Towards Generating Math Word Problems from Equations and Topics

Qingyu Zhou[*]
Harbin Institute of Technology
qyzhou@hit.edu.cn

Danqing Huang[*]
Microsoft Research
dahua@microsoft.com

Abstract

A math word problem is a narrative with a specific topic that provides clues to the correct equation with numerical quantities and variables therein. In this paper, we focus on the task of generating math word problems. Previous works are mainly template-based with pre-defined rules. We propose a novel neural network model to generate math word problems from the given equations and topics. First, we design a fusion mechanism to incorporate the information of both equations and topics. Second, an entity-enforced loss is introduced to ensure the relevance between the generated math problem and the equation. Automatic evaluation results show that the proposed model significantly outperforms the baseline models. In human evaluations, the math word problems generated by our model are rated as being more relevant (in terms of solvability of the given equations and relevance to topics) and natural (i.e., grammaticality, fluency) than the baseline models.

1 Introduction

A math word problem is a narrative which describes a story under a specific topic. Moreover, it provides clues to the correct equation interpreting mathematical relations of numerical quantities and variables. The two example problems in Table 1 belong to two different topics (ticket selling, land purchase) respectively. Meanwhile they share the same equation template interpreting the underlying mathematical relations between numbers and variables. To generate a math word problem, a system needs to produce a *topic*-specific story while maintaining the underlying *equation*.

There is a surge of interest in automatic math word problem generation (K. and Elliot, 2002; Deane and Sheehan, 2013; Polozov et al., 2015; Koncel-Kedziorski et al., 2016). Previous attempts are mainly based on templates. Polozov et al. (2015) consider several components (e.g., event graph construction, surface text realization), each with manual defined templates and rules. Koncel-Kedziorski et al. (2016) generate math problems by revising existing problems into a new topic. They use a problem as the verbal template and simply replace nouns and verbs with suitable words from the new topic. The generation of template-based systems is based directly on existing items with high coherence. However, they have clear limitations. As templates are fixed, the possible outputs are limited to follow template patterns without too many grammatical and lexical options. Additionally, they require manual effort to construct domain-specific templates.

Recently, neural network approaches to automatic generation of questions (Du et al., 2017; Zhou et al., 2017) and stories (Fan et al., 2018) have shown promising results. Despite their success, they cannot be directly applied to math word problem generation, since generation of math word problems need to maintain the underlying mathematical operations between quantities and variables, while at the same time ensuring the relevance of the output problem and a given topic.

In this paper, we propose a novel neural network model for ***Ma***th word Problem ***Gen***eration from ***E***quations and ***T***opics (**MAGNET**). The proposed model consists of three main components: an equation encoder, a topic encoder and a math problem decoder. The equation encoder is implemented with a bidirectional recurrent neural networks (RNN) which takes the equation tokens as input and produces a sequence of hidden vectors. The topic encoder maps the given topic words into continuous word representations. The decoder is a single directional RNN with dual-attention mechanism, which can dynamically extract information

[*]Equal contribution.

Equation Template: $x + y = [num0]$ $[num1] * x + [num2] * y = [num3]$
Problem 1: Tickets to local movie were sold at \$4.00 for adults and \$2.50 for students. If 267 tickets were sold for a total of \$1042.50, how many adult tickets were sold? **Topic:** ticket selling **Equation:** $x + y = 267, 4 * x + 2.5 * y = 1042.5$
Problem 2: A farmer bought 100 acres of land, part at \$300 an acre and part at \$450, paying for the whole \$42,200. How much land was there in each part? **Topic:** land purchase **Equation:** $x + y = 100, 370 * x + 450 * y = 42200$

Table 1: Math word problems of the same equation template but with different topics.

from equations and topic words. To leverage both the equation and topic information, we design an equation-topic fusion mechanism to enable the decoder to choose which information to use. Furthermore, to ensure that the generated math word problem is highly related to the given equations, we introduce a novel entity-enforced loss function which considers the correspondence between variables in the given equations and entities in the output problem.

Large-scale annotated math problem datasets play a crucial role in developing neural math problem generation systems. We propose to adapt Dolphin18K (Huang et al., 2016) as the training, development and test sets, since it is one of the current largest math problem datasets with diverse problem types. It contains 18,460 elementary math problems from Yahoo! Answers[1], with annotation of equations and answers.

Extensive experiments are conducted on the Dolphin18K dataset. We first propose three baseline methods: 1) a retrieve-based model that find the closest math problems in the training set; 2) a sequence-to-sequence model which takes only the equation as input (Equ2Math); 3) a neural decoder model conditioned on topic words (Topic2Math). We use three commonly used automatic evaluation metrics in recent text generation works, i.e., BLEU (Papineni et al., 2002), ROUGE (Lin, 2004) and METEOR (Denkowski and Lavie, 2014). Evaluation results on all three metrics show that our MAGNET model outperforms the baseline methods. To further examine the quality of generated math word problems, we also conduct human evaluations. Human evaluation results show that our MAGNET model performs better than the baseline systems on three aspects, i.e., 1) solvability to the given equation; 2) relevance to the given topic and 3) grammaticality and fluency of language.

Our contributions are three-folds:

1. We propose a novel end-to-end neural network model MAGNET to generate math word problems based on given equations and topics.
2. We introduce an Equation-Topic Fusion mechanism which helps the decoder incorporate both the information from the equation and the topic.
3. We design an entity-enforced loss function to improve the relevance between the generated math word problem and the given equations.

2 Related Work

Automatic question generation from text aims to generate questions taking text as input, which has the potential value of education purpose (Heilman, 2011). Previous question generation works focus on generating natural language questions from a given piece of text. Heilman (2011) employs a syntactic parser to parse the input text into a tree and extract answer candidates. Then a rule-based system transforms the tree into the corresponding question. Recently, generative neural network methods are also applied to this area since large-scale manually annotated passage-question pairs become available. Du et al. (2017) and Zhou et al. (2017) propose to use SQuAD (Rajpurkar et al., 2016) question answering dataset as the training data of question generation. In SQuAD dataset, the given passage is a piece of text from Wikipedia and the answer is a sub-span in it. Du et al. (2017) use a sequence-to-sequence model on the passage-question pair to generate questions. Their model takes the passage text as input to generate a question from it. Different from Du et al. (2017), they add the answer position to the model input as BIO

[1]https://answers.yahoo.com/

tagging features. However, these methods cannot directly applied to math word problem generation.

There are previous approaches specifically targeting math problem generation. Most of them are template based, such as natural language schemas (K. and Elliot, 2002) and semantic frames of conceptual structures (Deane and Sheehan, 2013). Polozov et al. (2015) propose a pipeline including equation generation, plot generation and surface text realization, which requires manually defined ontology and templates. These approaches are ensured to maintain highly-coherent story, but with the manual cost of template construction, which is difficult to extend to more domains. Recently, Koncel-Kedziorski et al. (2016) propose a rewrite-based approach. They generate new problems by simply replacing noun phrases and verbs in the existing math problems with words in the target topic. However, they do not consider global optimization of the whole problem that results in semantic incoherence.

Math problem solving, which can be formatted as learning the mapping from math problem to equations, is also related to our work. In this paper, we adapt a math problem dataset Dolphin18K for development. Dolphin18K (Huang et al., 2016) is constructed from Yahoo! Answer containing over 18,000 math problems. Previous to that, there are several datasets with size less than 2,000, such as VERB-375 (Hosseini et al., 2014), ALG514 (Kushman et al., 2014) and Dolphin1878 (Shi et al., 2015).

3 Problem Statement

Given an equation template and a target topic, our goal is to generate a math word problem in natural language. In this section, we first define the equation template and the topic, and then give the formal introduction of our task.

3.1 Equation Template

Equation template, introduced in Kushman et al. (2014), is a unique form of an equation system. For example, given an equation system as follows:

$$x + y = 20; x - 4 = y$$

We replace the numbers with tokens and generalize the equations as the following template:

$$x + y = [num0]; x - [num1] = y$$

Equation is a solution for a specific math problem, while an equation template can correspond to several math problems. Therefore, an equation template can be seen as an abstraction of a set of equations.

3.2 Topic

As pointed out in Koncel-Kedziorski et al. (2016), math problems are coherent stories with different topics (e.g., ticket selling or land purchase). In one math problem, there are words that act as topic indicators. For the problems in Table 1, the corresponding topic indicators are:
Problem 1: {*tickets, movie, adults, students, sold*}
Problem 2: {*farmer, bought, dollar, land, pay*}

Therefore, we extract the keywords of a math problem as its topic words for representing the topic. The details of topic words extraction will be described in Section 3.4.

3.3 Math Problem Generation

Now we can formally define the task of math word problem generation. Given an equation template E and a set of topic words T as input, the goal is to generate a math word problem P, satisfying:
(1) P is a piece of natural language text whose topic is T;
(2) P maintains the mathematical operations between numerical quantities and variables in the equation template E.

3.4 Dataset Creation

We create the math word problem generation dataset based on the Dolphin18K (Huang et al., 2016) dataset. Specifically, we construct (E, T, P) triple where E is an equation, T is a set of topic words, and P is the corresponding math word problem. In the Dolphin18K dataset, the equation E and math word problem P are given. Therefore, we need to extract the topic words from the text of P.

There are previous studies on the task of topic word extraction, such as simple counting of word frequency and LDA topic model (Blei et al., 2003). We practically observe that the TF-IDF method is effective which satisfies our needs. We calculate the scores of the words as follows:

$$tf_{ij} = \frac{n_{ij}}{\sum_k n_{kj}} \tag{1}$$

$$idf_i = \log \frac{|P|}{|j : t_i \in P_j| + 1} \tag{2}$$

$$score_{ij} = tf_{ij} * idf_i \tag{3}$$

where tf_{ij} is the term frequency of word i in problem P_j, and idf_i is the inverse document frequency of word i. We sort the score of each word i in P_j, and keep the **top** n_{tp} words as the problem's topic words.

4 MAGNET

As shown in Figure 1, our MAGNET model consists of three main parts, namely, the topic encoder, the equation encoder and the math word problem decoder. The topic encoder and equation decoder are used to map topic words and equations to continuous vectors. The decoder is a single directional recurrent neural network equipped with dual-attention mechanism which leverages by the equation-topic fusion mechanism.

4.1 Topic Encoder

The input topic T contains a set of keywords $\{t_1, t_2, \ldots, t_{n_{tp}}\}$. Considering the fact that these topic words do not have sequential or temporal relationships, we represent them as a set of word embeddings $\{tp_1, tp_2, \ldots, tp_{n_{tp}}\}$ as shown in the upper-left part of Figure 1. Specifically, the topic encoder is a lookup table which maps input topic words to the corresponding real-valued vectors.

4.2 Equation Encoder

The encoder is implemented as a single-layer bidirectional GRU (Cho et al., 2014) (BiGRU). We concatenate all the equations together with a special delimiter "," (indicates the end of an equation). The BiGRU reads the input equation tokens one-by-one, producing a sequence of hidden states $h_i = [\overrightarrow{h}_i; \overleftarrow{h}_i]$ with:

$$\overrightarrow{h}_i = \text{GRU}(x_i, \overrightarrow{h}_{i-1}) \tag{4}$$

$$\overleftarrow{h}_i = \text{GRU}(x_i, \overleftarrow{h}_{i+1}) \tag{5}$$

The initial states of the BiGRU are set to zero vectors, i.e., $\overrightarrow{h}_1 = 0$ and $\overleftarrow{h}_n = 0$.

4.3 Math Word Problem Decoder

At each time-step t, the decoder GRU holds its previous hidden state s_{t-1}, the embedding of previous output word y_{t-1} and the previous context vector c_{t-1}. With these previous states, the decoder GRU updates its states as given by Equation 6. To initialize the GRU hidden state, we use a linear layer with the last backward encoder hidden state $\overleftarrow{h}_1$ of equation as input:

$$s_t = \text{GRU}(w_{t-1}, c_{t-1}, s_{t-1}) \tag{6}$$

$$s_0 = \tanh(\mathbf{W}_d \overleftarrow{h}_1 + b) \tag{7}$$

Then the decoder first generates a readout state r_t and passes it through a maxout hidden layer (Goodfellow et al., 2013) to predict the next word with a softmax layer over the output vocabulary.

$$r_t = \mathbf{W}_r w_{t-1} + \mathbf{U}_r c_t + \mathbf{V}_r s_t \tag{8}$$

$$r'_t = [\max\{r_{t,2j-1}, r_{t,2j}\}]^\top_{j=1,\ldots,d} \tag{9}$$

$$p(y_t|y_{<t}) = \text{softmax}(\mathbf{W}_o r'_t) \tag{10}$$

where $\mathbf{W}_r$, $\mathbf{U}_r$, $\mathbf{V}_r$ and $\mathbf{W}_o$ are weight matrices. w_{t-1} is the word embedding of the previously generated word y_{t-1}. The readout state r_t is a $2d$-dimensional vector, and the maxout layer (Equation 9) picks the max value for every two numbers in r_t and produces a d-dimensional maxout vector r'_t. We then apply a linear transformation on r'_t to get a target vocabulary size vector and predict the next word y_t with the softmax operation.

4.4 Equation-Topic Fusion

To incorporate both the information of equation and topic, we propose the Equation-Topic fusion mechanism. Intuitively, the Equation-Topic Fusion mechanism enables the decoder to pay different portions of attention to the equation templates and topic words. For instance, when the decoder is generating descriptive words about the story, it should pay more attention to the topic words. Vice versa, the decoder should pay more attention to the equation if it is generating numbers or variables in the equation. In detail, the context vector c_t in Equation 6 and 8 is a fused vector of equation and topic. We employ two attention modules to produce the corresponding context vectors of equa-

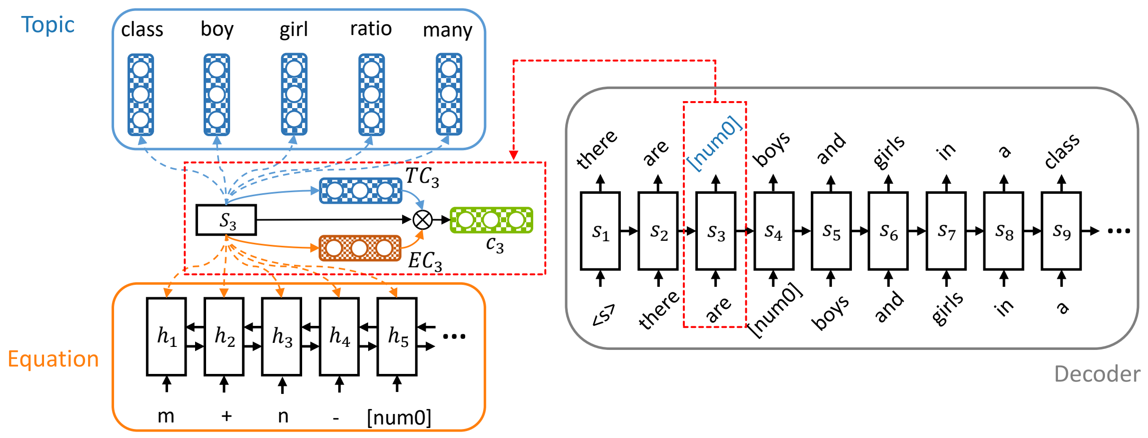

Figure 1: The overview diagram of MAGNET. For simplicity, we omit some units and connections. This figure shows the detail that the decoder is generating the third word by fusing the information from both the input equation and topic words.

tion and topic :

$$e_{t,i} = v_a^\top \tanh(\mathbf{W}_a s_t + \mathbf{U}_a \mathrm{repr}(\cdot)_i) \quad (11)$$

$$\alpha_{t,i} = \frac{\exp(e_{t,i})}{\sum_{i=1}^{n} \exp(e_{t,i})} \quad (12)$$

$$c(\cdot)_t = \sum_{i=1}^{n} \alpha_{t,i} \mathrm{repr}(\cdot)_i \quad (13)$$

where $\mathrm{repr}(\cdot)_i$ represents the vector of encoded equation tokens or topic words, which can be h_i or tp_i. The $v_a^\top$, $\mathbf{W}_a$ and $\mathbf{U}_a$ are learnable parameters. Since the equation and topic information are of different types, we use two sets of these parameters for equation and topic attention modules.

We represent the equation context vector $c(\text{equation})_t$ and topic context vector $c(\text{topic})_t$ as EC_t and TC_t respectively. To fuse EC_t and TC_t together, we predict a fusion coefficient g_t using an MLP:

$$g_t = \mathrm{sigmoid}(\mathbf{W}_f s_t + b) \quad (14)$$

$$c_t = g_t \cdot EC_t + (1 - g_t) \cdot TC_t, \quad (15)$$

where g_t is the fusion gate. Therefore, the context vector c_t is the combination of equation template and topic which is determined by the current decoding state s_t.

4.5 Entity-Enforced Loss

As we mention before, the generated math problems should be highly relevant to the given equation template. The *entities* in the generated math problem should correspond to the *variables* in the equations (e.g., m, n, $[num0]$). To ensure high relevance of equation template and the generated math problem, we propose an entity-enforced loss:

$$acc_e = \sum_{t=1}^{L} g_t \alpha_{t,e} \quad (16)$$

$$\mathcal{L}_e = \sum_{\forall e \in \text{equation}} \mathrm{ReLU}(1 - acc_e) \quad (17)$$

where L is the length of output problem, and ReLU is rectifier function defined as:

$$\mathrm{ReLU}(x) = \max(0, x) \quad (18)$$

The intuition behind the entity-enforced loss is that the model needs to attend to the entities in the given equations. In Equation 16, we accumulate the attention scores of variables in the equation for all the decoding time steps. Then a ReLU function is applied on $(1 - acc_e)$ to ensure that the entity e is attended for at least one time during decoding.

4.6 Objective Function

Given a training dataset with n equation-topic-question triples $\mathcal{D} = \{(E^{(1)}, T^{(1)}, P^{(1)}), \ldots, (E^{(n)}, T^{(n)}, P^{(n)})\}$, the training objective is to minimize the negative log likelihood loss $\mathcal{L}$ with respect to the model parameter θ:

$$\mathcal{L} = -\sum_{i}^{n} \log p(P^i | E^i, T^i; \theta) + \lambda \mathcal{L}_e \quad (19)$$

where λ is a hyper-parameter that controls the contribution of entity-enforced in the loss.

5 Experiment

In this section, we evaluate our model with both automatic and human evaluations.

5.1 Datasets

We conduct our experiments on the Dolphin18K dataset[2]. Since we need equation templates as input to generate math problems, we use its subset with equation annotation, which sums up to 10,644 problems with 5,738 equation templates. The average sentence length of a problem is 2.70. The average number of words in a problem is 32.72. We train on 8,515 examples and evaluate on 2,129 test examples, following the split setting in Huang et al. (2016).

As pre-processing, we obtain equation template and topic words for each problem as their input. We extract at most $n_{tp} = 10$ words with highest TF-IDF scores as the topic words in our experiments. According to the statistic, the average number of extracted topic words are 7.7 and 7.5 in the training and testing datasets respectively.

5.2 Baselines

We provide three baselines of math word problem generation, considering the input of equation template and topic words respectively[3].

KNN finds the closest problem in the training set given the input topic words. It first narrows down training problems to those with the same input equation template. Then a TF-IDF vector for topic words was created and KNN is applied to retrieve the nearest training problem.

Topic2Math generates math problems only given the input of topic words. Topics words are encoded by Topic Encoder component described in Section 4.1.

Equ2Math generates math problems only given the input of equation template. Equation template is encoded by Equation Encoder component described in Section 4.2.

[2]Other datasets are small and biased on problem types, which can be seen as subsets of the dataset we used.

[3]Previous works are not comparable: 1) The rules used in Polozov et al. (2015) are not publicly available; 2) Koncel-Kedziorski et al. (2016) have a different input from us, that their system needs a full math word problem and then rewrite it. While our system tries to generate a problem from scratch.

5.3 Implementation Details

The dimension of encoder/decoder hidden state and embedding are set to 512. The hyper-parameter λ in Equation 19 is 0.7. Dropout rate is set to 0.6. All model parameters are initialized using a Gaussian distribution with Xavier scheme (Glorot and Bengio, 2010). We use the Adam (Kingma and Ba, 2015) optimizer with its hyper-parameters set as: learning rate $\alpha = 0.001$, momentum parameters $\beta_1 = 0.9$ and $\beta_1 = 0.999$, and $\epsilon = 10^{-8}$. We also apply gradient clipping (Pascanu et al., 2013) with range $[-5, 5]$. The beam size is set to 3 in the decoding stage. We release the source code at an anonymous URL for blind review.

5.4 Automatic Evaluation

Though the automatic evaluation methods have their limitations in natural language generation evaluation, we use them as important evaluation methods since they are easily reproducible. Furthermore, in the task of math word problem generation, retaining some key information such as the quantities and entities can be well measured by the automatic evaluation methods.

5.4.1 Evaluation Metrics

We evaluate the performance of our model using three evaluation metrics following recent text generation works (Du et al., 2017; Zhou et al., 2017; Fan et al., 2018):

BLEU (Papineni et al., 2002) is a widely used evaluation method in machine translation and text generation.

ROUGE (Lin, 2004) is commonly to evaluate n-gram overlap of summaries with gold-standard sentences.

METEOR (Denkowski and Lavie, 2014) is provided following previous work (Koncel-Kedziorski et al., 2016).

5.4.2 Evaluation Results

Table 2 shows the evaluation results on automatic metrics. From the table, we can see that MAGNET significantly outperforms the baseline models on all metrics. On one hand, our model obtains a large improvement compared to the Equ2Math model (+19.24 ROUGE-1, +17.08 ROUGE-2, +14.02 ROUGE-L). On the other hand, compared to the Topic2Math model, our model has a (+1.15

Models	BLEU	ROUGE-1	ROUGE-2	ROUGE-L	METEOR
KNN	11.97	43.93	23.08	36.14	16.78
Equ2Math	5.00	35.88	12.94	29.56	13.51
Topic2Math	12.26	53.97	28.79	42.19	23.90
MAGNET w/o Entity	12.28	54.28	28.77	42.88	24.73
MAGNET	**12.47**	**55.12**	**30.02**	**43.58**	**24.76**

Table 2: Automatic evaluation results on Dolphin18K. MAGNET w/o Entity indicates the ablation model without entity-enforced loss.

Models	Equation Relevance		Topic Relevance		Solvability		Language Fluency	
	score	Kappa	score	Kappa	score	Kappa	score	Kappa
KNN	1.65	0.75	1.80	0.90	**2.07**	0.90	**2.92**	0.80
Equ2Math	1.97	0.68	1.58	0.85	1.98	0.80	2.62	0.75
Topic2Math	1.63	0.58	2.88	0.85	1.50	0.55	1.98	0.52
MAGNET w/o Entity	1.85	0.40	**2.90**	0.90	1.73	0.65	2.35	0.50
MAGNET	**2.08**	0.43	**2.90**	0.90	**2.07**	0.65	2.57	0.77

Table 3: Human evaluation results: the average scores from the three annotators together (ranged from 1 to 3, higher is better).

ROUGE-1, +1.23 ROUGE-2, +1.39 ROUGE-L) relative gain respectively. The improvement over the baselines demonstrates the usefulness of both equation and topic input. Moreover, without the entity-enforced loss, the performance drops on all metricsthe, which shows its effectiveness.

5.5 Human Evaluation

To better evaluate the performance of our system, we recruit three human annotators to judge the quality of the generated math problems, in addition to the automatic metrics. We randomly select 50 instances in the test set, and show the equation template and topic words with generated math problems from different models. We then ask the annotators to rate the outputs with scores ranged 1 to 3 from the following four aspects (detailed guidelines attached in supplementary):

1. Equation Relevance: the generated problem is relevant to the given equation;
2. Topic Relevance: the generated problem is relevant to the given topic words;
3. Solvability: the generated problem can be solved by an (given) equation;
4. Language Fluency: the generated problem is grammatical and fluent.

Table 3 reports the human evaluation results. As we can see, MAGNET has the highest scores across the three criteria of equation relevance (2.08), topic relevance (2.9), solvability (2.07), outperforming all the baselines and the ablation test. KNN performs the best in terms of language fluency, since as a retrieval-based method its outputs are existing problems in the training data. The Kappa (Randolph, 2005) values on all models range from 0.4 to 0.9, indicating relatively intermediate to excellent agreement among annotators. The human evaluation result is consistent with the automatic evaluation.

6 Discussion

To better understand the model, we show the attention visualization and qualitative analysis with some examples.

6.1 Effect of Model Fusion

To illustrate how MAGNET leverages both inputs, we visualize the output fusion coefficient g (top), and attention of topic words (middle) and equation template (bottom) in Figure 2.

We can see MAGNET generates a reasonable math problem. When generating the words such as “product”, “decreased” in the output, the fusion module is concentrated on the topic words;

Equation Template	$m = [num0]/[num1]/60$
Topic words	sun approximately light kilometer planet travels travel rate take minute
KNN	what is [num0] of [num1] ?
Equ2Math	if i ran [num0] mile in [num1] minute , what is my meter per hour ?
Topic2Math	find the speed of the light [num0] kilometer in [num1] minute [num2] minute .
MAGNET-Entity	a light travels [num0] kilometer at [num1] kilometer per h. how far would it take to travel [num2] kilometer ?
MAGNET	if a man travels [num0] kilometer in [num1] minute , what is the speed in kilometer per h ?
Ground truth	the sun is approximately [num0] kilometer from the planet saturn , and light from the sun travels to saturn at the rate of approximately [num1] kilometer per second. approximately how many minute does it take for light to travel from the sun to saturn?

Table 4: An example of generated math word problems.

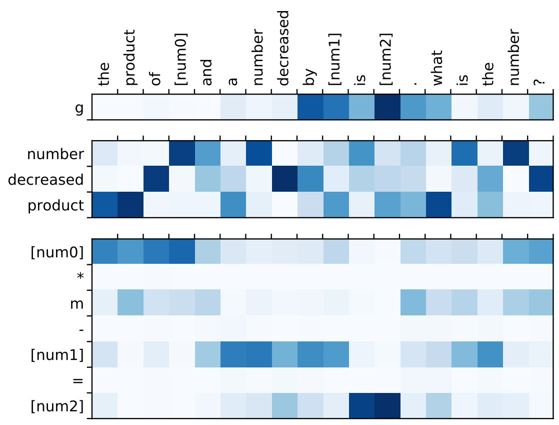

Figure 2: Fusion coefficient g and attention visualization example.

while the attention is focused on the equation template when generating the corresponding numbers ([num0], [num1], ...).

6.2 Qualitative Results

Table 4 shows an example case from the test set. We can see that MAGNET generates a more reasonable math problem, with respect to equation solvability and topic relevance. Please note that the equation template does not exist in the training data. Surprisingly, MAGNET has captured the constant "60" in the equation template and generates "kilometer per h" as unit conversion of minute to hour. The ablation model MAGNET-Entity does not generate reasonable problem as well as the baselines, while MAGNET generates "speed" word problem, perfectly describing the division of two numbers. This further demonstrates the effectiveness of the entity-enforced loss which encourages the relevance between the equation template and the output problem. Due to space limit, we attach more examples in the supplementary.

6.3 Error Analysis

Furthermore, we observe two main types of errors by our model (examples shown in Supplementary): (1) Problem soundness. The generated problem lacks semantic coherence. For example, the model generates "plants $[num0]$ feet of fence to build a fence" that is non-comprehensive; (2) Equation matchness. The input equation template is partially correlated to the output, but not an exact solution of it. This is somewhat expected, since we use the entity-enforced loss only as a soft constraint to ensure the relevance with equation.

7 Conclusion

In this work, we present MAGNET, a novel model for math word problem generation. It considers the input of both equations and topics using a fusion module. Additionally, an entity-enforced loss is introduced to ensure the relevance of equation and the problem during training. Experiments on a large-scale math problem dataset demonstrated our model can produce fluent math word problems that are highly relevant to the given equations and topics.

Future work could incorporate language models to improve the language fluency, and design more fine-grained models to improve the semantic coherence by employing harder constrains of equation template. Furthermore, we would like to extend to more diverse topics with external resources.

8 Acknowledgements

We would like to thank the annotators for their efforts in the evaluation process. Thanks to the anonymous reviewers for their helpful comments and suggestions.

References

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the EMNLP 2014*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Paul Deane and Kathleen Sheehan. 2013. Automatic item generation via frame semantics: Natural language generation of math word problems. In *Proceedings of the Annual Meeting of the National Council on Measurement in Education*, Chicago, America.

Michael Denkowski and Alon Lavie. 2014. Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the EACL 2014 Workshop on Statistical Machine Translation*.

Xinya Du, Junru Shao, and Claire Cardie. 2017. Learning to ask: Neural question generation for reading comprehension. In *Proceedings of the 55th ACL*, pages 1342–1352, Vancouver, Canada. Association for Computational Linguistics.

Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical neural story generation. In *Proceedings of the 56th ACL*, pages 889–898. Association for Computational Linguistics.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Aistats*, volume 9, pages 249–256.

Ian J Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron C Courville, and Yoshua Bengio. 2013. Maxout networks. *ICML (3)*, 28:1319–1327.

Michael Heilman. 2011. *Automatic factual question generation from text*. Ph.D. thesis, Carnegie Mellon University.

Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. 2014. Learning to solve arithmetic word problems with verb categorization. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*.

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 52nd Annual Meeting of the Association for Computational Linguistics*.

Singley Mark K. and Bennett Randy Elliot. 2002. Item generation and beyond: Applications of schema theory to mathematics assessment. In *Irvine, Sidney H.; Kyllonen, Patrick C. (eds.) Item Generation for Test Development. Mahwah, NJ: Lawrence Erlbaum Associates*.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of 3rd International Conference for Learning Representations*, San Diego.

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

Nate Kushman, Yoav Artzi, Luke Zettlemoyer, and Regina Barzilay. 2014. Learning to automatically solve algebra word problems. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out: Proceedings of the ACL-04 workshop*, volume 8. Barcelona, Spain.

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

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. *ICML (3)*, 28:1310–1318.

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

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Justus J Randolph. 2005. Free-marginal multirater kappa (multirater k [free]): An alternative to fleiss' fixed-marginal multirater kappa. *Online submission*.

Shuming Shi, Wang Yuehui, Chin-Yew Lin, Xiaojiang Liu, and Yong Rui. 2015. Automatically solving number word problems by semantic parsing and reasoning. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*.

Qingyu Zhou, Nan Yang, Furu Wei, Chuanqi Tan, Hangbo Bao, and Ming Zhou. 2017. Neural question generation from text: A preliminary study. *arXiv preprint arXiv:1704.01792*.