DISK: Domain-constrained Instance Sketch for Math Word Problem Generation

Tianyang Cao[1]*, **Shuang Zeng**[1,2]*, **Xiaodan Xu**[1,3], **Mairgup Mansur**[2], **Baobao Chang**[1]†
[1]Key Laboratory of Computational Linguistics, MOE, Peking University
[2]Tencent Inc.
[3]School of Software and Microelectronics, Peking University
{ctymy, zengs, diane1968, chbb}@pku.edu.cn, mairgupma@tencent.com

Abstract

A math word problem (MWP) is a coherent narrative which reflects the underlying logic of math equations. Successful MWP generation can automate the writing of mathematics questions. Previous methods mainly generate MWP text based on inflexible pre-defined templates. In this paper, we propose a neural model for generating MWP text from math equations. Firstly, we incorporate a matching model conditioned on the domain knowledge to retrieve a MWP instance which is most consistent with the ground-truth, where the domain is a latent variable extracted with a domain summarizer. Secondly, by constructing a Quantity Cell Graph (QCG) from the retrieved MWP instance and reasoning over it, we improve the model's comprehension of real-world scenarios and derive a domain-constrained instance sketch to guide the generation. Besides, the QCG also interacts with the equation encoder to enhance the alignment between math tokens (e.g., quantities and variables) and MWP text. Experiments and empirical analysis on educational MWP set show that our model achieves impressive performance in both automatic evaluation metrics and human evaluation metrics.

1 Introduction

Text generation has been broadly studied as an important task in the field of natural language processing. It aims to generate natural language text that is fluent, readable and faithful to the original input. Recent text generation studies mainly focus on the data-to-text generation, which generates textual output from structured data such as tables of records or knowledge graphs (KGs) (Puduppully et al., 2019a; Chen et al., 2019; Gong et al., 2019b; Zhao et al., 2020). In this paper, we focus on a relatively new type of data-to-text generation task: generating Math Word Problems (MWP) from equations (Zhou and Huang, 2019), which does not seem to have been fully studied by the community. Figure 1 shows two examples of this task. We aim to automatically generate coherent narratives which reflect the computational relationship within given equations. Successful math word problems generation has the potential to automate the writing of mathematics questions given equations to be solved. It can alleviate the burden of school teachers and further help improve the teaching efficiency.

Equations: $x = 6 * y$; $(x + y) * 3 = 147$
Problem: Jane travels ***6*** times faster than mike. Traveling in opposite directions they are ***147*** miles apart after ***3*** hrs. ***Find their rates of travel.***

Equations: $(1 - 1/3 - 9/20) * x = 245$
Problem: At a local high school, ***3/8*** of the students are freshman. ***1/4*** are juniors. And ***245*** are seniors. ***Find the total number of students.***

Figure 1: Two examples selected from the MWP generation dataset.

However, different from other data-to-text generation tasks, generating MWP text from abstract math equations is much more challenging. Firstly, an equation can be expressed by different MWP texts which differ in topic, style or grammar, known as one-to-many pattern. Take the Equation 1 in Figure 1 for example, "$x = 6 * y$" can be expressed as "Jane travels 6 times faster than Mike.", but it is also okay to express it as "The price of oranges is six times the apples.". So when grounding the input abstract math equations into a specific math problem, it is hard for a model to decide which scenes to choose for generation. Secondly, the math tokens in equations and natural language text in problems are from completely different symbolic representation space. So this gap increases the difficulty of establishing alignments between math tokens and natural language words, as shown in Figure 2. Such issue also confuses the generator thus makes generation process uncontrollable.

To overcome these challenges, we propose a

* Equal contribution.
† Corresponding author.

Proceedings of the 29th International Conference on Computational Linguistics, pages 6327–6339
October 12–17, 2022.

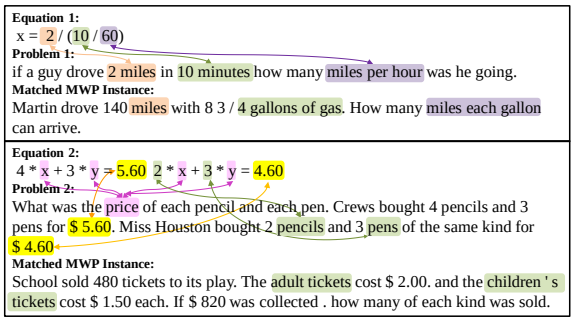

Figure 2: Two examples for illustration of the model. The corresponding MWP instance of each example is selected by a domain-dominated retriever with the participation of domain knowledge. The soft alignments between the math tokens and the corresponding MWP text spans are labeled in the same color.

novel MWP generator with **D**omain-constrained **I**nstance **SK**etch (**DISK**). We point out that MWP generation task can benefit from instance sketch restricted by domains. For two examples in Figure 2, DISK first utilizes a domain-dominated retriever to select an instance from a set of candidate MWP text. This instance can be regarded as to constrain the scene with which the math problem text to be generated would be related. Then, DISK produces a refined instance sketch from this instance to specify certain patterns for generating MWP text, using a Quantity Cell Graph (QCG) constructed from the instance. This graph reflects the backbone of the instance with entities and actions related to quantities as nodes. We conduct reasoning over QCG using Graph Convolutional Network (Kipf and Welling, 2017) to extract the final instance sketch, where math tokens can be contextualized with corresponding attributes and predicates via interaction between QCG nodes and equations. Finally, the model can generate MWP text with the refined instance sketch using a sequence generator.

Our contributions can be summarized as follows:

- We propose a domain-constrained instance sketch guided MWP text generation model, in which the domain information correspoding to the MWP text is automatically induced.
- Our model generates the instance sketch via Quantity Cell Graph enhanced encoding, it also contextualizes the math tokens with corresponding attributes and predicates via interaction between QCG nodes and equations.

Experiments show our model can generate more fluent and domain consistent MWP text, with promising performance improvement over strong baselines.

2 Related Work

MWP Generation: Early MWP generation methods are mostly template-based, including Answer Set Programming (ASP) (Polozov et al., 2015), schema and frame semantics (Singley and Bennett, 2002; Deane, 2003). With the development of deep learning framework, Zhou and Huang (2019); Wang et al. (2021) generate problem text given equation templates and keywords, where the keywords are extracted from the golden MWP via heuristic rules. Their model is learned with Seq2seq in an end-to-end manner 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 unavailable in real scenarios. And this paper focuses on the MWP generation solely from equations without keywords which is more suitable for practical scenarios. Another work Liu et al. (2020) adopts the external commonsense based knowledge graph (CSKG) to generate topic relevant sentences, while this method only considers the cases of linear equations and needs annotated topics for each equation.

Data-to-text Generation: Data-to-text generation transforms structured data into descriptive texts (Siddharthan, 2001; Gatt and Krahmer, 2018). Recent works have brought great promising performance to several data-to-text generation tasks, e.g., Puduppully et al. (2019a,b); Gong et al. (2019a); Wiseman et al. (2017) focus on report generation; Chisholm et al. (2017); Lebret et al. (2016) target at biography generation; Zhao et al. (2020); Gao et al. (2020) generate texts from a set of RDF triples considering structural information. Previous works have also designed content selection and text planning models to determine what to say and how to say (Puduppully et al., 2019a; Perez-Beltrachini and Lapata, 2018).

Retrieval-based Generation: The methods similar to our instance-based generation are the skeleton-then-response frameworks which are popular in dialogue response generation (Cai et al., 2019; Wu et al., 2019; Yu and Jiang, 2021; Cai et al., 2020). These models usually treat the input text as a query and the similar query along with its response in databases is then retrieved with In-

formation Retrieval (IR) systems. However, they rely on difference between the input query and retrieved query to identify informative words in the retrieved response. Thus existing retrieval-based methods can not be employed in our task since it is meaningless to measure the similarity between equations.

3 Methodology

The overview of our DISK is depicted in Figure 3. Our model follows a three-stage procedure: Firstly, given the input equations and a text-domain vector which is extracted by the **Domain Summarizer**, the **Matching Model** retrieves a most similar MWP instance in database by jointly measuring equation-text matching score and domain-text matching score. Secondly, the **Sketch Provider** enriches the original representation of the instance to yield a refined instance sketch, it filters out excessive information considering domain constraint and helps the model to understand quantity relationship by applying Graph Neural Network (GNN) over the Quantity Cell Graph (QCG). Thirdly, **Text Generator** generates MWP text via utilizing both the math equation contextualized by QCG and the refined instance sketch based on an encoder-decoder architecture.

3.1 Domain Summarizer

The domain summarizer takes the MWP text $y = \{y_i\}_{i=1}^{L}$ with length L as input and its goal is to collect underlying domain information in the MWP text, which contributes to instance retrieving. To this end, inspired by (Huang et al., 2018; Keskar et al., 2019), we assume K latent domains are depicted by the MWP text with different importance β_i and the text-domain vector can be expressed as the weighted sum of K trainable domain vectors.

We start by encoding the MWP text into a sequence of vectors via a transformer block:

$$[\boldsymbol{h}_1; \boldsymbol{h}_2; ..., \boldsymbol{h}_L] = Encoder_P([\boldsymbol{y}_1; \boldsymbol{y}_2; ...; \boldsymbol{y}_L]) \tag{1}$$

where $[;]$ denotes concatenation operation. We denotes $[\boldsymbol{h}_1; \boldsymbol{h}_2; ...; \boldsymbol{h}_L]$ as $\boldsymbol{H} \in \mathbb{R}^{L \times d}$, d is the embedding size. A global attention is applied to the output of the transformer:

$$\boldsymbol{h}_a = \sum_{i=1}^{L} \alpha_i \boldsymbol{h}_i \tag{2}$$

where the attention weight $\alpha_i = \text{softmax}(\boldsymbol{h}_i \boldsymbol{W}^a \overline{\boldsymbol{h}})$, $\overline{\boldsymbol{h}} = \frac{1}{L} \sum_{i=1}^{L} \boldsymbol{h}_i$. A non-linear transformation is utilized to fuse the encoded MWP text into K domain variables:

$$\tilde{\boldsymbol{D}} = tanh(\boldsymbol{W}_1^D(\boldsymbol{H}\boldsymbol{W}_2^D + \boldsymbol{b}_2^D) + \boldsymbol{b}_1^D) \tag{3}$$

$\tilde{\boldsymbol{D}} \in \mathbb{R}^{K \times d}$ and parameters $\boldsymbol{W}_1^D \in \mathbb{R}^{K \times L}$, $\boldsymbol{W}_2^D \in \mathbb{R}^{d \times d}$. Each row vector in $\tilde{\boldsymbol{D}}$ corresponds to a different domain contained in MWP text. Such process can be treated as a soft clustering and we hope each domain expresses its unique aspect. Similar to Luxburg (2004), we employ an auxiliary loss function to restrict the derived K domain representation to be orthogonal with each other:

$$\mathcal{L}_D = ||\tilde{\boldsymbol{D}}\tilde{\boldsymbol{D}}^T - \boldsymbol{I}_{K \times K}|| \tag{4}$$

$\boldsymbol{I}_{K \times K}$ is an identity matrix. We then map $\tilde{\boldsymbol{D}}$ and $\boldsymbol{h}_a$ to a domain distribution with an attention mechanism (Bahdanau et al., 2014):

$$\beta_i = \text{Softmax}(\boldsymbol{v}^T tanh(\boldsymbol{H}^d \boldsymbol{h}_a + \boldsymbol{U}^d \tilde{\boldsymbol{D}}_{i,:})) \tag{5}$$

where $\boldsymbol{v}, \boldsymbol{H}^d, \boldsymbol{U}^d$ are learnable parameters. β_i indicates the domain distribution of the given MWP. Our model then learns a trainable domain embedding $\boldsymbol{E} \in \mathbb{R}^{K \times d}$ and uses $\{\beta_i\}_{i=1}^{K}$ to compute the text-domain vector over $\boldsymbol{E}$: $\boldsymbol{h}_d = \sum_{i=1}^{K} \beta_i \boldsymbol{E}_{i,:}$. Note that the domain summarizer only works during training process. During test, we enumerate each discrete domain vector in $\boldsymbol{E}$ to be fed into the matching model, which will be illustrated later.

3.2 Matching Model

The matching model aims to match one MWP instance from the training corpus which is most consistent with the given equation. Intuitively, incorporating the domain variable helps our model better recognize MWP text with similar domain grounding to the golden problem text, since it's difficult to infer from the equation only. Thus it's rational to combine the text-domain vector and the math equation to retrieve an additional instance.

Our matching model ranks all MWP texts from a pre-defined set $P = \{P_1, P_2, ..., P_{|P|}\}$, and returns the most consistent one with given equation-MWP pair (x, y), where P is prepared by uniformly sampling from the training corpus. The text-domain vector $\boldsymbol{h}_d$ and equation embedding $\{\boldsymbol{x}_i\}_{i=1}^{N}$ serve as input, where $\boldsymbol{x}_i$ is the sum of corresponding token embedding and type embedding, here type embedding is incorporated to distinguish quantities, numbers and operations in math equations. Each text $P_i = p_1^i p_2^i ... p_{|P_i|}^i$ is encoded

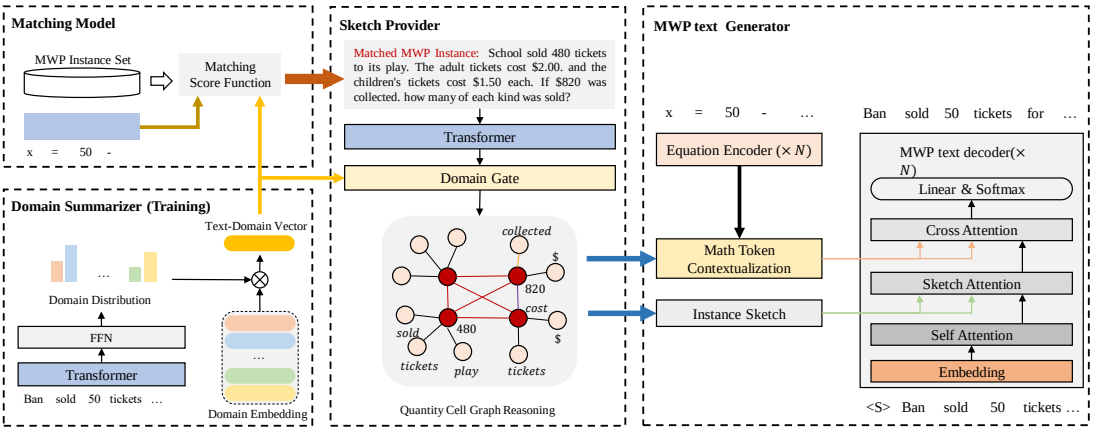

Figure 3: The diagram of proposed DISK. The text-domain vector is the summation of domain embeddings. First, the matching model predicts the most consistent MWP instance based on the equation tokens together with the text-domain vector. Then, the sketch provider learns to refine the retrieved MWP instance with a domain gate and the Quantity Cell Graph, it also contextualizes the equation representation to help the model understand the alignment between equations and MWP text. Finally, the generator consumes both the instance sketch and contextualized equation representation for generating.

into context representation $\left\{\boldsymbol{u}_j^i\right\}_{j=1}^{|P_i|}$ through transformer blocks (denoted as $Encoder_Q$). However, the dataset provides no supervision for the matching model, we then annotate the golden labels by ranking the BERTScore (Zhang* et al., 2020) between each candidate MWP text and the ground-truth MWP, i.e., BERTScore(P_i, y) $1 \leq i \leq |P|$, getting the top one P_{l^*} as the selected instance.

Equation to MWP Matching: Equation to MWP matching score is measured in token-level. Firstly, we encode $\{\boldsymbol{x}_i\}_{i=1}^N$ into $\boldsymbol{C} = \{\boldsymbol{c}_i\}_{i=1}^N$ via a transformer block, $[\boldsymbol{c}_1, \boldsymbol{c}_2, ..., \boldsymbol{c}_N] = Encoder_E([\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_N])$. A nonlinear function is then used to compute the correlation score between the j-th token in the equation and the k-th token of P_i:

$$\gamma_{i,j,k} = g_1(\boldsymbol{c}_j) \cdot g_2(\boldsymbol{u}_k^i) \tag{6}$$

where $g_1(\cdot), g_2(\cdot)$ are both multi-layer perceptrons (MLP). Next, we aggregate token level relevance to determine text-level score:

$$s_{em}(i) = \underset{j,k}{mean}\, \gamma_{i,j,k} \tag{7}$$

Domain to MWP Matching: We interact the text-domain vector $\boldsymbol{h}_d$ with the context representation of P_i: $\left\{\boldsymbol{u}_j^i\right\}_{j=1}^{|P_i|}$ to obtain the domain to MWP matching score. Similar to (2), we apply a global attention to calculate the summation of P_i, which we denote as $\boldsymbol{h}_p^i$. We then compute the domain vector to the ith MWP text relevance vector via a bilinear transformation:

$$\boldsymbol{r}_i = \boldsymbol{h}_d \boldsymbol{W}^r \boldsymbol{h}_p^i \tag{8}$$

$\boldsymbol{W}^r \in \mathbb{R}^{d \times d' \times d}$ is a parameter. Finally, we combine (7) and (8) to produce the normalized distribution over $|P|$ candidate MWP texts:

$$s(i) = \text{Softmax}(s_{em}(i) + \boldsymbol{w}^r \boldsymbol{r}_i)\ i \in [1, |P|] \tag{9}$$

where $\boldsymbol{w}^r \in \mathbb{R}^{d'}$ is a learnable parameter. With the label l^* annotated, we add a cross entropy loss to supervise the result of the matching model:

$$\mathcal{L}_M = -\log(s(l^*)) \tag{10}$$

For inference, the top one MWP text with the highest matching score is selected to be fed into the next sketch provider.

3.3 Sketch Provider

The sketch provider aims to generate the instance sketch by making soft modification to the MWP instance representation, since the generative model should capture underlying patterns contained in the instance, not simply copy the instance. We achieve this goal in two aspects: 1) we add a domain gate to refine tokens that have high relevance with the domain information, 2) we incorporate the Quantity Cell Graph to enable our model to better understand complex question scenarios while

maintaining those spans semantically aligned with equation tokens.

Firstly, for the encoded representation of P_{l^*} : $\{\boldsymbol{u}_i\}_{i=1}^{|P_{l^*}|}$ (which has been processed by $Encoder_Q$ in the matching model), we employ a soft gate controlled by the domain vector $\boldsymbol{h}_d$ to better flow the important context in the original matched text:

$$\begin{aligned} \boldsymbol{q}_i &= \sigma(\boldsymbol{W}^q\,[\boldsymbol{h}_d; \boldsymbol{u}_i]) \\ \boldsymbol{u}'_i &= \tanh(\boldsymbol{W}^Q \boldsymbol{u}_i) \odot \boldsymbol{q}_i \;\; 1 \le i \le |P_{l^*}| \end{aligned} \tag{11}$$

Quantity Cell Graph Constructing and Reasoning Targeting at better exploiting the retrieved MWP instance, we should enrich the instance encoding with quantity relationship information, as well as effectively guide the alignment between abstract equation tokens and MWP text tokens. Inspired by Zhang et al. (2020), we introduce the Quantity Cell Graph (QCG), whose nodes contain a subset of tokens in the the MWP text related to numerical values. As is shown in Fig 3, a Quantity Cell Graph is composed of a set of Quantity Cells (QC): $QCG = \{QC_1, QC_2, ..., QC_m\}$, where m denotes the number of quantities in the matched MWP instance P_{l^*}. Each cell QC_i can be expressed as $\{v_i^q\} \bigcup \left\{v_{i,1}^a, v_{i,2}^a, ...\right\}$, where v_i^q is the ith quantity token and $v_{i,1}^a, v_{i.2}^a, ...$ is the corresponding attributes or predicates. We resort to Dependency Tree[1] and Constituency Tree [2] to extract attributes related to each quantity token. Details can be found in Appendix B. We argue that the extracted tokens are salient properties related to quantities and show explicit alignment with the input equations. With the nodes in the Quantity Cell Graph mentioned above, we add an edge between two nodes if 1) they are both quantity nodes, 2) one is the quantity node and another is the attribute node belonging to it. Next, we initialize the node representation of the QCG by concatenating the corresponding output of $Encoder_Q$ and its POS tag embedding, which is denoted as $\boldsymbol{S}^0 = \{\boldsymbol{s}_k\}_{k=1}^{|G|}$, $|G|$ is the node number of the QCG. Graph Convolutional Network (Kipf and Welling, 2017) is applied to capture the dependencies between QCG nodes:

$$\boldsymbol{S}^{l+1} = ReLU(GCN(\boldsymbol{S}^l, \boldsymbol{A})) \tag{12}$$

where $\boldsymbol{S}^l$ is the node representation after the l-th layer, $\boldsymbol{A} \in \{0,1\}^{|G|\times|G|}$ is the adjacency matrix.

[1]https://demo.allennlp.org/dependency-parsing

[2]https://demo.allennlp.org/constituency-parsing

Graph2Text Augmentation After graph network reasoning, we need a fusion block to propagate the aggregated information of the QCG back to the text representation $\boldsymbol{U}' = \{\boldsymbol{u}'_i\}_{i=1}^{|P_{l^*}|}$. To locate the position of each node in the original matched text, we establish a binary matrix $\boldsymbol{M} \in \{0,1\}^{|P_{l^*}|\times|G|}$, where $\boldsymbol{M}_{ij} = 1$ if the i-th token in the MWP instance is the j-th node in the graph. As each column of $\boldsymbol{M}$ corresponds to one quantity node or attribute node in the QCG, we update $\boldsymbol{u}'_i$ with a GRU module if the i-th token participates in the QCG reasoning:

$$\tilde{\boldsymbol{U}} = GRU(\left[\boldsymbol{U}'; \boldsymbol{M}\boldsymbol{S}^{\mathcal{L}}\boldsymbol{W}^U\right]) \tag{13}$$

where $\boldsymbol{S}^{\mathcal{L}}$ is the output of the last GCN layer and $\boldsymbol{W}^U$ is a parameter matrix. $\tilde{\boldsymbol{U}}$ is treated as the output instance sketch.

Math Token Contextualization As mentioned before, the attribute words related to quantities are beneficial to help the model identify the soft alignment pattern between the math equation tokens and retrieved MWP instance. The encoded vector sequence of the input equations $\boldsymbol{C}$ attends to the QCG nodes to receive graph information:

$$\begin{aligned} \boldsymbol{G} &= \text{Softmax}(\boldsymbol{C}\boldsymbol{W}^G\boldsymbol{S}^{\mathcal{L}}) \\ \overline{\boldsymbol{C}} &= ReLU(\boldsymbol{G}\boldsymbol{S}^{\mathcal{L}}) \end{aligned} \tag{14}$$

We then calculate two update gate $\boldsymbol{f} = \sigma(\boldsymbol{W}^g\left[\boldsymbol{C}; \overline{\boldsymbol{C}}\right])$ and $\boldsymbol{g} = \sigma(\boldsymbol{W}^f\left[\boldsymbol{C}; \overline{\boldsymbol{C}}\right])$, which combines $\boldsymbol{C}$ and $\overline{\boldsymbol{C}}$ to obtain contextualized equation representation $\tilde{\boldsymbol{C}}$:

$$\tilde{\boldsymbol{C}} = \boldsymbol{g} \odot \boldsymbol{C} + (\boldsymbol{1} - \boldsymbol{g}) \odot tanh(\boldsymbol{W}^Z\left[\boldsymbol{C}; \boldsymbol{f} \odot \overline{\boldsymbol{C}}\right]) \tag{15}$$

3.4 MWP Generator

The MWP generator maps the math equation tokens $x_1x_2...x_N$ to the MWP text, we employ a transformer based encoder-decoder architecture. Here our encoder shares its parameters with $Encoder_E$ in matching model to capture common attention features among them. To enable the decoder to rewrite the domain-constrained instance sketch produced by the sketch provider in a fine-grained manner, we insert an extra sketch attention layer between the original self-attention layer and cross-attention layer. It aggregates details of the sketch by attending to output of sketch provider $\tilde{\boldsymbol{U}}$:

$$\boldsymbol{H}' = MultiHeadAttn(\mathbf{Q} : \boldsymbol{H}_p, \mathbf{K} : \tilde{\boldsymbol{U}}, \mathbf{V} : \tilde{\boldsymbol{U}}) \tag{16}$$

where $\boldsymbol{H}_p$ is the hidden state coming from the previous layer. Residual connection and layernorm is also added after the sketch attention. For the cross attention layer, the new representation $\tilde{\boldsymbol{C}}$ coming from Math Token Contextualization module is used as both the key and value. The hidden state of the last decoder layer is projected to vocabulary distribution and predicts the next token. The domain vector $\boldsymbol{h}_d$ is directly fed into the MWP text decoder and serves as the first input embedding (instead of the embedding of a start symbol <S>). The generation loss can then be modeled as:

$$\mathcal{L}_G = -\sum_{t=1}^{L} \log p(y_t|y_{<t}, \{x_i\}_{i=1}^{N}, \tilde{\boldsymbol{U}}, \boldsymbol{h}_d) \quad (17)$$

$y_{<t}$ is the tokens generated before the t-th step.

3.5 Model Training

For training, we combine the three loss terms mentioned above and the total loss becomes:

$$\mathcal{L}_{total} = \mathcal{L}_D + \mathcal{L}_M + \mathcal{L}_G \quad (18)$$

For inference, our model traverses over all K possible latent domain vectors in $\boldsymbol{E}$ and generates K candidate MWP texts $Y^1, Y^2, ..., Y^K$, among which the problem text with the maximum log likelihood score is chosen as the final output.

4 Experiments

4.1 Dataset

Our dataset is based on Dolphin18K (Huang et al., 2016) crawled from Yahoo. Since Huang et al. (2016) only releases a subset of Dolphin18K (3154 examples), which is insufficient for a modern data-driven generation model. So we reuse the python scripts provided by Huang et al. (2016) to crawl and collect extra data, then the size of dataset is extended to 14943 examples. We conduct some data preprocessing by deleting those equation-problem text pairs whose problem text length is longer than 45 tokens or less than 10 tokens. Finally 9643 samples are preserved. More detailed statistics of the dataset are listed in Appendix C.

4.2 Baselines

We compare DISK against the following models. 1) **Seq2seq** (Bahdanau et al., 2014): Seq2seq is first proposed for machine translation task. In this paper, we implement Seq2seq with attention mechanism and copy mechanism. 2) **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 tasks. 3) **DeepGCN** (Guo et al., 2019): Math equations can be converted into a pre-ordered expression tree and MWP generation can then be naturally modeled as graph-to-sequence learning. 4) **Transformer** (Vaswani et al., 2017): The state of-the-art model in several text generation tasks. 5) **DualCG** (Wei et al., 2019): In this paper we employ DualCG to integrate equation to MWP generation and MWP to equation solving in a unified framework. 6) We also compare our model with the vanilla **BART** (Lewis et al., 2020), which is a strong pretrained model using the standard seq2seq Transformer architecture. We fine-tune BART on our MWP dataset. Note we do not use retrieval-based baselines since they mostly require IR system, while it's unsuitable to treat math symbols as the query.

4.3 Automatic Evaluation

We compare different methods using BLEU (average of BLEU-1 and BLEU-2) (Papineni et al., 2002), ROUGE-L (Lin, 2004), METEOR (Banerjee and Lavie, 2005), BERTScore (Zhang* et al., 2020), which is an advanced evaluation metric for text generation based on contextual embedding, 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. We report the performance of all models in terms of automatic evaluation in Table 1.

We also conduct ablation studies and the results are also reported in Table 1. The setting is as follows: 1) w/o DG: We remove the domain gate in the Sketch Provider. 2) w/o QCG: We remove both the QCG reasoning and the Math Tokens Contextualization block. The encoded MWP instance, after being processed by the domain gate, is directly fed into the generator in this case. 3) w/o MTC: The model without Math Token Contextualization. 4) w/o CS: The model without the instance sketch, i.e., the whole Matching Model and Sketch Provider are removed and only the Domain Summarizer and the Generator are preserved.

It can be observed that 1) our proposed model significantly outperforms the strongest DualCG in BLEU, ROUGE-L and BERTScore, respectively.

Model	BLEU	ROUGE-L	BERTScore	METEOR	Dist1(%)	Dist2(%)	NR(%)
Seq2seq (Bahdanau et al., 2014)	2.59	20.25	82.98	18.51	14.56	34.99	47.60
SeqGAN (Yu et al., 2016)	2.62	19.22	82.56	17.63	12.96	30.02	44.00
DeepGCN (Guo et al., 2019)	3.04	20.94	83.07	19.48	16.81	45.17	49.21
Transformer (Vaswani et al., 2017)	3.14	21.84	83.81	20.26	12.94	43.51	44.84
DualCG (Wei et al., 2019)	3.60	21.43	83.99	20.63	15.47	46.01	40.97
BART$_{large}$ (Lewis et al., 2020)	4.15	22.26	**86.35**	22.30	12.77	46.76	43.47
DISK	**5.84**	**28.49**	85.01	**27.16**	**15.56**	**50.41**	52.62
w/o DG	5.33	27.36	84.90	25.33	10.74	35.62	49.37
w/o QCG	4.86	26.87	84.96	26.56	13.88	45.19	55.15
w/o MTC	5.14	28.08	84.62	26.16	13.63	43.90	55.51
w/o CS	4.59	26.96	84.27	25.32	12.53	43.67	**57.68**

Table 1: Automatic evaluation results of different models in MWP generation dataset. NR is the abbreviation for Number Recall. All results are averaged for five runs.

It yields higher results in most language quality metrics even when compared with BART$_{large}$. Besides, our model also improves the informativeness and diversity of generated MWP. 2) simply letting the MWP decoder attend to the retrieved MWP (w/o QCG) will degrade the performance by 16.78%, 5.69%, 0.06% in BLEU, ROUGE-L and BERTScore, respectively, which proves the Quantity Cell Graph can guide the generator to understand the quantity relationship and better exploit the retrieved MWP instance. It's notable that since BART maintains exact distributed word representation, it may show advantage in contextual embedding based metrics such as BERTScore. Moreover, as an intermediate result, we report the capacity of the matching model in Appendix F.

4.4 Performance on Different Types of Equations

Table 2 shows the performance on different subsets of the MWP generation dataset (divided by the number of variables). We can see the proposed method outperforms baselines by a large margin in all subsets. Intuitively, the more variables the equation contains, the more imperatively the generation process needs the guidance of instance sketch. It's easy to show our model obtains more absolute gain in More Than Three-VAR subset than One-AVR or Two-VAR ones.

4.5 Human Evaluation

To better measure the actual generation quality, we recruit three human annotators to judge the quality of different models, which includes four aspects listed as follows. 1) **Fluency**: Fluency mainly judges whether the problem text is fluent, i.e., whether some grammar errors occur in generated MWP. 2) **Coherence**: Coherence weights if the problem text is consistent in text-level. 3) **Solvability-1 (S1)**: As our target is a math word problem, we should pay attention to whether the problem text can be solved, i.e., in what percentage we can set up the same (or equivalent) equations and solve them according to the generated problem text. 4) **Solvability-2 (S2)**: 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.

We randomly select 100 generated MWP texts and score them in five grades. We then project the scores to 1∼5, where higher scores indicate better performance. Moreover, we assess the inter-annotator agreement by Cohen's kappa κ, which reflects the agreement between scores given by different raters. The averaged results are reported in Table 3. We can clearly see that the proposed model

		BLEU	ROUGE-L	BERTScore
One-VAR	**DualCG**	2.88	19.99	83.59
	Trans	2.18	19.43	83.45
	BART$_{large}$	3.16	19.83	85.94
	DISK	3.87	27.17	84.45
		BLEU	**ROUGE-L**	**BERTScore**
Two-VAR	**DualCG**	3.75	23.05	84.41
	Trans	3.97	22.82	84.52
	BART$_{large}$	4.77	25.26	87.00
	DISK	6.33	29.03	85.51
		BLEU	**ROUGE-L**	**BERTScore**
More Than Three-VAR	**DualCG**	2.00	17.74	84.33
	Trans	3.33	21.10	83.15
	BART$_{large}$	1.86	17.16	84.97
	DISK	4.59	25.63	84.26

Table 2: Performance on different subsets on our MWP generating dataset. Trans is short for Transformer.

	Fluency		Coherence		S1(%)	S2(%)
	score	κ	score	κ		
DISK	**4.00**	0.413	**4.08**	0.497	**36**	**56**
Seq2seq	3.78	0.256	3.48	0.483	23	34
SeqGAN	3.75	0.305	3.28	0.520	20	40
DeepGCN	3.61	0.295	3.55	0.494	29	52
Transformer	3.80	0.333	3.53	0.421	20	45
DualCG	3.88	0.346	3.66	0.455	28	53
BART$_{large}$	3.56	0.398	3.73	0.454	31	52

Table 3: Human evaluation results: comparison between the proposed model and baseline models.

Equ: $equ : 250 + 400 = x \; equ : 1625/x = y$ **MT**: 2 vehicles traveling different directions. same start point and time. one vehicle is 60 mph. the other is 55 mph. In how many hours will they be 500 miles apart. **Ours**: Two cars leave Denver traveling in opposite directions. One has a speed of 250 mph and the other airplane averages 400 mph. How many hours will the trip be 1625 miles apart. (*BLEU*: 15.47) **Seq2seq**: A <UNK> of deposit costs $ 400. 000 a t the end of the year. the total interest is $ 1625 . 00. What is the total cost of the total. **SeqGAN**: quotient of a certain number is 400. If the number of students in the first 250 is 400. What is the number. **DeepGCN**: The car ran a t an average speed of 400 km per hour faster than the other. If the speed of a 400 mi / h faster. What was the speed of the plane in miles per hour. **Trans**: planes went to school a t a speed for the trip takes 250 mph for 400 hours. How long will the plane travel in the trip. **DualCG**: Joe received 250 miles for 250 miles . and gas a trip of 250 miles per hour for $ 400 to the week. He drove 400 miles per hour faster . What was the average speed for the trip.
Equ: $equ : x + y = 35 \; equ : x/y = 2/5$ **MT**: total of 1600 people work for a company. The ratio of male to female employees is 3 : 5. How many more females than males are there in the company. **Ours**: The ratio of boys to girls in at a certain school is 5 : 2. If there are total 35 boys and girls. how many of each are there. (*BLEU*: 9.09) **Seq2seq**: The school art club is having a exhibit. The ratio of the school paintings are in two parts is 2 / 5 of the number. What is the number ? **SeqGAN**: A carpet is 3 times as many more than the other. The total value is 3. **DeepGCN**: The ratio of the larger of the two numbers is 35. The ratio of the smaller number of goals and the other is 5 / 2. What are the two numbers. **Trans**: Pat . 35 students and 5 questions. If the total of the students are seniors and 2 take both the total. how many of each. **DualCG**: The sum of two numbers is 35. The larger number is 2 less than the smaller number. Find the larger number.

Table 4: Two examples of math word problems generated by different models. Transformer is abbreviate to Trans. Equ and MT represent the equation and the matched MWP instance, respectively. Quantity-related attributes and predicates in the instance that are picked up and rewritten in the generated MWP are colored for better readability.

performs much better than other models, not only in fluency and coherence, but also in the solvability of generated math word problems.

4.6 Effectiveness of the Domain Gate and Quantity Cell Graph

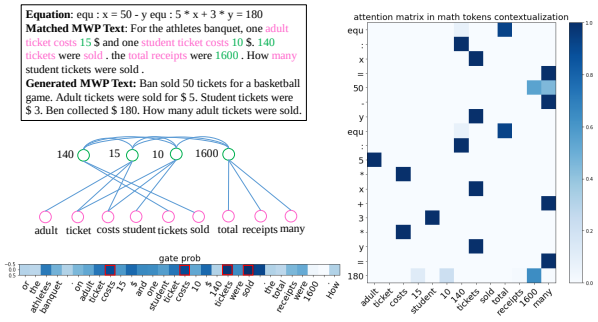

Figure 4: Visualization of a test case, which shows 1) the retrieved MWP text and the generated MWP text 2) the extracted QCG from the retrieved MWP text 3) the value of the domain gate on different tokens in (11) 4) the attention matrix between the input equation representation and the QCG, namely, $\boldsymbol{G}$ in (14)

We show the effectiveness of the domain-dominated soft gate and the Quantity Cell Graph Reasoning through qualitative analysis. Fig 4 presents a test case processed by our model. The heatmap in the left lower corner indicates the relevance of each token to the text-domain vector in the matched MWP text. The top-4 tokens are marked with the red box. Words that highlight the characteristic of one certain domain, e.g., "tickets", "sold", "cost"... are assigned with higher weight to be fused into the next block. The heatmap on the right hand side presents the probability that each equation token attends to the nodes in the QCG (after normalization). It is easy to show: number "5" is aligned with "adult", since "5" is the price of adult tickets; number "3" is aligned with "student", since "3" is the price of student tickets; both "x" and "y" are aligned with "tickets", since "x" and "y" both imply the number of tickets; "180" is aligned with "1600" as 1600 is the total receipts in the matched MWP instance and "180" also refers to the total sales in our generated text. It's reasonable to believe that the math token contextualization enhances the semantic alignment between math equations and the matched MWP instance.

4.7 Case Study

Table 9 shows two examples in the test dataset generated by different models. Additional examples can be found in Appendix G. We can observe that: 1) DISK gives consistent context in text-level while keeping readability, which verifies it's effective to assign a domain vector to each MWP text. 2) The generated MWP text expresses plausible attributes related to quantities by making an analogy with the matched instance. E.g., in case 2, the matched text discusses the ratio of male employees and female employees, while the MWP generated by our model says "the ratio of boys to girls". Besides, it's interesting that though in case 2, the output given by our system receives low BLEU score, it's still a logically reasonable and feasible MWP. So BLEU score may not be suitable for evaluating the performance of MWP generation. According to the above analysis, it is obvious that instance sketch provider improves the informativeness of the given equation via correctly understanding and exploiting the connections among QCG nodes.

5 Conclusion

We propose DISK, which introduces latent discrete domains for matching appropriate MWP instance and refines its representation. We also extract Quan-

tity Cell Graph to enhance the sketch-guided generator and help our model better understand math equations in real scenarios. Experimental results on the extended Dolphin18K Dataset show the superiority of our model.

Acknowledgement

This paper is supported by the National Science Foundation of China under Grant No.61876004 and 61936012, the National Key R&D Program of China under Grand No.2018AAA0102003.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv: Computation and Language*.

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: an automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization@ACL 2005, Ann Arbor, Michigan, USA, June 29, 2005*, pages 65–72. Association for Computational Linguistics.

Deng Cai, Yan Wang, Wei Bi, Zhaopeng Tu, Xiaojiang Liu, Wai Lam, and Shuming Shi. 2019. Skeleton-to-response: Dialogue generation guided by retrieval memory. In *NAACL*, pages 1219–1228. Association for Computational Linguistics.

Hengyi Cai, Hongshen Chen, Yonghao Song, Xiaofang Zhao, and Dawei Yin. 2020. Exemplar guided neural dialogue generation. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, pages 3601–3607. ijcai.org.

Shuang Chen, Jinpeng Wang, Xiaocheng Feng, Feng Jiang, Bing Qin, and Chin-Yew Lin. 2019. Enhancing neural data-to-text generation models with external background knowledge. In *EMNLP-IJCNLP*, pages 3022–3032. Association for Computational Linguistics.

Andrew Chisholm, Will Radford, and Ben Hachey. 2017. Learning to generate one-sentence biographies from wikidata. 1:633–642.

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

Hanning Gao, Lingfei Wu, Po Hu, and Fangli Xu. 2020. Rdf-to-text generation with graph-augmented structural neural encoders. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, pages 3030–3036. ijcai.org.

Albert Gatt and Emiel Krahmer. 2018. Survey of the state of the art in natural language generation: Core tasks, applications and evaluation. *J. Artif. Intell. Res.*, pages 65–170.

Heng Gong, Xiaocheng Feng, Bing Qin, and Ting Liu. 2019a. Table-to-text generation with effective hierarchical encoder on three dimensions (row, column and time). *arXiv: Computation and Language*.

Li Gong, Josep Crego, and Jean Senellart. 2019b. Enhanced transformer model for data-to-text generation. In *Proceedings of the 3rd Workshop on Neural Generation and Translation*, pages 148–156. Association for Computational Linguistics.

Zhijiang Guo, Yan Zhang, Zhiyang Teng, and Wei Lu. 2019. Densely connected graph convolutional networks for graph-to-sequence learning.

Chenyang Huang, Osmar Zaiane, Amine Trabelsi, and Nouha Dziri. 2018. Automatic dialogue generation with expressed emotions. pages 49–54.

Danqing Huang, Shuming Shi, Chin-Yew Lin, Jian Yin, and Wei-Ying Ma. 2016. How well do computers solve math word problems? large-scale dataset construction and evaluation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 887–896. Association for Computational Linguistics.

Nitish Keskar, Bryan McCann, Lav Varshney, Caiming Xiong, and Richard Socher. 2019. Ctrl: A conditional transformer language model for controllable generation.

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

Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *ICLR*. OpenReview.net.

Nate Kushman, Yoav Artzi, Luke Zettlemoyer, and Regina Barzilay. 2014. Learning to automatically solve algebra word problems. In *ACL*, pages 271–281. Association for Computational Linguistics.

Remi Lebret, David Grangier, and Michael Auli. 2016. Neural text generation from structured data with application to the biography domain. pages 1203–1213.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880. Association for Computational Linguistics.

Chinyew Lin. 2004. Rouge: A package for automatic evaluation of summaries. pages 74–81.

Tianqiao Liu, Qian Fang, Wenbiao Ding, Zhongqin Wu, and Zitao Liu. 2020. Mathematical word problem generation from commonsense knowledge graph and equations. *CoRR*, abs/2010.06196.

Ulrike Luxburg. 2004. A tutorial on spectral clustering. *Statistics and Computing*, 17:395–416.

Kishore Papineni, Salim Roukos, Todd Ward, and Weijing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. pages 311–318.

Laura Perez-Beltrachini and Mirella Lapata. 2018. Bootstrapping generators from noisy data. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1516–1527. Association for Computational Linguistics.

Oleksandr Polozov, Eleanor O'Rourke, Adam M. Smith, Luke Zettlemoyer, Sumit Gulwani, and Zoran Popović. 2015. Personalized mathematical word problem generation. In *IJCAI 2015*.

Ratish Puduppully, Li Dong, and Mirella Lapata. 2019a. Data-to-text generation with content selection and planning. *AAAI 2019*, 33(01):6908–6915.

Ratish Puduppully, Li Dong, and Mirella Lapata. 2019b. Data-to-text generation with entity modeling. *ACL 2019*, pages 2023–2035.

Shuming Shi, Yuehui Wang, Chin-Yew Lin, Xiaojiang Liu, and Yong Rui. 2015. Automatically solving number word problems by semantic parsing and reasoning. In *EMNLP*. Association for Computational Linguistics.

Advaith Siddharthan. 2001. Ehud reiter and robert dale. *Building Natural Language Generation Systems*. cambridge university press, 2000. $64.95/£37.50 (hardback), 234 pages. *Nat. Lang. Eng.*, (3):271–274.

Mark Singley and Randy Bennett. 2002. Item generation and beyond: Applications of schema theory to mathematics assessment.

Shyam Upadhyay and Ming-Wei Chang. 2017. Annotating derivations: A new evaluation strategy and dataset for algebra word problems. In *EACL*. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. pages 5998–6008.

Zichao Wang, Andrew Lan, and Richard Baraniuk. 2021. Math word problem generation with mathematical consistency and problem context constraints. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5986–5999. Association for Computational Linguistics.

Bolin Wei, Ge Li, Xin Xia, Zhiyi Fu, and Zhi Jin. 2019. Code generation as a dual task of code summarization. In *NeurIPS*, pages 6559–6569.

Sam Wiseman, Stuart M Shieber, and Alexander M Rush. 2017. Challenges in data-to-document generation. pages 2253–2263.

Yu Wu, Furu Wei, Shaohan Huang, Yunli Wang, Zhoujun Li, and Ming Zhou. 2019. Response generation by context-aware prototype editing. In *AAAI*, pages 7281–7288. AAAI Press.

Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2016. Seqgan: Sequence generative adversarial nets with policy gradient. *arXiv: Learning*.

Xiaojing Yu and Anxiao Jiang. 2021. Expanding, retrieving and infilling: Diversifying cross-domain question generation with flexible templates. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 3202–3212, Online. Association for Computational Linguistics.

Jipeng Zhang, Lei Wang, Roy Ka-Wei Lee, Yi Bin, Yan Wang, Jie Shao, and Ee-Peng Lim. 2020. Graph-to-tree learning for solving math word problems. In *ACL*, pages 3928–3937. Association for Computational Linguistics.

Tianyi Zhang*, Varsha Kishore*, Felix Wu*, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with bert. In *ICLR*. OpenReview.net.

Chao Zhao, Marilyn Walker, and Snigdha Chaturvedi. 2020. Bridging the structural gap between encoding and decoding for data-to-text generation. In *ACL*, pages 2481–2491, Online. Association for Computational Linguistics.

Qingyu Zhou and Danqing Huang. 2019. Towards generating math word problems from equations and topics. *INLG 2019*, pages 494–503.

A Task Definition

Our system maps a set of equations $\{E_1, E_2, ..., E_{|E|}\}$ to the MWP text: $\boldsymbol{y} = y_1 y_2 ... y_L$ which reflects logic of equations. Each equation consists of a sequence of math tokens: $E_k = x_1 x_2 ... x_{|E_k|}$, where $|E_k|$ is the length of k-th equation measured by the number of math tokens. Each math token belongs to one of the following three types: math operator (e.g., $+, -, *, \div, ...$), number (e.g., $0.2, 1, 30, ...$), variable (e.g., $x, y, z, ...$). L is the the length of problem text. Our objective is to estimate the following conditional probability depending on

equations and previously generated words $\boldsymbol{y}_{<t}$:

$$P(\boldsymbol{y}|\boldsymbol{x}) = \prod_{t=1}^{L} P(y_t|\boldsymbol{y}_{<t}, E_1, E_2, ...) \quad (19)$$

B Details on Constructing Quantity Cell Graph

In this section, we describe the rules for extracting quantity-related attributes as follows.

- We consider those tokens which are labeled as *Nouns* or *Verbs* and are within two hops starting from the quantity token in the dependency tree.
- We firstly traverse the nodes in the constituency tree starting from the root node in a depth-first manner, and backtracks when the visited node contains no more than F (F is a hyperparameter) leaf nodes. Such operation yields several subtrees and each token in the original text belongs and only belongs to one subtree. We detect the tokens belonging to the same subtree as the quantity token and are labeled as *Nouns* or *Verbs*.

C Dataset

Dataset Information: Table 5 provides our data statistics.

Motivation of Extending Dolphin18K: MWP solving datasets currently used include Alg514 (Kushman et al., 2014), Dolphin1878 (Shi et al., 2015), DRAW-1K (Upadhyay and Chang, 2017), Dolphin18K (Huang et al., 2016). Table 6 gives the statistic of these datasets. Alg514, Dolphin1878, DRAW-1K are all public available, while neural generation models for generative tasks are usually data-hungry thus equation-MWP pairs in those datasets are insufficient. Though Dolphin18K is a large-scale dataset, only a part of it (3154) are released. Moreover, existing datasets only include a certain type of MWP text, e.g., MWP text for linear equations, which restricts their practical application. We then reuse the python script provided by Huang et al. (2016) and acquire 14943 equation-MWP text pairs in total from Yahoo !. Generally, the public available datasets can be treated as the subset of our dataset. Next, we conduct data preprocess as follows, which is beneficial to train the generation model:

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

	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 5: Statistic of datasets

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

Table 6: Statistics of several existing MWP solving datasets. Avg EL, Avg Ops refer to average equation length and average numbers of operators in equations, respectively. $*$ indicates only 3154 equation-MWP pairs of Dolphin18K are available.

D Experimental Settings

The batch size for training is 32. The vocabulary size is set as 13k. The hidden size for both our model and baseline models is 256. We use 2 layers transformer block in our model and the baseline Transformer model. All multi-head attention layers are implemented with 8 heads. The embeddings are randomly initialized and are trained together with our model. The domain number is set as $K = 25$, however, the results for different values of K are also presented in this paper. The size of the candidate MWP set prepared for retrieving is $|P| = 500$. For extracting the QCG with constituency parser, the hyper-parameter is set as $F = 5$ and the graph network is stacked for 2 layers. To calculate the BERTScore, we use the tool released by the author on Github [3]. We train all models for 40 epoch. To prevent overfitting, we set the dropout probability as 0.2. We use the Adam optimizer (Kingma and Ba, 2014) with the learning rate $lr = 0.0005$ and momentum $\beta_1 = 0.9$, $\beta_2 = 0.999$.

E Impact of Different Domain Numbers

Fig 5 compares the fluctuation of BLEU and ROUGE-L when the number of domains changes

[3] https://github.com/Tiiiger/bert_score

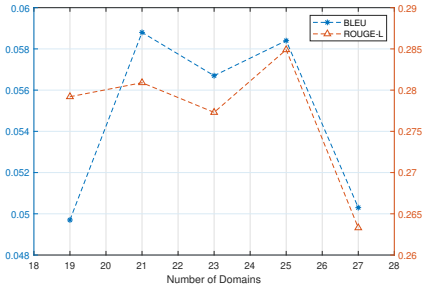

Figure 5: Performance with different domain numbers on the test dataset.

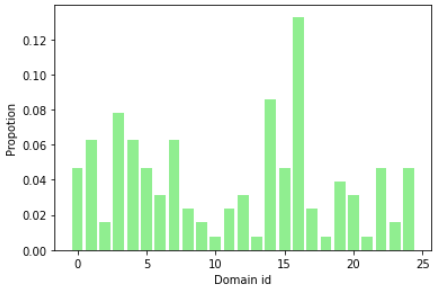

Figure 6: Proportion of test cases which is conditioned on each domain.

from 19 to 27. The proposed model receives consistent improvement compared against the baselines with different numbers of domains, while the peak value appears when $K = 21$ or $K = 25$. Even though K is set to 19 or 27, our model still exceeds baselines, which demonstrates its generalization capacity.

Moreover, one interesting problem is whether each domain plays a role during test. To this end, we investigate the percentage of output MWP text which is conditioned on each domain in the whole test set, the result is reported in Fig 6. We can find our model doesn't lead to "domain collapse", i.e., all cases are generated from the same domain, since the distribution of domains is generally balanced.

F Effectiveness of the Domain Gate and Quantity Cell Graph

We also conduct ablation study to measure our model's ability of understanding the retrieved MWP instance and properly exploiting the quantity relationship implied in it. As is shown in Table 7, we report the semantic matches in deep between the retrieved MWP instance and the generated MWP. When we discard the Quantity Cell Graph module or the Math Token Contextualization module, the relevance between the MWP instance and generated MWP both drop, which indicates the interaction between quantity-related attributes and enriching equation tokens with concrete scenarios are crucial for performance improvement.

	METEOR	BERTScore
Full Model	**24.09**	**85.01**
w/o QCG	22.70	84.79
w/o MTC	22.86	84.28

Table 7: Semantic similarity between the retrieved MWP instance and generated MWP from different models.

G Example Outputs

We show more cases produced by different models in Table 9 to demonstrate the performance of our proposed model.

H Error Analysis

We analyze and conclude the bad cases generated by our system and the typical problems are listed in Table 8. The first example shows *Logical Error*, where \$2000 is the capital, rather than the total amount of money after the period of deposit. There also exists similar errors in other instances such as the reversal of minuend and subtrahend. This shows the model ignores operation logic and underlying knowledge of constants. Consequently, auxiliary tasks such as quantity relation extraction and number sorting deserve to be considered. The second example shows *Missing Conditions*, since the total price, which is a necessary condition, is omitted in the generated MWP. This may be caused by the absence of sentence-level context planning, since one problem text usually consists of multiple sentences and each sentence conveys certain intention.

Table 8: Two examples to illustrate the drawbacks of the current method.

Equation: $equ: 2000 * (1 + 0.04)^5 = x \; equ: x - 2000 = y$	
Logical Errors:	Identify the interest earned for 5 years at the interest rate of 4% . \$ 2000 was deposited at the end of interest. How much of profit did he has.
Equation: $equ: x + y = 400 \; equ: 2 * x + 3 * y = 1050$	
Missing Conditions:	The attendance at a baseball game was 400 people. Student tickets cost \$ 2 and adult tickets cost \$ 3. The red rose theater sells tickets for \$ 4.50 and \$ 6.00. How many tickets of each type were sold.

Equ: $equ : x + y = 360\ equ : 0.18 * x + 0.09 * y = 360 * 0.17$
MT: one can drink is **15% orange juice**. another is **10% orange juice**. How many **liters** should be **mixed** in order to get 5 l that is 11% orange juice. How many liters of the 15% orange juice should be in the **mixture**.
Ours: **soybean meal** is **18% protein**. **cornmeal** is **9% protein**. How many **pounds** of each should be **mixed** together in order to get **360 lb mixture** that is 17% protein. (wh)
Seq2seq: kets to a junior high school play cost $ 3.50. and student tickets are $ 360. How many adult tickets were sold.
SeqGAN: The sum of two numbers is 360 . The sum of the numbers is 360 . What is the number.
DeepGCN: <UNK> has 360 pounds . cornmeal is 18% protein. If the total interest on at 0.09 is 360 protein. How many pounds of each should be mixed together in order to get 360 lb that is
Trans: soybean meal is 18% protein. cornmeal is 9% protein. How many grams of each should be mixed together in order to get 360 grams of the mixture.
DualCG: A chemist has a solution of 360 solution and a 20% solution of alcohol. How many liters of a solution that is the mixture that no solution to make a mixture that is 50% alcohol.

Equ: $equ : x + y = 35\ equ : x/y = 2/5$
MT: total of 1600 people work for **a company**. **The ratio of male to female employees** is 3 : 5. How many more **females than males** are there in the company.
Ours: The **ratio of boys to girls** in at **a certain school** is 5 : 2. If there are total 35 **boys and girls**. how many of each are there.
Seq2seq: The school art club is having a exhibit. The ratio of the school paintings are in two parts is 2 / 5 of the number. What is the number ?
SeqGAN: A carpet is 3 times as many more than the other. The total value is 3.
DeepGCN: The ratio of the larger of the two numbers is 35. The ratio of the smaller number of goals and the other is 5 / 2. What are the two numbers.
Trans: Pat . 35 students and 5 questions. If the total of the students are seniors and 2 take both the total. how many of each.
DualCG: The sum of two numbers is 35. The larger number is 2 less than the smaller number. Find the larger number.

Equ: $equ : 1/6 * x + 7 = 2/3 * x$
MT: **A truck's gas tank** is 1 / 4 full. After 15 **gallons of gas** are **added**, the tank is 7 / 8 full . What is the **gallon capacity** of the gas tank?
Ours: **Deandre's gas tank** is 2 / 7 full. After he **buys** 6 **gallons of gas**. it is 2 / 3 full. How many gallons can **Deandre's tank hold**.
Seq2seq: If a sum of 1 and 7 / 3 of the other digit. What is the value of x .
SeqGAN: <UNK> can paint a house in 6 hours . If the same time it takes 7 / 3 hours . How many days will it take to go 2 miles apart.
DeepGCN: The sum of the first three numbers is 7. the sum of the first number and the number is 7. the result is the same as when the result is one. Find the number
Trans: 1 / 6 of a number is 7 / 2 of the number. Find the number.
DualCG: If 1 / 6 of a number is 2 / 6. Find the number.

Equ: $equ : 250 + 400 = x\ equ : 1625/x = y$
MT: **2 vehicles traveling** different directions. same **start point** and time. one vehicle is **60 mph**. the other is **55 mph**. In how many hours will they be **500 miles apart**.
Ours: **Two cars leave Denver traveling** in opposite directions. One has a **speed of 250 mph** and the other airplane **averages 400 mph**. How many hours will the trip be **1625 miles apart**.
Seq2seq: A <UNK> of deposit costs $ 400. 000 a t the end of the year. the total interest is $ 1625 . 00. What is the total cost of the total.
SeqGAN: quotient of a certain number is 400. If the number of students in the first 250 is 400. What is the number.
DeepGCN: The car ran a t an average speed of 400 km per hour faster than the other. If the speed of a 400 mi / h faster. What was the speed of the plane in miles per hour.
Trans: planes went to school a t a speed for the trip takes 250 mph for 400 hours. How long will the plane travel in the trip.
DualCG: Joe received 250 miles for 250 miles . and gas a trip of 250 miles per hour for $ 400 to the week. He drove 400 miles per hour faster . What was the average speed for the trip.

Table 9: Four examples of math word problems generated by different models. Transformer is abbreviate to Trans. Equ and MT represents the equation and the matched MWP instance, respectively. Quantity-related attributes and predicates in the instance that are picked up and rewritten in the generated MWP are colored for better readability.