploiting neural network based approaches that generating MWPs from equations and topics in an end-to-end manner (Zhou and Huang, 2019; Liyanage and Ranathunga, 2020). Zhou and Huang (2019) designed a neural network with two encoders to fuse information of both equations and topics and dual-attention mechanism to generate relevant MWPs. Liyanage and Ranathunga (2020) tackled the generation problem by using the long short term memory network with enhanced input features, such as character embeddings, word embeddings and part-of-speech tag embeddings.

The closest work to our approach is Zhou and Huang (2019) and the main differences are as follows: (1) Zhou and Huang (2019) directly encode the equation by a single-layer bidirectional gated recurrent unit (GRU), while we first convert equations into Levi graph and conduct the encoding by the GGNN model; (2) instead of directly using the pre-trained embeddings of similar words given the topic, we choose to learn the topic relevant representations from an external CSKG; and (3) we choose to use the VAE framework to promoting more diverse results.

3 Learning from Commonsense Knowledge and Equations

Our objective is to automatically generate a significant number of diverse MWPs in students' real-life scenarios from valid equations. In addition, we support the personalized generation in which students (or teachers) can determine the story plots of MWPs by specifying topics and mapping relations between variables and entities (i.e., "x: chicken, y: rabbits", "x: apple, y: banana", etc.). A topic indicates a type of real-world scenarios, such as animals, fruits, etc.

As shown in Figure 1, we adopt the encoder-decoder generation framework. The input includes a set of equations and a knowledge graph with a specific topic. We construct Levi graphs (Levi, 1942) from symbolic equations and the CSKG respectively (See Section 3.1). After that, we employ GGNNs to extract the full graph structure information about equations and real-life story plots (See Section 3.2). Then, we generate target sentence by a conditional VAE with a self-planning module (See Section 3.3). The self-planning module enables the decoder to pay different portions of attention to the equations and the CSKG.

Please note that in this paper, we focus on generating MWPs with linear equations of two variables without any constraint. Our framework can be easily generalized into MWPs with different numbers of variables with little modification.

3.1 Levi Graph Construction

3.1.1 Equation Based Symbolic Graph

The equation based symbolic graph is designed to capture the relations among mathematical variables and numerical quantities, and build connections between mathematical variables and the corresponding commonsense knowledge. In this work, we consider the linear equations (with two variables) behind the MWPs as $ax + by = m; cx + dy = n$, where x and y are the variables and a, b, c, d, m, and n are positive integer quantities. More equation variants are discussed in Appendix A.1.

Equations are first converted to a symbolic graph as shown in Figure 2 (a). In the symbolic graph, edge labels, i.e., *Add to res*, *Mul*, etc. representing the mathematical relations play important roles in the MWP generation, where "*Add to res*" indicates addition operation to the result operand and "*Mul*" indicates multiply operation. In order to well capture such relations, we model the edge labels as explicit nodes. Following previous work in Beck et al. (2018), we transform the symbolic graph into its equivalent edge-enhanced Levi graph (Levi, 1942) by adding two nodes for each labeled edge. One node denotes the forward direction of
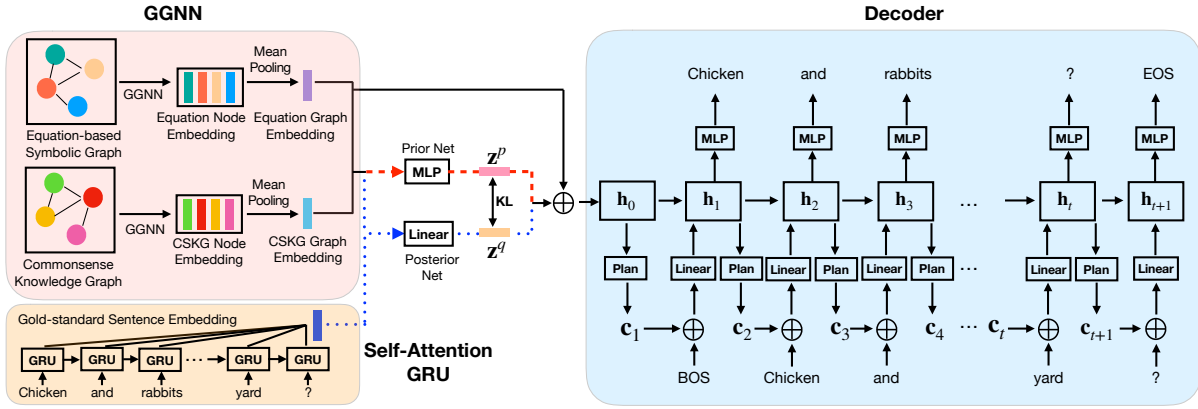
Figure 1: The overview of the proposed framework. The blue dot line (or red dash line) is only enabled in the training (or inference) stage. $\oplus$ denotes the vector concatenation. **Linear** represents the linear transformation and **Plan** denotes the self-planning module (discussed in Section 3.3).

the relation and one represents the reverse. By adding reverse nodes, we encourage more information flow from the reverse direction, in the same way, RNN-based encoders benefit from right-to-left propagation. Furthermore, we explicitly add self-loop edges to each node in the Levi graph. The symbolic Levi graph is depicted in Figure 2 (b). More details on Levi graph transformation can be found in Appendix A.2.

3.1.2 Commonsense Based Knowledge Graph

In order to generate valid questions in students' real-life scenarios, we utilize explicit knowledge from a self-derived CSKG specifically designed for MWP generation. We have to admit that our CSKG is of particularly tiny size compared to publicly available knowledge graphs like ConceptNet and Wikipedia. However, we have exclusive relationships that can be utilized for MWPs generations, i.e., (apple, has unit of measurement, pound), (banana, has price unit, yuan), (chicken, has feet number, 2), etc. These commonsense knowledge triples are extracted from MWP texts in a semi-automatic manner. Specifically, for each MWP, we first apply the part-of-speech tagger from Stanford CoreNLP[2] with some heuristic rules for automatic commonsense knowledge extraction. Furthermore, because the generation process requires high-quality commonsense information, we ask crowd workers to verify the extracted results, which includes both entities and the corresponding attributes. For example, for the MWP shown in Table 1, the auto-parsed entities are *Chickens* and *Rabbits* and the extracted relations are (1) a *belong_to* relation showing that the *Chickens* belong to *livestock*; and (2) a *has_head_entity* relation showing that the *Chick-*

ens* have head entity *head*. The similar relations about Rabbits are extracted as well. We form these triples into our commonsense knowledge graphs. Moreover, we explicitly select a few entities from the above extraction process as "topics" and these topic terms can be revised by the crowd workers if they are mis-extracted. The topic entities are obtained from a given K-12 educational vocabulary. Figure 2 (c) illustrates a sample of a CSKG with a topic of *Livestock*.

With the help of CSKG, students or teachers are able to set their own preferences when generating MWPs by choosing different topics, such as zoo, transportation, etc. This external commonsense knowledge provides additional background information that improves the generated results diversity. Moreover, the CSKG improves the generation quality by alleviating ill-informed wordings or sentences. For instance, in spite of no grammatical errors, it makes no sense to have "rabbits live in the ocean" or "apple has two feet". Similar to the Levi graph construction procedure in Section 3.1.1, we introduce additional nodes for relations in CSKG and add reverse and self-loop edges. The CSKG Levi graph is shown in Figure 2 (d).

3.2 Gated Graph Neural Encoding

Following the success of GGNN models (Beck et al., 2018; Ruiz et al., 2019), we use GGNNs to capture both the mathematical relations among variables and quantities and the real-life associations among entities in the MWPs. Specifically, let $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ be an edge-enhanced Levi graph where $\mathcal{V}$ and $\mathcal{E}$ are the sets of nodes and edges. Let $a^{v,u}$ be the similarity between node v and node u from its row-wise normalized adjacent matrix. Given an input Levi graph $\mathcal{G}$ that may represent

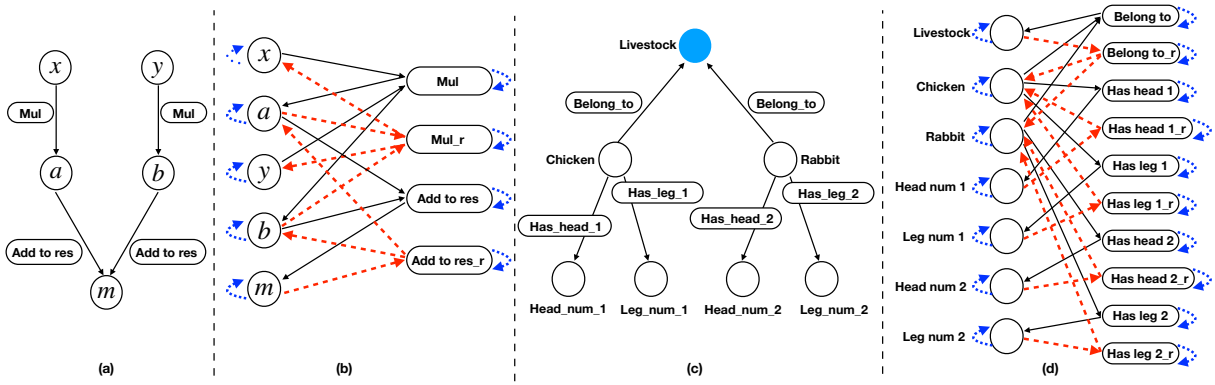---

[2] https://nlp.stanford.edu/software/tagger.shtml

4228

Figure 2: (a) a sample symbolic graph of equation "ax+by = m"; (b) the edge-enhanced Levi graph of the same equation; (c) an illustrative sample of the CSKG under topic *livestock*; (d) the corresponding edge-enhanced Levi graph of CSKG. The red dash arrows represent the reverse edges and the blue dot arrows represent the self-loop edges. The subscript "_r" denotes the artificially added reverse nodes. The blue node in Figure 2(c) denotes the given topic.

either the equations or the CSKG, the basic recurrence of the GGNN model is defined as follows:

$$\mathbf{g}_0^v = \mathbf{e}_0^v; \quad \boldsymbol{\gamma}_t^v = \sum_{u \in N_{(v)}} a^{v,u} \mathbf{g}_{t-1}^u;$$

$$\mathbf{z}_t^v = \sigma(\mathbf{W}^z \boldsymbol{\gamma}_t^v + \mathbf{U}^z \mathbf{g}_{t-1}^v);$$
$$\mathbf{r}_t^v = \sigma(\mathbf{W}^r \boldsymbol{\gamma}_t^v + \mathbf{U}^r \mathbf{g}_{t-1}^v);$$
$$\widetilde{\mathbf{g}_t^v} = \tanh(\mathbf{W}^h \boldsymbol{\gamma}_t^v + \mathbf{U}^h (\mathbf{r}_t^v \odot \mathbf{g}_{t-1}^v));$$
$$\mathbf{g}_t^v = (1 - \mathbf{z}_t^v) \odot \mathbf{g}_{t-1}^v + \mathbf{z}_t^v \odot \widetilde{\mathbf{g}_t^v}$$

where $\mathbf{e}_0^v$ denotes the initial embedding of node v. $N_{(v)}$ is the set of neighbor nodes for v and σ is the sigmoid function. $\odot$ is the component-wise multiplication function and $\mathbf{z}_t^v$ and $\mathbf{r}_t^v$ are gating vectors.

Let $\mathbf{G}_0 = [\mathbf{g}_0^1; \mathbf{g}_0^2; \cdots ; \mathbf{g}_0^{|\mathcal{V}|}]$ be the initial word embedding matrix of all the nodes and $\mathbf{G}_n$ be the matrix of representation of node embeddings from the above GGNN model after n iterations, i.e., $\mathbf{G}_n = [\mathbf{g}_n^1; \mathbf{g}_n^2; \cdots , \mathbf{g}_n^{|\mathcal{V}|}]$. Similar to He et al. (2016), we ease the downstream learning tasks with embedding augmentation. We apply a linear transformation on the concatenation of $\mathbf{G}_0$ and $\mathbf{G}_n$, i.e., $\mathbf{G}_* = \mathbf{W}_*[\mathbf{G}_0; \mathbf{G}_n]$. Such augmented node representations contain abstract context information, which are used in our language generator in Section 3.3. Let $\mathbf{G}_*^e$ and $\mathbf{G}_*^k$ be the augmented GGNN embeddings of the equations and the CSKG. Meanwhile we apply a mean pooling operation over $\mathbf{G}_*^e$ and $\mathbf{G}_*^k$ to get the graph-level equation representation ($\mathbf{g}_*^e$) and CSKG representation ($\mathbf{g}_*^k$).

3.3 Conditional VAE with Self-Planning

In this section, we introduce our VAE architecture with the self-planning module for the MWP generation. Our self-planning module makes dynamic fusion on the learned representations of equations and CSKG to generate the MWPs.

Let Y be the random variable representing the texts of MWPs and Z be the diversity promoting latent variable of the distribution of the MWPs. Let C be the random variable representing the conditions of both the explicit equations and the implicit CSKG learned from GGNNs. We model the MWP generation by the conditional distribution as follows: $p(Y|C) = \int p(Y|C, Z)p(Z|C)dZ$ where $p(Y|C, Z)$ is the MWP generator and $p(Z|C)$ is the prior net. Since the integration of Z is intractable, we apply variational inference and optimize the evidence lower bound as follows:

$$\log p(Y|C) \geq \mathbb{E}_{q(Z|C,Y)}\Big[\log p(Y|C, Z)\Big]$$
$$- D_{KL}\Big(q(Z|C,Y)||p(Z|C)\Big)$$

where $D_{KL}(\cdot||\cdot)$ denotes the KL-divergence.

Following conventions, we assume both the prior net and posterior net of Z following the isotropic Gaussian distributions, i.e., $p(Z|C) \sim \mathcal{N}(\boldsymbol{\mu}^p, \sigma^p \mathbf{I})$ and $q(Z|C, Y) \sim \mathcal{N}(\boldsymbol{\mu}^q, \sigma^q \mathbf{I})$. The prior net only encodes the given conditions of both the explicit equations and the implicit CSKG while the posterior net encodes both given conditions and the texts of MWPs. Both the prior net and the posterior net are built upon the GGNNs shown in Figure 1 as follows:

$$[\boldsymbol{\mu}^p; \log \boldsymbol{\sigma}^p] = \mathbf{MLP}([\mathbf{g}_*^e; \mathbf{g}_*^k]);$$
$$[\boldsymbol{\mu}^q; \log \boldsymbol{\sigma}^q] = \mathbf{W}^q([\mathbf{g}_*^e; \mathbf{g}_*^k; \mathbf{GRU}(\mathbf{y})]) + \mathbf{b}^q$$

Due to the flexibility of language, there may exist more than one reasonable expression that covers the same input but in different sequence. For example, "Chickens and rabbits were in the yard. Together they had 27 heads and 86 legs." can be rewritten as "Teacher finds 27 heads and 86 legs in the yard, in which there are only chickens and rabbits.". The former expression can be viewed as

the plan of first generating commonsense sentences and then the symbolic sentences, while the latter one is first generating symbolic sentences then commonsense sentences. We capture such diversity of reasonable presentations with both latent variable Z and input graphs C. Different samples of Z will lead to different self-planning results. To start the decoding process, we initialize the hidden state $\mathbf{h}_0 = [\mathbf{z}; \mathbf{g}^e_*; \mathbf{g}^k_*]$, where $\mathbf{z}$ is sampled from the posterior net $q(Z|C, Y) \sim \mathcal{N}(\boldsymbol{\mu}^q, \boldsymbol{\sigma}^q\mathbf{I})$ and the prior net $p(Z|C) \sim \mathcal{N}(\boldsymbol{\mu}^p, \boldsymbol{\sigma}^p\mathbf{I})$ during the training and inference procedures respectively.

At each decoding time step t, we dynamically decide the portions of input information from the equations and the CSKG respectively based on the current hidden state $\mathbf{h}_t$, which can keep track of the current generating state. We use the attention mechanism to conduct the self-planning between explicit symbolic equations and implicit CSKG. The dynamic self-planning module takes the decoder's current hidden state ($\mathbf{h}_t$), node representations of equations ($\mathbf{G}^e_*$) and CSKG ($\mathbf{G}^k_*$) as input and outputs the context-aware planning state ($\mathbf{c}_t$) of the current time step. Specifically, we compute $\mathbf{c}_t$ as follows:

$$\mathbf{c}_t = \beta_t * \mathbf{c}^e_t + (1 - \beta_t) * \mathbf{c}^k_t; \quad \beta_t = softmax(\mathbf{W}^\beta \mathbf{h}_t);$$

$$\mathbf{c}^e_t = \sum_{v \in \mathcal{V}^e} \alpha^e_{t,v} \mathbf{g}^e_v; \quad \mathbf{c}^k_t = \sum_{v \in \mathcal{V}^k} \alpha^k_{t,v} \mathbf{g}^k_v;$$

$$\alpha^e_{t,v} = \exp(\mathbf{o}^e_{t,v}) / \sum_{v' \in \mathcal{V}^e} \exp(\mathbf{o}^e_{t,v'});$$

$$\alpha^k_{t,v} = \exp(\mathbf{o}^k_{t,v}) / \sum_{v' \in \mathcal{V}^k} \exp(\mathbf{o}^k_{t,v'});$$

$$\mathbf{o}^e_{t,v} = \mathbf{v}^{e\top} \tanh(\mathbf{W}^e \mathbf{h}_t + \mathbf{U}^e \mathbf{g}^e_v);$$

$$\mathbf{o}^k_{t,v} = \mathbf{v}^{k\top} \tanh(\mathbf{W}^k \mathbf{h}_t + \mathbf{U}^k \mathbf{g}^k_v)$$

where β_t represents the self-planning distribution at time step t.

The final context vector is the fusion of the symbolic and commonsense knowledge graphs. The next-step hidden state ($\mathbf{h}_{t+1}$) is the combination of current hidden state ($\mathbf{h}_t$), self-planning context state ($\mathbf{c}_t$) and the representation of currently generated word ($\mathbf{w}_t$), i.e, $\mathbf{h}_{t+1} = \mathbf{GRU}(\mathbf{h}_t, \mathbf{W}^d[\mathbf{c}_t; \mathbf{w}_t] + \mathbf{b}^d)$ where $\mathbf{W}^d$ and $\mathbf{b}^d$ are the linear transformation matrix and the bias term. We further generate the next word by feeding hidden state $\mathbf{h}_{t+1}$ to linear transformation and softmax layer to get the next-token probability distribution.

The final objective function consists (1) maximizing the probability of ground-truth sequence texts, which promotes the predictions generated by the posterior net and the MWP generator closer to the distribution of the gold-standard data; and (2) minimizing the KL-divergence between posterior distribution ($p(Z|C, Y)$) and prior distribution ($p(Z|C)$).

4   Experiments

In this work, we crawled 5,447 MWPs of linear equations from a third-party website, and each MWP consists of two unknown variables and two equations. It covers 119 topics and the average length of an MWP is 62 words. In one CSKG, the average number of entities is 17.067 and the average number of edges is 29.102. We randomly select 544 of them as our validation set, and 546 of them as our gold-standard test (GT) set. Please note that different from previous work of automatically solving the MWPs such as *MAWPS* (Koncel-Kedziorski et al., 2016b) and *MathQA* (Amini et al., 2019), we focus on the generation task of MWPs of more than one linear equations in students' real-life scenarios by using topics in our CSKG. Both *MAWPS* and *MathQA* datasets do not contain MWPs that have two or three equations and variables. Furthermore, there is no explicit topics associated with the MWPs in these publicly available MWP datasets.

We use following evaluation metrics: (1) *BLEU-4*: the 4-gram overlap score against gold-standard sentences (Papineni et al., 2002); (2) *METEOR*: n-gram overlap with paraphrase and language-specific considerations (Denkowski and Lavie, 2014); (3) *ROUGE-L*: the overlap of longest common subsequence between candidate and gold-standard sentences (Lin, 2004); (4) *Self-BLEU*: the diversity measurement of averaging BLEU scores of four generated MWP pairs given the same input (Zhu et al., 2018).

Meanwhile, we conduct two human evaluation studies to comprehensively evaluate the quality of the generated MWPs. First, we ask three evaluators to rate from the following aspects ranging from 1 to 3: (1) *Equation Relevance*: how relevant is MWP with respect to the input equations? (2) *Topic Relevance*: how relevant is MWP with respect to the given topic? and (3) *Language Coherence*: whether the MWP is coherent and well-organized. We use the average scores from three human evaluators as our final results.

Before training, in order to ensure that each question is answerable, we first use sympy's equation solver[3] to solve all the algebraic equations in the

---

[3] https://docs.sympy.org/latest/modules/solvers/solvers.html

Method	BLEU-4	METEOR	ROUGE-L	Self-BLEU	Equation Rel.	Topic Rel.	Language Coh.
Template	19.836	22.685	36.545	57.309	2.493 (0.377)	2.276 (0.419)	2.256 (0.431)
CVAE	20.971	23.278	37.557	57.560	1.880 (0.514)	2.266 (0.498)	1.819 (0.529)
MAGNET	15.075	20.084	35.695	71.032	1.850 (0.579)	2.633 (0.502)	1.725 (0.502)
UniLM	17.499	20.562	34.934	74.198	2.103 (0.614)	2.546 (0.720)	1.683 (0.605)
Transformer	21.612	24.423	39.250	77.571	2.486 (0.537)	2.580 (0.413)	2.313 (0.476)
MaKE w/o symbolic	19.788	22.386	38.459	63.849	2.016 (0.670)	**2.890** (0.317)	2.376 (0.522)
MaKE w/o CSKG	15.441	19.662	33.188	**56.683**	2.491 (0.682)	2.240 (0.271)	2.092 (0.634)
MaKE w/o planning	21.611	23.782	40.053	62.549	2.575 (0.695)	2.695 (0.528)	2.195 (0.472)
MaKE	**23.322**	**24.537**	**40.076**	60.29	**2.668** (0.595)	2.788 (0.273)	**2.492** (0.484)

Table 2: Evaluation results (± standard deviation) on GT set. *Rel.* and *Coh.* are short for relevance and coherence.

dataset. We tokenize our training data with BPE method (Kudo and Richardson, 2018) and extend the subword vocabulary with our pre-defined special tokens. More preprocessing details can be found in Appendix A.3. During training, we initialize the GGNN parameters with normal distribution $N(0, 0.02)$ and the number of GGNN hops is set to 3. The dimension of word embedding is 128 with random initialization. We utilize GRU for all RNNs and the hidden state size is 512. The size of sampled latent variable is set to 128, and we apply reparameterization trick during training and inference. We set the teacher forcing probability to 0.5 and train our model using Adam optimizer (Kingma and Ba, 2014) with learning rate scheduling. The batch size is set to 32 and the beam search width is set to 5. All hyper-parameters are tuned on the development set. We use linear KL annealing technique following Fu et al. (2019) to alleviate the KL collapse problem and apply scheduled sampling to alleviate the exposure bias problem in GRU training (Bengio et al., 2015).

We compare our *MaKE* against several strong baselines: (1) the template based method, i.e., *Template*; (2) conditional VAE that captures the diversity in the encoder and uses latent variables to learn a distribution over potential intents, i.e., *CVAE* (Zhao et al., 2017); (3) an RNN-based seq2seq model with equation-topic fusion mechanism and entity-enforced loss, i.e., *MAGNET* (Zhou and Huang, 2019); (4) the SOTA pre-trained language model with a shared Transformer network and self-attention masks, i.e., *UniLM* (Dong et al., 2019); and (5) a standard Transformer-based seq2seq model, i.e., *Transformer* (Vaswani et al., 2017). More details are provided in Appendix A.4.

Please note that we do not select rewriting based approaches as the baselines in this work. This is because rewriting based approaches require a very

large pre-stored question bank and it only works when the input equations are matched in the question bank.

4.1 Results and Analysis

Evaluation Results on GT Set. Results on the GT set are listed in Table 2, which shows that our *MaKE* outperforms all baseline methods in terms of both automatic and human evaluation metrics. Specifically, from Table 2, we find: (1) comparing *MaKE* and *Template*, *Template* doesn't perform well in language coherence and topic relevance. This is because the MWP templates are stereotyped. Mismatches between the template context and the re-filled words lead to incoherent texts; and (2) comparing *MaKE* and seq2seq baselines, with rich representations of equations and CSKG, *MaKE* is able to better capture mathematical relations and improve MWP quality with real-life plots under the given topic.

Turing Test Results on GT Set. For each existing MWP in the GT set, we generate a new MWP of the same equations but with a different topic. We show such pairs to the human evaluators and ask them to distinguish which one is the generated MWP. We measure the results of this artificial "Turing Test" via *Fool Ratio*, i.e., the fraction of instances in which a model is capable of fooling the evaluators. Ideally, perfect MWP generation will lead to random guesses and the ideal *Fool Ratio* would be 50%. Finally, we get an averaged *Fool Ratio* of 39.38% (36.08%, 42.49% and 39.56% from three annotators respectively). This demonstrates that the generation quality is 78.76% (39.38/50) as good as the quality from human teachers.

Ablation Study. We compare MaKE with three different ablation methods, namely *MaKE w/o symbolic*, *MaKE w/o CSKG* and *MaKE w/o planning*. Specifically, for *MaKE w/o symbolic*, we only in-

Equations: y-x=6; 8y-4x=64;	**Topic**: Rowing boat; **Entities**: x: Small boat; y: Big boat;
CVAE	There are 6 in ChangLong Park. <unk> 4 small boats, 8 big boats, small boat can carry 64 people. There are 64 more people picking in big boats than in small boats. How many people are there on the big boat?
MAGNET	Reward, small boat, each can sit 4. Big boat, each can sit 8. Small boat more than 6, more than 64. Please tell the number of small boat?
UniLM	There are small boats and big boats in the competition. There are 6 sitting in big boats, 8 big boats. On the scene, one is more than the big boat with 8 people, there are sitting in small boat, 64 people for total.
Transformer	In order to reward the students who did well in this test, Teacher Fang decided to take 4 and 6 with a total of 64 people to go boating on the weekend! A small boat can seat 4 people, and a big boat can seat 8 people. Teacher Fang rents 6 more small boats than the big boats. How many small boats does Teacher Fang rent?
MaKE	A company has two types of boats, and the number of big boats is 6 more than that of small boats. Each small boat can accommodate 4 people and each big boat can accommodate 8 people. When all boats are filled up, the number of people in the small boats is 64 less than that in the big boats. How many big boats are there?

Table 3: Illustrative examples of the MWP generation comparison with unseen equations. There is no results from *Template* because it doesn't work on unseen equations. The incorrect part is highlighted in red color.

put the CSKG to the model, get rid of the equation-based symbolic graph and leave the other components unchanged. For *MaKE w/o CSKG*, we retain symbolic graph in the input and discard CSKG. For *MaKE w/o planning*, the input remains unchanged, but the decoder becomes a normal GRU and the attention score is computed for all nodes in symbolic graph and CSKG simultaneously.

Table 2 shows the results of ablation study. Without the self-planning module, we observe that the model's self-BLEU performance has decreased, which empirically supports our assumption that the design of self-planning module can capture the flexibility in the language. Meanwhile, the performance of our model drops by 1.71% in *BLEU-4*, 0.75% in *METEOR*, 0.02% in *ROUGE-L* and 2.259% in *Self-BLEU*, which also proves the effectiveness of self-planning module. The *MaKE w/o CSKG* approach achieves the best *Self-BLEU* score but the worst human evaluation scores, which indicates that the representations of CSKG help form valid MWPs in real-life scenarios. This is because we utilize CSKG as a commonsense constraint on the generated MWPs, which results in a limited number of words that can be generated by the *MaKE* method under that condition. When we remove such constraint in *MaKE w/o CSKG*, the model only needs to satisfy the symbolic equation conditions, regardless of what the topic is or which entity the unknown variable corresponds to. Hence the search space for words will become larger, which will directly increase the Self-BLEU score. However, it has the drawback that may cause the generated texts to violate the commonsense knowledge. The *MaKE w/o symbolic* shows a sig-

nificant decrease on all the automatic evaluation metrics except for the topic relevance score, which is reasonable since understanding of the mathematical variables, numerical quantities, operations, and their relations is essential in generating logical coherent MWPs.

Equations: x=y; 2x+4y=48; **Topic**: Livestock
Entities: x: Chicken; y: Rabbit;
1. Rabbits and chicken are in one cage. The number of rabbits is 0 less than that of chickens. They have 48 legs in total. How many rabbits and chickens in cage?
2. There are the same number of chickens and rabbits in the yard, and the total number of legs is 48. How many rabbits and chickens are in the yard?
3. Chicken and rabbits are in the same cage. Xiaoming counted the number of heads of the two animals and found that the number of chicken heads was 0 more than the number of rabbit heads. There are 48 legs in total. May I ask how many chickens are there?
4. Xiaojun is very good at math, but today there is a difficult problem for him: A chicken has 1 head and 2 legs, and a rabbit has 1 head and 4 legs. There are chickens and rabbits in the same cage, and the number of chickens is equal to the number of rabbits. There are 48 feet in total, so how many chickens and how many rabbits are there?

Table 4: An illustrative example of the diverse MWP generation made by *MaKE*.

Qualitative Case Study. Because of the GGNN encodings of equations, our *MaKE* model is able to handle a wide range of mathematical relations, including both addition and subtraction, i.e., a, b, m, c, d and n may be either positive or negative in $ax + by = m$; $cx + dy = n$. We quantitatively compare the generation quality of *MaKE* with other baselines and the results are shown in Table 3. Fur-

thermore, we show the diverse results of *MaKE* qualitatively in Table 4. Additional examples can be found in Appendices A.5 - A.6. As we can see, (1) *CVAE* and *Transformer* cannot interpret the equations correctly and fail to generate desired MWPs; (2) our *MaKE* approach is able to generate diverse enough MWPs in real-life scenarios.

Large-scale Human Evaluation Results. Besides evaluations on the GT set, which is usually limited in educational scenarios (Xu et al., 2019; Wang et al., 2020), we conduct evaluations on the large-scale generated results. We randomly create 100 valid linear equations and ensure that none of them appears in our training set. Meanwhile, we select top 30 common real-life topics. For each pair of equation and topic, we generate 5 MWPs accordingly and therefore, we obtain 15,000 MWPs. We conduct a human evaluation to assess the quality of these generated MWPs and the results are shown in Table 5. We can see that our method outperforms baseline models by a large margin.

Method	Equation Rel.	Topic Rel.	Language Coh.
CVAE	1.583 (0.493)	2.550 (0.487)	1.366 (0.605)
MAGNET	1.603 (0.430)	2.467 (0.222)	1.517 (0.508)
UniLM	1.451 (0.530)	2.416 (0.690)	1.150 (0.441)
Transformer	1.699 (0.395)	2.351 (0.391)	2.416 (0.460)
MaKE	**2.308 (0.507)**	**2.558 (0.228)**	**2.505 (0.461)**

Table 5: Evaluation results ($\pm$ standard deviation) on the large-scale generated MWP data. There is no results from *Template* because it doesn't work on unseen equations.

Error Analysis. To better understand the limitation of our approach, we manually review 150 equations and the corresponding generated MWPs. The two major problems are: missing information and language disfluency. We show two representative examples in Table 6. In the example of missing information, the information that the small boat can accommodate 2 people and big boat can accommodate 4 people are ignored because some MWPs in the training set often ignore this "pre-leaned" knowledge like chickens have two legs. Language disfluency problem is introduced due to the limit size of training data under certain specific topic. This can be alleviated or addressed by either collecting more MWP training data or provide more information in CSKG to explicitly control the context of the generated text, such as the fact that livestock often live on farms and marine animals are found in the ocean, etc.

Equations: x-y=6; 2x-4y=10;	**Topic**: Rowing boat
Entities: x: Small boat; y: Big boat;	
Missing information	Teacher Mr.Huang and his 35 students come to row the boat. They find 6 more small boats than big boats. There are 10 more students in the small boat than in the big boat. How many big boats are there?

Equations: x-y=1; 6x-8y=0;	**Topic**: Insects
Entities: x: Cockroaches; y: Ants;	
Language disfluency	There are two types of heads: cockroaches and ants. Cockroaches have 1 more head than ants, and cockroaches have 0 more than ant legs. How many cockroaches and ants respectively?

Table 6: Illustrative examples that demonstrate the typical problems of the current system.

5 Conclusion

In this paper, we presented a neural encoding-decoding architecture for MWP generation. Comparing with the existing NQG algorithms, the advantages of our *MaKE* are: (1) it extracts intrinsic representations of both the equation based symbolic graph and the CSKG; (2) it automatically selects and incorporates information from equations and knowledge graphs during the decoding process; and (3) it is able to generate relevant, coherent and diverse MWPs in students' real-life scenarios. Experimental results on real-world educational MWP data sets demonstrate that *MaKE* outperforms other SOTA NQG approaches in terms of both automatic evaluation metrics and human evaluation metrics. In the future, we plan to explore the MWP generation problems for more mathematical variables with high-order operations, and explore the method to incorporate commonsense knowledge from publicly available CSKG like ConceptNet or Wikipedia.

Acknowledgements

References

Aida Amini, Saadia Gabriel, Shanchuan Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Ha-

jishirzi. 2019. MathQA: Towards interpretable math word problem solving with operation-based formalisms. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2357–2367.

Daniel Beck, Gholamreza Haffari, and Trevor Cohn. 2018. Graph-to-sequence learning using gated graph neural networks. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, pages 273–283.

Andinet Assefa Bekele. 2020. Automatic generation of amharic math word problem and equation. *Journal of Computer and Communications*, 8(8):59–77.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 1171–1179.

Thomas P Carpenter, Mary Kay Corbitt, Henry S Kepner, Mary Montgomery Lindquist, and Robert E Reys. 1980. Solving verbal problems: Results and implications from national assessment. *The Arithmetic Teacher*, 28(1):8–12.

Suleyman Cetintas, Luo Si, Yan Ping Xin, Dake Zhang, Joo Young Park, and Ron Tzur. 2010. A joint probabilistic classification model of relevant and irrelevant sentences in mathematical word problems. *Journal of Educational Data Mining*, 2(1):83–101.

Paul Deane and Kathleen Sheehan. 2003. Automatic item generation via frame semantics: Natural language generation of math word problems.

Michael Denkowski and Alon Lavie. 2014. Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of The Ninth Workshop on Statistical Machine Translation*, pages 376–380.

Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. 2019. Unified language model pre-training for natural language understanding and generation. In *Advances in Neural Information Processing Systems*, pages 13063–13075.

Hao Fu, Chunyuan Li, Xiaodong Liu, Jianfeng Gao, Asli Celikyilmaz, and Lawrence Carin. 2019. Cyclical annealing schedule: A simple approach to mitigating kl vanishing. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 240–250.

Jian Guan, Xiaoxi Mao, Changjie Fan, Zitao Liu, Wenbiao Ding, and Minlie Huang. 2021a. Long text generation by modeling sentence-level and discourse-level coherence. *arXiv preprint arXiv:2105.08963*.

Jian Guan, Zhexin Zhang, Zhuoer Feng, Zitao Liu, Wenbiao Ding, Xiaoxi Mao, Changjie Fan, and Minlie Huang. 2021b. OpenMEVA: A benchmark for evaluating open-ended story generation metrics. *arXiv preprint arXiv:2105.08920*.

Daya Guo, Yibo Sun, Duyu Tang, Nan Duan, Jian Yin, Hong Chi, James Cao, Peng Chen, and Ming Zhou. 2018. Question generation from sql queries improves neural semantic parsing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1597–1607.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Rik Koncel-Kedziorski, Ioannis Konstas, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2016a. A theme-rewriting approach for generating algebra word problems. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1617–1628.

Rik Koncel-Kedziorski, Subhro Roy, Aida Amini, Nate Kushman, and Hannaneh Hajishirzi. 2016b. 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.

Taku Kudo and John Richardson. 2018. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71.

Ghader Kurdi, Jared Leo, Bijan Parsia, Uli Sattler, and Salam Al-Emari. 2020. A systematic review of automatic question generation for educational purposes. *International Journal of Artificial Intelligence in Education*, 30(1):121–204.

Friedrich Wilhelm Levi. 1942. *Finite geometrical systems: six public lectues delivered in February, 1940, at the University of Calcutta*. University of Calcutta.

Yikang Li, Nan Duan, Bolei Zhou, Xiao Chu, Wanli Ouyang, Xiaogang Wang, and Ming Zhou. 2018. Visual question generation as dual task of visual question answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6116–6124.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Haochen Liu, Zitao Liu, Zhongqin Wu, and Jiliang Tang. 2020. Personalized multimodal feedback generation in education. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 1826–1840.

Vijini Liyanage and Surangika Ranathunga. 2020. Multi-lingual mathematical word problem generation using long short term memory networks with enhanced input features. In *Proceedings of The 12th Language Resources and Evaluation Conference*, pages 4709–4716.

DeKita G Moon-Rembert and Juan E Gilbert. 2019. Illmatics: A web-based math word problem generator for students' distal and proximal interests. In *E-Learn: World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education*, pages 842–848. Association for the Advancement of Computing in Education.

Nasrin Mostafazadeh, Ishan Misra, Jacob Devlin, Margaret Mitchell, Xiaodong He, and Lucy Vanderwende. 2016. Generating natural questions about an image. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 1802–1813.

John C Moyer, Larry Sowder, Judith Threadgill-Sowder, and Margaret B Moyer. 1984. Story problem formats: Drawn versus verbal versus telegraphic. *Journal for Research in Mathematics Education*, pages 342–351.

Liangming Pan, Wenqiang Lei, Tat-Seng Chua, and Min-Yen Kan. 2019. Recent advances in neural question generation. *arXiv preprint arXiv:1905.08949*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

Oleksandr Polozov, Eleanor O'Rourke, Adam M Smith, Luke Zettlemoyer, Sumit Gulwani, and Zoran Popović. 2015. Personalized mathematical word problem generation. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*.

DeKita Moon Rembert, Naja A Mack, and Juan E Gilbert. 2019. Exploring the needs and interests of fifth graders for personalized math word problem generation. In *Proceedings of the 18th ACM International Conference on Interaction Design and Children*, pages 592–597.

Luana Ruiz, Fernando Gama, and Alejandro Ribeiro. 2019. Gated graph convolutional recurrent neural networks. In *2019 27th European Signal Processing Conference*, pages 1–5. IEEE.

Xingwu Sun, Jing Liu, Yajuan Lyu, Wei He, Yanjun Ma, and Shi Wang. 2018. Answer-focused and position-aware neural question generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3930–3939.

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.

Ke Wang and Zhendong Su. 2016. Dimensionally guided synthesis of mathematical word problems. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, pages 2661–2668.

Wentao Wang, Guowei Xu, Wenbiao Ding, Yan Huang, Guoliang Li, Jiliang Tang, and Zitao Liu. 2020. Representation learning from limited educational data with crowdsourced labels. *IEEE Transactions on Knowledge and Data Engineering*.

Sandra Williams. 2011. Generating mathematical word problems. In *2011 AAAI Fall symposium series*.

Guowei Xu, Wenbiao Ding, Jiliang Tang, Songfan Yang, Gale Yan Huang, and Zitao Liu. 2019. Learning effective embeddings from crowdsourced labels: An educational case study. In *2019 IEEE 35th International Conference on Data Engineering*, pages 1922–1927. IEEE.

Kaichun Yao, Libo Zhang, Tiejian Luo, Lili Tao, and YanJun Wu. 2018. Teaching machines to ask questions. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pages 4546–4552.

Jianxing Yu, Xiaojun Quan, Qinliang Su, and Jian Yin. 2020. Generating multi-hop reasoning questions to improve machine reading comprehension. In *Proceedings of The Web Conference 2020*, pages 281–291.

Xingdi Yuan, Tong Wang, Caglar Gulcehre, Alessandro Sordoni, Philip Bachman, Saizheng Zhang, Sandeep Subramanian, and Adam Trischler. 2017. Machine comprehension by text-to-text neural question generation. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 15–25.

Shiyue Zhang and Mohit Bansal. 2019. Addressing semantic drift in question generation for semi-supervised question answering. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 2495–2509.

Tiancheng Zhao, Ran Zhao, and Maxine Eskenazi. 2017. Learning discourse-level diversity for neural dialog models using conditional variational autoencoders. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 654–664.

Qingyu Zhou and Danqing Huang. 2019. Towards generating math word problems from equations and topics. In *Proceedings of the 12th International Conference on Natural Language Generation, INLG 2019,*

Tokyo, Japan, October 29 - November 1, 2019, pages 494–503.

Qingyu Zhou, Nan Yang, Furu Wei, and Ming Zhou. 2018. Sequential copying networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Yaoming Zhu, Sidi Lu, Lei Zheng, Jiaxian Guo, Weinan Zhang, Jun Wang, and Yong Yu. 2018. Texygen: A benchmarking platform for text generation models. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 1097–1100.

A Appendix

A.1 Equation Variants

In our scenario, the general expression formula can be formed as follows:

$$\eta_0\varphi_0 x \eta_1\varphi_1 y = \varphi_2\eta_2\varphi_3 \tag{1}$$

$$\eta_3\varphi_4 x \eta_4\varphi_5 y = \varphi_6\eta_5\varphi_7 \tag{2}$$

where $\eta_* \in \{+, -, \times, \div\}$ are operators in equations, and φ_* are numeric numbers. To be noticed, only one of the operators between η_0 (η_3) and η_1 (η_4) may be a minus operator, or neither. The equation variants are derived from different configurations of operators η_*.

We go into detail for eq.(1) discussing all its possible variants, eq.(2) keeps the same behavior as eq.(1), and different combinations of eq.(1) and eq.(2) will lead to different system of linear equation in two unknowns. According to the presence or absence of the operator η_2 and numeric number φ_3, we show two different possible symbolic graph structures in Figure 3. In Figure 3 (a), φ_2 is equal to "m", η_2 and φ_3 are empty. Thus we connect node a and node m with relation "Minuend to res"; connect node b and node m with relation "Subtrahend to res"; representing ax and by are minuend and subtrahend element in "ax-by=m" respectively. In Figure 3 (b), φ_2, η_2 and φ_3 are equal to "c", "+" and "d" respectively. In order to be consistent with the graph structure described in Figure 3 (a), we first add a dummy node dum in our symbolic graph, then connect node c and node dum with relation "Add to dummy", and connect node d and node dum with relation "Add to dummy". In this way, the dummy node can represent the expression "c+d".
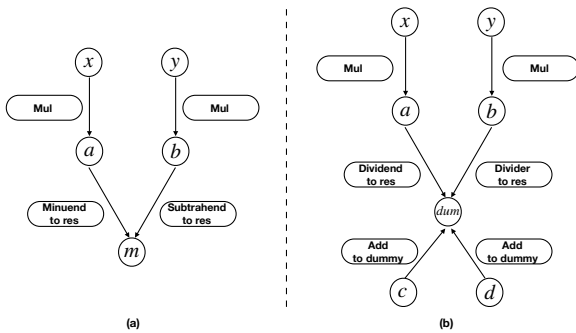

Figure 3: (a) a symbolic graph of equation "ax-by=m"; (b) a symbolic graph of equation "ax/by=c+d"; *dum* denotes the dummy node in the symbolic graph.

A.2 Levi Graph Transformation

Let $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathcal{R}\}$ be a directed symbolic graph with nodes $v_i \in \mathcal{V}$ and labeled edges $(v_i, r, v_j) \in \mathcal{E}$. As shown in Figure 2 (a), where $r \in \mathcal{R}$ is a relation type, i.e., *Add to res*, *Mul*, etc. Let $|\mathcal{V}|$ and $|\mathcal{E}|$ denote the number of nodes and edges, respectively. We convert the graph $\mathcal{G}$ into an unlabeled and directed bipartite graph $\mathcal{G}^t = \{\mathcal{V}^t, \mathcal{E}^t\}$ with levi transformation by converting each labeled edge $(v_i, r, v_j) \in \mathcal{E}$ into two unlabeled edges $(v_i, r), (r, v_j) \in \mathcal{E}^t$, where $|\mathcal{V}^t| = |\mathcal{V}| + |\mathcal{E}|$. Intuitively, transforming a graph into its Levi graph form turns original edges into additional nodes, which allows us to directly encode edge label information with word embeddings and guarantee the relation message passing with multi-hop reasoning.

A.3 Training and Testing

We obtain non-lexical text by replacing the numbers in the question text with the pre-defined special tokens in our symbolic equation graph and CSKG. The procedures are similar to the example in Table 8 with the following differences:

- Matching words for unknown variables are first extracted from the gold MWP, and query our private database with the given topic word to construct our commonsense knowledge graph.

- *MaKE* transforms operators η_* into equation graph edge labels (relations), and numeric number φ_* into equation graph nodes v.

- More words in MWP texts are replaced with special tokens in CSKG. Take the sentence in Table 8 as an example, *wheels* are replaced by one node (*counting entity*) in the corresponding commonsense graph.

During training *MaKE*, the input is CSKG and the equation based symbolic graph, and the output target is the delexicalized words sequence. We apply the same word refilling post-processing procedure to obtain the final MWP.

A.4 Baseline Methods Details

Template In addition to neural baselines, we use a problem-specific, template-based generator. The template-based method first finds MWP problems with the same type of input equations in the question bank given the input topic words. For instance, the query equation is $x + y = 6; 2x - 4y = 6$ and

Equation templates: x+y=α; bx-cy=d	**Topic**: vehicle
Query text: There are α *x_entity* and *y_entity* in the parking lot. Each *x_entity* has *b* wheels and each *y_entity* has *c* wheels. *x_entity* has *d* more total wheels than *y_entity*. How many *x_entity* are there?	
Query variables: *x_entity*: motorcycles, *y_entity*: cars	
Generated MWP There are 6 motorcycles and cars in the parking lot. Each motorcycles has 2 wheels and each cars has 4 wheels. Motorcycles has 6 more total wheels than cars. How m--any motorcycles are there?	

Table 7: Query process in our template-based-method.

Input equation: x+y=6; 2x-4y=6	**Topic**: vehicle
General expression formula:	$\eta_0 \varphi_0 x \eta_1 \varphi_1 y = \varphi_2 \eta_2 \varphi_3$
	$\eta_3 \varphi_4 x \eta_4 \varphi_5 y = \varphi_6 \eta_5 \varphi_7$
Input sequence expression: $[\eta_0, \varphi_0, \eta_1, \varphi_1, \varphi_2, \eta_2, \varphi_3, \eta_3, \varphi_4, \eta_4, \varphi_5, \varphi_6, \eta_5, \varphi_7, \text{Topic}]$	
Input sequence for given example: [pad, φ_0, +, φ_1, φ_2, pad, pad, pad, φ_4, -, φ_5, φ_6, pad, pad, *x_entity*, *y_entity*, vehicle]	
Output MWP: There are φ_2 *x_entity* and *y_entity* in the parking lot. Each *x_entity* has φ_4 wheels and each *y_entity* has φ_5 wheels. *x_entity* has φ_6 more total wheels than *y_entity*. How many *x_entity* are there?	

Table 8: Input sequence for seq2seq method, φ_* are numeric number in equations, and η_* are operators. If there is no valid operator or number for a given special token, we fill it with a pad token.

the query topic is *vehicle*. As shown in Table 7, we first delexicalize the input equation pairs with special tokens and save them for post-processing. After query our question bank with the delexicalized equation pairs and the topic word, we obtain the pre-stored MWP template and matching words for unknown variables. Finally we fill the MWP template with the previously saved delexicalized words and obtain the generated MWP.

CVAE (Zhao et al., 2017) Similar to previous work, we apply a seq2seq model and adopt a latent variable to capture the diversity of MWPs. We replace the hierarchical encoder with a one-layer GRU, and the initial state of the decoder is the combination of a latent variable and the final state of the encoder. As shown in Table 8, we apply the delexicalization process and sequence transformation operations for all the training data. The input sequence includes special tokens, operators and the topic word. After refilling the special tokens with corresponding matching words, we obtain the final MWP.

MAGNET (Zhou and Huang, 2019) MAGNET is a previously proposed seq2seq MWP generation framework. Following the original implementation, we utilize a bidirectional RNN to encode equation sequence and encode topic word with a word representation lookup table. The decoder is a single directional RNN with equation-topic fusion mechanism to leverage both equation and topic information. We follow the same input sequence as described in Table 8, but split the equation sequence and the topic words as separate inputs.

UniLM (Dong et al., 2019) A pre-trained natural language generation model with transformer encoder and decoder blocks. We fine-tune UNILM on MWP generation task with the same input and output token sequence described in CVAE method.

Transformer (Vaswani et al., 2017) We included a Transformer-based seq2seq model which has proved its success in machine translation tasks. The same input sequence as described in previous method.

A.5 Additional MWP Generation Comparison

We provide additional illustrative examples of the MWP generation comparison with unseen equations in Table 9.

A.6 Additional Diverse MWP Results

We provide additional diverse MWP results in Table 10.

	Equations: x=2y; 4x+6y=56; **Topic**: Vehicle
	Entities: x: Small car; y: Large car;
CVAE	There are many small cars and big cars in the parking lot. There are 6 people in these cars and 56 people in big cars. How many small cars and big cars are there?
MAGNET	We need to clean up a total of large cars and small cars. We know that small car can transport 6, and the number of large cars is the number of times. Please tell the number of small cars?
UniLM	Doctors have produced a lot of small cars and large cars. There are 0 cars in total. These two types of cars have 56 people. How many of these two cars have a total of 56 people, small cars?
Transformer	There are 56 people in the class to go rowing, and there are 0 cars in total. Among them, the small cars have 4 people each, and the big cars have 6 people each. The small cars have (), and the big cars have ().
MaKE	A company needs to transport 56 wheels in one transportation task. We know that each large car can carry 6 wheels and each small car can carry 4 wheels. The company has completed the task with only one time transportation, and the number of used small car is twice the number of large car. Q: They used () small car.
	Equations: -x+y=20; 2x-4y=10; **Topic**: Livestock
	Entities: x: Chicken; y: Rabbit;
CVAE	Chickens and rabbits in the same cage. The number of chickens is twice the number of chickens. The number of rabbits and rabbits is twice the number of chickens, and the total number of legs is 10. How many chickens and rabbits are there each?
MAGNET	Chickens and rabbits in the same cage. The number of chickens is 20 more than that, and there are 10 rabbit legs. How many rabbits are there?
UniLM	How many chickens and rabbits are there in the 10-legged training centre?
Transformer	There were 20 chickens and rabbits. The total number of legs of the chickens was 10 more than the rabbits, and the chickens had () only.
MaKE	The number of chickens is 20 less than the number of rabbits, and the number of feet for chickens is 10 more than that of rabbits. How many chickens and rabbits are there?
	Equations: x+y=192/16; 22x+14y=192; **Topic**: Dormitory
	Entities: x: Large dormitory; y: Small dormitory;
CVAE	In the average daily None dormitory of the fifth grade, Teacher Chen gave four students, two small dormitories can accommodate 22 students. Two groups can accommodate 22 students, and the residence is just so big that the dormitory can accommodate 192 students. There are () rooms in the dormitory group. There are 14 students in the small dormitory, which just happened to be full.
MAGNET	There are 16 people in HongKong. The large dormitory can accommodate 22 people, and the small dormitory can accommodate 14 people. A total of 192 people, how many large dormitory?
UniLM	The school arranges accommodation for students. The big dormitory can accommodate 22 students in the dormitory, and the small dormitory can accommodate students. A total of 192 students can live in the dormitory. On average, how many rooms are there in each small dormitory?
Transformer	192 students from grade 1 to grade 6 go to the spring trip. There are 16 teachers in total. Students have two kinds of dormitories to choose. The large dormitory can live in 22 people, and the small dormitory can live in 14 people. In total, 192 dormitories are booked. What is the number for the booked small dormitory and large dormitory separately?
MaKE	The Youth Hostel is designed to accommodate 22 people in the large dormitory and 14 people in the small dormitory. One day, there were 192 travelers and the hostel was just about full with an average of 16 people per dormitory. How many large dormitories are there in the youth hostel?

Table 9: Illustrative examples of the MWP generation comparison with unseen equations. () represents the question that the student needs to solve.

Equations: x+y=100; 20x+30y=2600; **Topic**: Buy ticket
Entities: x: front row ticket; y: back row ticket
1. The company organized staff to go to the Shanghai Circus to see the show. Tickets purchased on the official website would receive discounts, 20 yuan off for each front row ticket and 30 yuan off for each back row ticket. They bought 100 tickets on official website and saved 2,600 yuan, how many front row tickets did they buy?
Entities: x: child ticket; y: adult ticket
2. There are adult and child tickets available at the acrobatic ticket office in Linjiang Park. The price of the adult ticket is 30 yuan and the price of the child ticket is 20 yuan. A total of 100 acrobatic tickets were sold today, and the revenue was 2,600 yuan. How many adult tickets were sold?
Entities: x: one-way ticket; y: round-trip ticket
3. The bus station sold 100 tickets today and received 2,600 yuan. There were two kinds of tickets on their website, a round-trip ticket for ¥30 and a one-way ticket for ¥20. How many round-trip tickets were sold today?
Equations: -y+x=20; 2y+4x=66; **Topic**: Livestock
Entities: x: cow; y: duck
1. When Sun Wukong returned to Mount Huaguo, he showed a magic spell to the monkeys. A group of cows and a group of ducks emerged. After counting, they found that there were 66 legs. If we know that the number of cows is 20 more than that of ducks. Then how many cows and ducks for each?
Entities: x: cow; y: duck
2. Xiaohong's farm has ducks and cows. The number of ducks is 20 less than the number of cows. There are 66 legs in total, Xiaohong have () cows and () ducks.
Entities: x: cow; y: duck
3. In a pasture, there are cows and ducks. There are 20 more cows than ducks, and the total number of legs of the cows and ducks is 66. There are () ducks?

Table 10: Additional illustrative example of the diverse MWP generation made by *MaKE*. () represents the question that the student needs to solve.