Investigating Math Word Problems using Pretrained Multilingual Language Models

Anonymous ACL submission

Abstract

In this paper, we revisit math word problems (MWPs) from the *cross-lingual* and *multilingual* perspective. We construct our MWP solvers over pretrained multilingual language models using the sequence-to-sequence model with copy mechanism. We compare how the MWP solvers perform in cross-lingual and multilingual scenarios. To facilitate the comparison of cross-lingual performance, we first adapt the large-scale English dataset MathQA as a counterpart of the Chinese dataset Math23K. Then we extend several English datasets to bilingual datasets through machine translation plus human annotation. Our experiments show that the MWP solvers may not be transferred to a different language even if the target expressions share the same numerical constants and operator set. However, it can be better generalized if problem types exist on both source language and target language.

1 Introduction

How to use machine learning and NLP techniques to solve Math Word Problems (MWPs) has attracted much attention in recent years (Hosseini et al., 2014; Kushman et al., 2014; Roy et al., 2015; Ling et al., 2017; Wang et al., 2017, 2018; Amini et al., 2019). Given a math problem expressed in human language, a MWP solver typically first converts the input sequence of words to an *expression tree* consisting of math operators and numerical values, and then invokes an executor (such as the *eval* function in Python) to execute the expression tree to obtain the final numerical answer. Figure 1 shows an example math word problem, the correct expression tree, and the final answer.

Despite the relatively simple syntax of these expression trees, building MWP solvers is not a trivial task, and researchers have proposed various methods to tackle the different challenges of this problem such as statistical methods (Kushman et al., 2014; Roy et al., 2015), parsing-based methods (Shi et al., 2015) and generation-based methods (Wang et al., 2018; Xie and Sun, 2019). However, an aspect that has been largely overlooked is cross-lingual and multilingual MWP solving, i.e., whether a MWP solver trained on one human language can still work on another human language, or whether a MWP solver trained on multiple human languages together is more effective than a solver trained on only one language. We believe this is an interesting aspect to study for the following reasons. First, in cognitive science, people have long studied the relationship between humans' numerical processing abilities and language abilities, and found that on the one hand, the two are largely independent (Xu and Spelke, 2000), but on the other hand, "acquiring and mastering symbolic representations of exact quantities critically depends on language and instruction" (Van Rinsveld et al., 2015). It is therefore also intriguing to study whether machines separately acquire arithmetic and language abilities. Second, with pre-trained large-scale multilingual language models such as mBERT (Devlin et al., 2019) and XLM-R (Conneau et al., 2020), which presumably project different human languages into a common embedding space, we have seen some success in cross-lingual NLP tasks such as XNLI (Conneau et al., 2018) and

Problem: A chef needs to cook 9 potatoes. He has already cooked 7. If each potato takes 3 minutes to cook, how long will it take him to cook the rest?

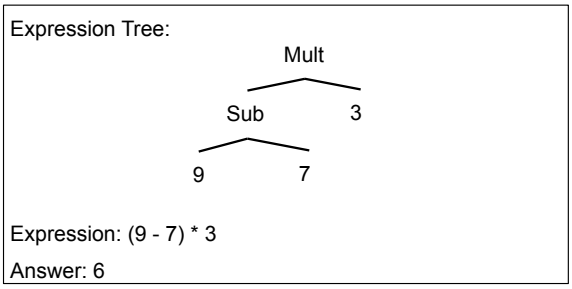

Figure 1: Example of an MWP and its expression tree.

MLQA (Lewis et al., 2020) in both zero-shot and few-shot settings (Wu and Dredze, 2019; Conneau et al., 2020). It is therefore reasonable to expect that for MWP solving, there is the possibility of transferring machine's capability of MWP solving from one language to another by leveraging these pre-trained multilingual language models.

In this paper, we conduct an empirical study to understand to what extent MWP solvers can work in cross-lingual and multilingual settings. Specifically, we ask the following questions: (1) Cross-lingual setting: Given a model trained with monolingual dataset, can the model solve MWPs over another language? (2) Multilingual setting: Can combining datasets of different languages further boost the performance for each language? (3) Can we identify some critical factors that may affect the results in (1) and (2)?

In order to empirically answer the questions above, we need multilingual MWP datasets, which are limited currently. We first use large scale datasets like Math23K (Wang et al., 2017) and MathQA (Amini et al., 2019) as monolingual MWPs resource and further adapt MathQA to have the same operator set and expression style with Math23K. To better evaluate the models with parallel corpus, we extend some existing MWP datasets by translating them from English into Chinese. We then conduct three sets of experiments on the constructed datasets. We find that: (1) a cross-lingual MWP solver finetuned on one language cannot work on a second language, even if they are sharing the same decoding vocabulary, (2) a multilingual MWP solver may not boost performance for all the training languages but can improve those problems of similar types if one training language is close to the evaluation language, (3) combining (1) (2), we think for multilingual MWP solvers, despite language similarity, the performance relies heavily on domain similarity (problem types).

Our work makes the following contributions: (1) To the best of our knowledge, we are the first to study cross-lingual and multilingual MWP solving, and we empirically demonstrate that cross-lingual MWP solving is still difficult, but multilingual MWP solving is to some extent effective. (2) We discover that multilingual MWP solving is mostly effective for questions with similar problem types. (3) Our constructed datasets can help other researchers to further study cross-lingual and multilingual MWP solving.

2 Cross-lingual and Multilingual MWP Solvers

In this work, as we are focusing on the cross-lingual and multilingual properties of MWPs, we need to train separate MWP solvers using different datasets. Our cross-lingual MWP solver will be trained using one language but evaluated using another. Our multilingual MWP solver can be trained on all languages available and evaluated separately. To suffice these goals, it would be better if the problems in different languages have comparable properties. Since we are using pretrained multilingual language models as the sequence embedder of the encoder, all the languages can be projected into a shared representation. However, the candidate datasets also need to share a common operator set and numerical constants to make the decoding process consistent. But some of the categories from MathQA do not exist on Math23K or one of the operators is not in our permitted set. Therefore, we need to adapt MathQA as a counterpart of Math23K sharing the same decoding vocabulary, including operators and constants.

2.1 Adaptation of MathQA

We adapt MathQA by doing the following:

1) We notice that the annotated formulas in MathQA are function calls of predefined functions which can be converted into a tree using an abstract syntax tree (AST) parser.
2) To be consistent with Math23K, which covers only basic arithmetic operators like addition (Add), subtraction (Sub), division (Div), multiplication (Mult) and exponentiation (Pow), we keep only functions in MathQA that can be expressed in such operators. For example, $volume_sphere(r)$ from MathQA equalizes to $\frac{4}{3}\pi r^3$ and is adapted using the method shown in Figure 2. Formulas containing operators not used in Math23K, like *sine* and *permutation*, are not considered in this work. A full list of adapted operators can be found in Table 5 of Section B.
3) Upon constructing the trees using permitted operators, we evaluate each sample to verify its correctness against its ground-truth answer. Those cases that fail to get the correct answer are not considered in this work.

After the adaptation, we get the adapted MathQA dataset of solvable problems with comparable sizes and question types to Math23K. For Math23K, we further sample a development set of size 1000 from

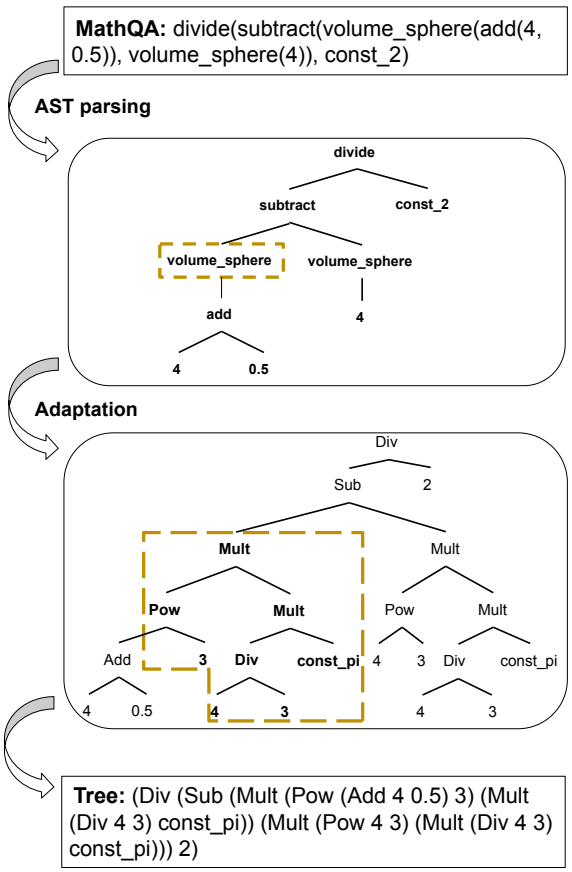

Figure 2: Adaptation of MathQA to Math23K. The part highlighted with dashed lines shows the adaptation of the function *volume_sphere*.

	Math23K		MathQA	
	w/o Pow	w/ Pow	w/o Pow	w/ Pow
Train	21,107	21,161	15,302	16,645
Dev	995	1,000	2,263	2,479
Test	999	1,000	1,532	1,653

Table 1: Statistics of different splits for Math23K and the adapted MathQA.

its training set. Considering the operator Pow has only several training and evaluating instances on Math23K, we separate them with others to make a fairer adaptation of MathQA to Math23K. We show the statistics of both Math23K and the adapted MathQA in Table 1. In this work, all the experiments will be conducted on the dataset marked with w/o Pow.

2.2 Zero-shot Cross-lingual Evaluation Datasets

To test cross-lingual transferability of MWP solvers, we use the problem-type-specific datasets AddSub (Hosseini et al., 2014), SingleOp (Roy et al., 2015) and MultiArith (Roy and Roth, 2015) discussed in Section A.3 as evaluation datasets. To extend these datasets for cross-lingual evaluation, we make use of online machine translation APIs to translate them into Chinese and further manually refine the translations to be more native. For each dataset, we list an example in Table 2, in both English (En) and Chinese (Zh).

3 Experiments

3.1 Experiment Setup

Evaluation metrics: The model is expected to be a math problem solver, so the generated expressions should be executable by a specific compiler and executor. During evaluation, each problem is counted as solved if the absolute error rate for the executed value and the target value is lower than a predefined threshold. In our experiments, we choose $1e^{-4}$ as the threshold. The final evaluation metric is the accuracy of solved problem against all the problems.

Methods to be compared: We empirically compare the following cross-lingual: (1) **mBERT-zh** is using original multilingual BERT (Devlin et al., 2019) but trained over Math23K only; (2) **mBERT-en** is using original multilingual BERT (Devlin et al., 2019) but trained over the adapted MathQA only, and multilingual methods: (1) **mBERT-xl** is using original multilingual BERT (Devlin et al., 2019) but trained by mixing Math23K and the adapted MathQA; (2) **XLM-R-xl** is using XLM-R (Conneau et al., 2020) but trained by mixing Math23K and the adapted MathQA.

Other experiment settings: We choose to use multi-lingual BERT (mBERT) (Devlin et al., 2019) for cross-lingual training. We train our models using one Nvidia 2080ti and a batch size of 160. The learning rate is set to $3e^{-5}$ with a scheduler supporting polynomial decay. The training lasts for at most 150 epochs and will stop after 30 epochs if no improvement is observed.[1]

3.2 Results

We list experiment results of all the methods in Table 3.

Cross-lingual MWP Solver The first research question we want to answer is to what extent

[1] We will release our configuration files for all the experiments in a public code repository.

Dataset	AddSub	SingleOp	MultiArith
Problem Types	addition, subtraction	single operation	multi-step
En	Keith has 20 books . Jason has 21 books . How many books do they have together ?	Lisa flew 256 miles at 32 miles per hour. How long did Lisa fly?	A chef needs to cook 9 potatoes. He has already cooked 7. If each potato takes 3 minutes to cook, how long will it take him to cook the rest?
Zh	基思有20本书。杰森有21本书。他们总共有多少本书?	丽莎以每小时32英里的速度飞行了256英里。丽莎飞了多长时间?	厨师需要煮9个土豆。他已经煮了7个了。如果每个土豆煮3分钟，剩下的他要煮多久?
Size	395	562	600

Table 2: Examples from each dataset used for zero-shot cross-lingual evaluation.

Model	Test		Zero-shot					
	Math23K	MathQA	AddSub		SingleOp		MultiArith	
	zh	en	zh	en	zh	en	zh	en
mBERT-zh	76.5	3.3	30.9	10.4	66.0	32.7	51.2	15.7
mBERT-en	0.5	77.9	2.8	6.1	5.0	10.5	5.0	3.2
mBERT-xl	76.3	79.0	35.2	24.1	69.8	41.6	45.0	16.0
XLM-R-xl	75.5	79.0	39.0	21.3	67.4	40.4	44.7	18.3

Table 3: Comparisons of different cross-lingual models over Test set and zero-shot datasets.

a MWP solver trained on one language can work on another language, with the help of pre-trained multilingual language models. Table 3 shows that the MWP solvers trained using either Math23K (**mBERT-zh**) or MathQA (**mBERT-en**) have achieved impressive performance when tested in the same language. However, the performance over a different language drops drastically and is almost negligible. In a word, the MWP solver is almost non-transferable when it is trained on one language but evaluated over a second with the same operator set.

Multilingual MWP Solver The second research question we want to answer is whether training a MWP solver on multiple languages helps improve its effectiveness compared with training on a single language. We can see that mixing two languages to train can give us a more language-agnostic model as the performance on Test split of both languages are competitive with monolingual cases. What's more, on the newly extended bilingual datasets, there are consistent improvements for most of the datasets, especially for the English language.

Considering the difficulty of problems, these bilingual evaluation datasets are closer to Math23K (primary school) than to MathQA (GRE or GMAT). Adding that mBERT-zh is also doing better than mBERT-en on English language, we suspect domain similarity is more important than language for MWP solvers.

4 Conclusion

In this paper, we revisit the math word problems using a generation-based method constructed over pretrained multilingual models. To assist analysis of cross-lingual properties of math solvers, we adopt two large-scale monolingual datasets and further adapts MathQA into the same annotation framework with Math23K. We also reuse earlier smaller datasets and upgrade them into bilingual datasets by machine translation and manual checking. Our experiments show that the MWP solvers may not be transferred to a different language even if the target expressions have the same operator set and constants. But for both cross-lingual and multilingual cases, it can be better generalized if problem types exist on both source language and target language. Problems considered to be easy by humans may still be hard for a math solver trained over the same language but from a different domain. This tells us that for math word problem solvers, it might be beneficial to consider balancing different question types and permitted operators during training.

References

Aida Amini, Saadia Gabriel, Shanchuan Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. 2019. MathQA: Towards interpretable math word problem solving with operation-based formalisms. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2357–2367, Minneapolis, Minnesota. Association for Computational Linguistics.

Daniel G. Bobrow. 1964. Natural language input for a computer problem solving system. Technical report, USA.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.

Alexis Conneau, Ruty Rinott, Guillaume Lample, Adina Williams, Samuel Bowman, Holger Schwenk, and Veselin Stoyanov. 2018. XNLI: Evaluating cross-lingual sentence representations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2475–2485, Brussels, Belgium. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O.K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1631–1640, Berlin, Germany. Association for Computational Linguistics.

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 (EMNLP)*, pages 523–533, Doha, Qatar. Association for Computational Linguistics.

Danqing Huang, Shuming Shi, Chin-Yew Lin, and Jian Yin. 2017. Learning fine-grained expressions to solve math word problems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 805–814, Copenhagen, Denmark. Association for Computational Linguistics.

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, Berlin, Germany. Association for Computational Linguistics.

Bugeun Kim, Kyung Seo Ki, Donggeon Lee, and Gahgene Gweon. 2020. Point to the Expression: Solving Algebraic Word Problems using the Expression-Pointer Transformer Model. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3768–3779, Online. Association for Computational Linguistics.

Rik Koncel-Kedziorski, Subhro Roy, Aida Amini, Nate Kushman, and Hannaneh Hajishirzi. 2016. MAWPS: A math word problem repository. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1152–1157, San Diego, California. 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 (Volume 1: Long Papers)*, pages 271–281, Baltimore, Maryland. Association for Computational Linguistics.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. ALBERT: A lite BERT for self-supervised learning of language representations. In *International Conference on Learning Representations*.

Patrick Lewis, Barlas Oguz, Ruty Rinott, Sebastian Riedel, and Holger Schwenk. 2020. MLQA: Evaluating cross-lingual extractive question answering. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7315–7330, Online. Association for Computational Linguistics.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 158–167, Vancouver, Canada. Association for Computational Linguistics.

Jinghui Qin, Lihui Lin, Xiaodan Liang, Rumin Zhang, and Liang Lin. 2020. Semantically-aligned universal tree-structured solver for math word problems. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*,

pages 3780–3789, Online. Association for Computational Linguistics.

Subhro Roy and Dan Roth. 2015. Solving general arithmetic word problems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1743–1752, Lisbon, Portugal. Association for Computational Linguistics.

Subhro Roy, Tim Vieira, and Dan Roth. 2015. Reasoning about quantities in natural language. *Transactions of the Association for Computational Linguistics*, 3:1–13.

Shuming Shi, Yuehui Wang, 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*, pages 1132–1142, Lisbon, Portugal. Association for Computational Linguistics.

Shyam Upadhyay, Ming-Wei Chang, Kai-Wei Chang, and Wen-tau Yih. 2016. Learning from explicit and implicit supervision jointly for algebra word problems. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 297–306, Austin, Texas. Association for Computational Linguistics.

Amandine Van Rinsveld, Martin Brunner, Karin Landerl, Christine Schiltz, and Sonja Ugen. 2015. The relation between language and arithmetic in bilinguals: insights from different stages of language acquisition. *Frontiers in psychology*, 6:265.

Lei Wang, Yan Wang, Deng Cai, Dongxiang Zhang, and Xiaojiang Liu. 2018. Translating a math word problem to a expression tree. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1064–1069, Brussels, Belgium. Association for Computational Linguistics.

Mingxuan Wang, Zhengdong Lu, Jie Zhou, and Qun Liu. 2017. Deep neural machine translation with linear associative unit. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 136–145, Vancouver, Canada. Association for Computational Linguistics.

Shijie Wu and Mark Dredze. 2019. Beto, bentz, becas: The surprising cross-lingual effectiveness of BERT. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 833–844, Hong Kong, China. Association for Computational Linguistics.

Zhipeng Xie and Shichao Sun. 2019. A goal-driven tree-structured neural model for math word problems. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 5299–5305. International Joint Conferences on Artificial Intelligence Organization.

Fei Xu and Elizabeth S Spelke. 2000. Large number discrimination in 6-month-old infants. *Cognition*, 74(1):B1–B11.

Lipu Zhou, Shuaixiang Dai, and Liwei Chen. 2015. Learn to solve algebra word problems using quadratic programming. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 817–822, Lisbon, Portugal. Association for Computational Linguistics.

A Preliminaries

A.1 The MWP Solver Task

We first formally define the task of building MWP solvers. Given a math word problem with n words $W = (w_1, w_2, \ldots, w_n)$, and k numerical values $N = (n_0, n_1, \ldots, n_k)$, the model needs to generate a flattened tree representation using operators from permitted operator set $\mathcal{O}$ and numerical values from constants $\mathcal{C}$ and N. The generated tree should be able to be evaluated via some compiler and executor to return a numerical value.

A.2 Solution Framework

A MWP solver needs to generate executable code for a target programming language to be evaluated by an executor compiled for the programming language.

Our MWP solver is built upon a sequence-to-sequence model with copy mechanism (Gu et al., 2016). Specifically, we use a pretrained multilingual model as the encoder to get contextualized representations of math word problems. Due to the word piece tokenizer, the encoded context is not well-aligned to original input words. We choose to map these word pieces back to input words through mean pooling. Then we pass the mean pooled word representations to a bidirectional LSTM. Finally, we use a LSTM decoder with copy mechanism, which takes in the last decoded word vector and intermediate reading states, to predict the next token one by one. When the decoding finishes, we are expecting to get a linear tree representation. We attach the full model details in Section B.

Given the decoded tree representation, we first convert the generated linear tree representation into a piece of python expression with basic operators (+,-,*,/,**), then use the built-in function *eval* in Python to execute the generated code.

A.3 Existing Datasets

We use two large-scale datasets for this cross-lingual research. One is Math23K (Wang et al.,

Dataset	Problem Types	Size
AddSub (Hosseini et al., 2014)	Add Sub	395
SingleOp (Roy et al., 2015)	Add Sub Mult Div	562
MultiArith (Roy and Roth, 2015)	(Add, Sub) (Sub, Add) (Add, Mult) (Add, Div) (Sub, Mult) (Sub, Div)	600

Table 4: Datasets which are focusing on specific problem types.

2017) in Chinese and the other is MathQA (Amini et al., 2019) in English. Although the two datasets are similar in size and question types, there are still differences in terms with permitted operators and annotations.

Math23K The dataset Math23K (Wang et al., 2017) contains math word problems for elementary school students in *Chinese* (zh) and is crawled from multiple online education websites. The dataset focuses on arithmetic problems with a single-variable and contains 23,161 problems labeled with structured equations and answers.

MathQA The dataset is a new large-scale, diverse dataset of 37k multiple-choice math word problems in *English* (en). Each question is annotated with an executable formula using a new operation-based representation language (Amini et al., 2019). It covers multiple math domain categories.

Other datasets focusing on specific problem types These datasets are smaller in size but more focused on specific problem types. We follow the dataset naming conventions from MAWPS (Koncel-Kedziorski et al., 2016).

Specifically, AddSub (Hosseini et al., 2014) covers arithmetic word problems on addition and subtraction for third, fourth, and fifth graders. Its problem types include combinations of additions, subtractions, one unknown equation, and U.S. money word problems. SingleOp (Roy et al., 2015) is a dataset with elementary math word problems of single operation. MultiArith (Roy and Roth, 2015) includes problems with multiple steps which we listed all the seven types in Table 4. These datasets are all in *English* (en). We will illustrate how we extend them into bilingual datasets in Section 2.2.

B Method

In this section, we construct a generation-based MWP solver using a sequence-to-sequence model with copy mechanism.

Our whole model can be visualized in modules through Figure 3. The detailed illustration for each module is given as following:

Encoder Our encoder is built upon a pretrained multilingual transformer, either BERT or XLM-R. Suppose our input word w_i is tokenized into word pieces $(x_{i1}, x_{i2}, \ldots)$ and let $\mathbf{h}_{ij} \in \mathbb{R}^{d_h}$ denotes the hidden vector produced by the pretrained model representing x_{ij}. We use average pooling to get the representation for the word w_i, denoted as $\mathbf{h}_i$. Then we feed this contextualized representation of the math word problem into a two-layer bidirectional LSTM. The output of this biLSTM is the encoder hidden states for decoding, denoted as $\mathbf{M} = (\mathbf{m}_0, \mathbf{m}_1, \ldots, \mathbf{m}_n)$.

Decoder We use a LSTM cell as the decoding cell to predict the next token. For each decoding step t, the cell will accept the embedding for previous word as input and output a decoder state $\mathbf{s}^t \in \mathbb{R}^{d_s}$. Most of the numerical values in MWPs do not exist in the target vocabulary. Therefore, we need copy mechanism (Gu et al., 2016) to facilitate generation of numerical values during decoding. The copy scores are calculated as follows,

$$u_i^t = \sigma(\mathbf{m}_i^\intercal \mathbf{W}_c)\mathbf{s}^t \quad (1)$$

where $\mathbf{W}_c \in \mathbb{R}^{d_h \times d_s}$. However, the embedding of a copied token will be identical to an out-of-vocabulary token. To better capture the information from last decoding step, we use the copy score to further derive a state of selecting from source tokens, which is called *Selective Read*.

$$\mathbf{q}^t = \mathrm{softmax}(\mathbf{u}^t) \quad (2)$$

$$\mathbf{b}^t = \sum_i q_i^t \mathbf{m}_i \quad (3)$$

We use a bilinear attention to attentively read information from $\mathbf{M}$, getting the context vector $\mathbf{c}^t$, which is called *Attentive Read*.

$$v_i^t = \sigma(\mathbf{m}_i^\intercal \mathbf{W}_a \mathbf{s}^t + b) \quad (4)$$

$$\mathbf{d}^t = \mathrm{softmax}(\mathbf{v}^t) \quad (5)$$

$$\mathbf{c}^t = \sum_i d_i^t \mathbf{m}_i \quad (6)$$

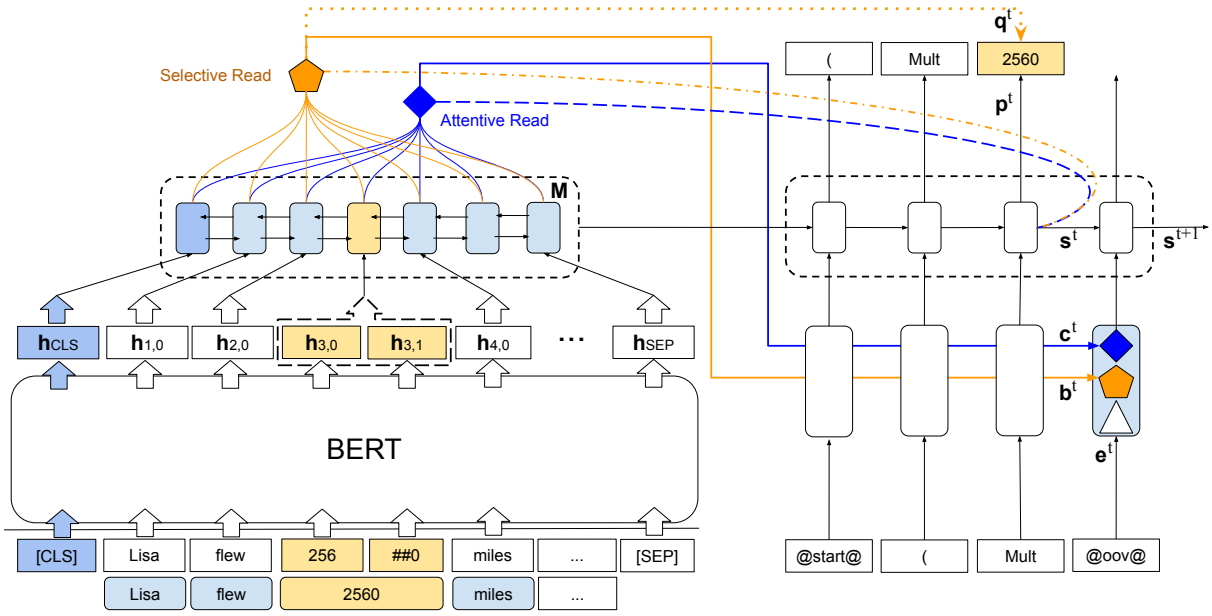

Figure 3: Sequence-to-sequence model with copy mechanism.

where $\mathbf{W}_a \in \mathbb{R}^{d_h \times d_s}$.

From the problem definition, the target vocabulary is $\mathcal{V} = \mathcal{O} \cup \mathcal{C}$. The generation score for the next token is given by:

$$\mathbf{p}^t = \mathbf{W}_d^\intercal \mathbf{s}^t + b \tag{7}$$

where $\mathbf{W}_d \in \mathbb{R}^{d_s \times |\mathcal{V}|}$.

The state updating process for the decoding cell takes in a fused information of last word embedding $\mathbf{e}^t \in \mathbb{R}^{d_e}$, selective read state $\mathbf{b}^t$ and attentive read state $\mathbf{c}^t$.

$$\mathbf{s}^{t+1} = f(\mathbf{W}_s[\mathbf{e}^t, \mathbf{b}^t, \mathbf{c}^t], \mathbf{s}^t) \tag{8}$$

where $\mathbf{W}_s \in \mathbb{R}^{d_s \times (d_e + d_h + d_h)}$.

C Related work

Solving Math Word Problems (MWPs) has been attracting researchers since the emergence of artificial intelligence. STUDENT(Bobrow, 1964) is a rule-based math word problem solver which contains a pipeline that consists of heuristics for pattern transformation. Many researchers start with the fundamental problem types like addition and subtraction (Hosseini et al., 2014) or those that have only one single operator (Roy et al., 2015). Roy and Roth (2015) look at problems that require multi-steps using two or more operators. The question types of MWPs are also expanding. Rather than focusing on problems that need only one variable, Kushman et al. (2014) propose a dataset ALG514 which includes problems with a system of equations. With the development of deep learning, there has been a demand for large-scale datasets with more variations. Dolphin18K (Huang et al., 2016) is a large-scale dataset that is more than 9 times of the size of previous ones, and contains many more problem types. Math23K (Wang et al., 2017) contains math word problems for elementary school students in Chinese language and is crawled from multiple online education websites. MathQA (Amini et al., 2019) is a new large-scale, diverse dataset of 37k multiple-choice math word problems in English and each problem is annotated with an executable formula using a new operation-based representation language. HMWP (Qin et al., 2020) contains three types of MWPs: arithmetic word problems, equations set problems, and non-linear equation problems.

Various approaches have been proposed to solve MWPs. Template-based approaches (Kushman et al., 2014; Zhou et al., 2015; Upadhyay et al., 2016; Huang et al., 2017) are widely adopted as numbers appeared in the expressions are usually sparse in the representation space and the expressions may fall into the same category. More recently, the community is also paying more attention to train a math solver by fine-tuning pretrained language models. For example, EPT (Kim et al., 2020) adopts ALBERT (Lan et al., 2020) as the encoder for its sequence-to-sequence module.

The monolingual performance gains achieved re-

Adapted Operators	Filtered Operators
add,subtract,multiply, rectangle_area,divide, speed,power,negate,inverse, square_area,sqrt, square_edge_by_area, cube_edge_by_volume, volume_cube,surface_cube, square_perimeter, rectangle_perimeter, stream_speed,triangle_area, triangle_perimeter,surface_sphere, volume_sphere,rhombus_area, quadrilateral_area,volume_cylinder, circle_area,volume_cone,circumface, diagonal,volume_rectangular_prism, original_price_before_loss, original_price_before_gain, p_after_gain, square_edge_by_perimeter,negate_prob	floor,choose,min,tangent,sine, reminder,lcm,factorial,gcd,max, permutation, triangle_area_three_edges, surface_cylinder,rhombus_perimeter, surface_rectangular_prism, speed_in_still_water,log

Table 5: Operators that are adapted in MathQA.

cently have not been evaluated from cross-lingual and cross-domain perspectives. Therefore, we decide to revisit MWPs using current SOTA pre-trained multilingual language models to construct a competitive math solver and conduct experiments over various bilingual evaluations.