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.

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, “\(x = 6 \times y\) 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

Equations: \(x = 6 \times y\); \((x + y) \times 3 = 147\)

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: \((1 - 1/3 - 9/20) \times x = 245\)

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.
School sold 480 tickets to its play. The adult tickets cost $2.00, and the children’s tickets cost $1.50 each. If $820 was collected, how many of each kind was sold.

Equation 1:
\[ x = \frac{-2}{(10 / 60)} \]

Problem 1:
If a guy drove 2 miles in 10 minutes how many miles per hour was he going.

Matched MWP Instance:
Martin drove 140 miles with 8 3/4 gallons of gas. How many miles each gallon can arrive.

Equation 2:
\[ 4 \times x + 3 \times y = 5.60 \]
\[ 2 \times x + 3 \times y = 4.60 \]

Problem 2:
What was the price of each pencil and each pen. Crevs bought 4 pencils and 3 pens for $5.60. Miss Houston bought 2 pencils and 3 pens of the same kind for $4.60.

Matched MWP Instance:
School sold 480 tickets to its play. The adult tickets cost $2.00, and the children’s tickets cost $1.50 each. If $820 was collected, how many of each kind was sold.

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.

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 (Singly 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 $\beta_i$ 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:

$$[h_1; h_2; \ldots; h_L] = Encoder_F([y_1; y_2; \ldots; y_L])$$  \hspace{1cm} (1)

where $[:]$ denotes concatenation operation. We denote $[h_1; h_2; \ldots; h_L]$ as $H \in \mathbb{R}^{L \times d}$, $d$ is the embedding size. A global attention is applied to the output of the transformer:

$$h_a = \sum_{i=1}^L \alpha_i h_i$$  \hspace{1cm} (2)

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

$$\tilde{D} = \tanh(W_1^D (HW_2^D + b_2^D) + b_1^D)$$  \hspace{1cm} (3)

$\tilde{D} \in \mathbb{R}^{K \times d}$ and parameters $W_1^D \in \mathbb{R}^{K \times L}$, $W_2^D \in \mathbb{R}^{d \times K}$. Each row vector in $\tilde{D}$ 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:

$$\mathcal{L}_D = ||\tilde{D} \tilde{D}^T - I_{K \times K}||$$  \hspace{1cm} (4)

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

$$\beta_i = \text{Softmax}(v^T \tanh(H^d h_a + U^d \tilde{D}_i))$$  \hspace{1cm} (5)

where $v, H^d, U^d$ are learnable parameters. $\beta_i$ indicates the domain distribution of the given MWP. Our model then learns a trainable domain embedding $E \in \mathbb{R}^{K \times d}$ and uses $\{\beta_i\}_{i=1}^K$ to compute the text-domain vector over $E$: $h_d = \sum_{i=1}^K \beta_i E_i$. Note that the domain summarizer only works during training process. During test, we enumerate each discrete domain vector in $E$ 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 $P = \{P_1, P_2, \ldots, P_{|P|}\}$, 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 $h_d$ and equation embedding $\{x_i\}_{i=1}^N$ serve as input, where $x_i$ 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 $P_i = p_1^i p_2^i \ldots p_{|P_i|}^i$ 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.

Equation to MWP Matching: Equation to MWP matching score is measured in token-level. Firstly, we encode \( \{x_i\}_{i=1}^N \) into \( C = \{c_i\}_{i=1}^N \) via a transformer block. \( [c_1, c_2, \ldots, c_N] = \text{Encoder}_E([x_1, x_2, \ldots, x_N]) \). 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 \( P_i \):

\[
\gamma_{i,j,k} = g_1(c_j) \cdot g_2(u_k^i)
\]

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

\[
s_{em}(i) = \text{mean}_{j,k} \gamma_{i,j,k}
\]

Domain to MWP Matching: We interact the text-domain vector \( h_d \) with the context representation of \( P_i \): \( \{u_j^i\}_{j=1}^{P_i} \) 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 \( h_p^i \). We then compute the domain vector to the \( i \)-th MWP text relevance vector via a bilinear transformation:

\[
r_i = h_d W^r h_p^i
\]

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
where $S$ is the quantity node and another is the attribute node belonging to it. Next, we initialize the node number of the QCG, Graph Con- dencing with quantity relationship information, as mentioned above, we add an edge between quantities and show explicit alignment with the input instance is the matched MWP instance. The encoded vector representation of the QCG by concatenating 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. 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 $S_0 = \{s_k\}_{k=1}^{|G|}$, where $|G|$ is the node number of the QCG. Graph Convolutional Network (Kipf and Welling, 2017) is applied to capture the dependencies between QCG nodes:

\begin{equation}
S_i^{l+1} = \text{ReLU}(GCN(S_i^{l}, A)) \tag{12}
\end{equation}

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

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

\begin{equation}
\tilde{U} = GRU([U'; MS'CU]) \tag{13}
\end{equation}

where $S'$ is the output of the last GCN layer and $U'$ is a parameter matrix. $\tilde{U}$ 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 $C$ attends to the QCG nodes to receive graph information:

\begin{align}
G &= \text{Softmax}(CW^G S^C) \\
\overline{C} &= \text{ReLU}(\overline{GS}^C) \tag{14}
\end{align}

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

\begin{equation}
\tilde{C} = g \odot C + (1 - g) \odot \text{tanh}(W^Z [C; f \odot C]) \tag{15}
\end{equation}

3.4 MWP Generator

The MWP generator maps the math equation tokens $x_1, x_2, ..., x_N$ 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 $\tilde{U}$:

\begin{equation}
H' = \text{MultiHeadAttn}(Q : H_p, K : \tilde{U}, V : \tilde{V}) \tag{16}
\end{equation}

\footnote{\url{https://demo.allennlp.org/dependency-parsing}}

\footnote{\url{https://demo.allennlp.org/constituency-parsing}}
where $H_p$ is the hidden state coming from the previous layer. Residual connection and layer-norm is also added after the sketch attention. For the cross attention layer, the new representation $\hat{C}$ 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 $h_d$ 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:

$$L_G = -\sum_{t=1}^{L} \log p(y_t|y_{<t}, \{x_i\}_{i=1}^{N}, \hat{U}, h_d)$$  \hspace{1cm} (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:

$$L_{total} = L_D + L_M + L_G$$  \hspace{1cm} (18)$$

For inference, our model traverses over all $K$ possible latent domain vectors in $E$ 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.
It yields higher results in most language quality metrics even when compared with BART\textsubscript{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\sim 5, where higher scores indicate better performance. Moreover, we assess the inter-annotator agreement by Cohen’s kappa $\kappa$, 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
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

![Image of visualization](image)

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, $G$ 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 instances 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


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*.


Our system maps a set of equations \( \{ E_1, E_2, ..., E_k \} \) to the MWP text: \( y = y_1 y_2 ... y_L \), which reflects logic of equations. Each equation consists of a sequence of math tokens: \( E_k = x_1 x_2 ... x_l | E_k | \), 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., +, -, *, ÷, ...), 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 $y_{<t}$:

$$P(y|x) = \prod_{t=1}^{L} P(y_t|y_{<t}, E_1, E_2, ...)$$  \hspace{1cm} (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 backtrack 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 pre-process as follows, which is beneficial to train the generation model:

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

- We manually correct the wrong spelling words in MWP text.

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

*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
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.

<table>
<thead>
<tr>
<th></th>
<th>METEOR</th>
<th>BERTScore</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full Model</td>
<td>24.09</td>
<td>85.01</td>
</tr>
<tr>
<td>w/o QCG</td>
<td>22.70</td>
<td>84.79</td>
</tr>
<tr>
<td>w/o MTC</td>
<td>22.86</td>
<td>84.28</td>
</tr>
</tbody>
</table>

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.

**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.

**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.

**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.

**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.

**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 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.