Midway Report: Multilingual Deep Neural Math Word Problem Solver

Ramya Sree Boppana `rboppana6`

Shrija Mishra `smishra309`

Sheryl Ratnam `sratnam6`

1 Abstract

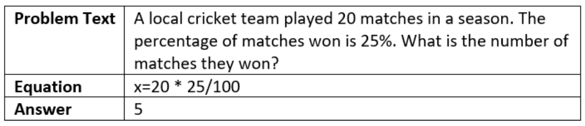
In this project, we set out to develop a language agnostic math word problem solver through the implementation of deep learning-based methods. We utilized subsets of two multilingual large-scale datasets, Math23K (Chinese language) and Dolphin-S (English language), to train and test our language-agnostic models. In addition to obtaining the data and performing data pre-processing procedures, we implemented the second component of our two-stage T-RNN model, which involves answer generation. We built and ran our model and compared the results to the baseline model. In the remainder of the semester, we intend to complete the implementation and evaluation of our model by building out the first component, which relies on equation template prediction, and incorporating a language agnostic embedding module.

2 Goal

Our goal remains the same as detailed in our initial project proposal. While various symbolic or statistical learning methods have been proposed to address the challenge of automatic math word problem solving, our project builds upon a different family of deep learning (DL) based methods, first introduced by (Wang et al., 2017) with Deep Neural Solver (DNS). We extrapolate the template-based solution architecture proposed by (Wang et al., 2019) to design a language-agnostic automatic math word problem solver. Our multifaceted approach is modeled after the development of T-RNN and is trained and tested against subsets of two large-scale datasets, Math23K and Dolphin-S.

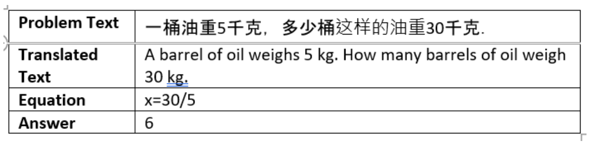
Dataset Name	#prob	#temp	#words
Dolphin-S	140	80	19K
Math23K	23161	2187	822K

Table 1: Data Summary Statistics



Problem Text	A local cricket team played 20 matches in a season. The percentage of matches won is 25%. What is the number of matches they won?
Equation	x=20 * 25/100
Answer	5

Figure 1: Dolphin-S Sample Problem



Problem Text	一桶油重5千克，多少桶这样的油重30千克.
Translated Text	A barrel of oil weighs 5 kg. How many barrels of oil weigh 30 kg.
Equation	x=30/5
Answer	6

Figure 2: Math23K Sample Problem

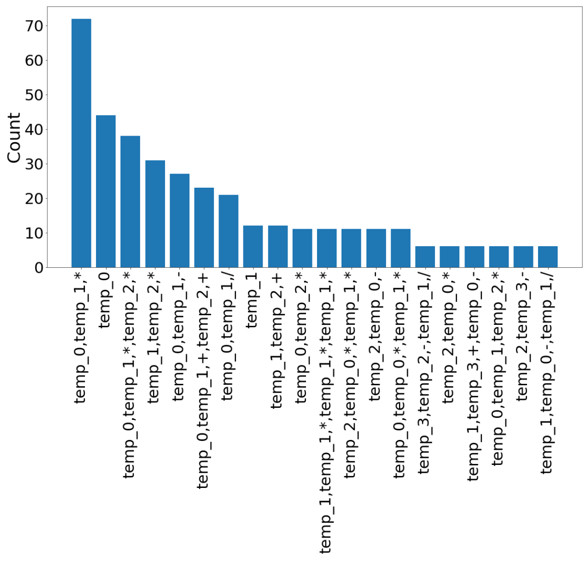


Figure 3: Dolphin-S

3 Progress

3.1 Data

As mentioned in the proposal, we collected Dolphin18K (Huang et al., 2016) as the English dataset and Math23K (Wang et al., 2017) for the Chinese dataset. Dolphin 18K has diverse English math word problems ranging from one variable to multiple variable equations. We concentrated on problems involving a single variable equation. Af-

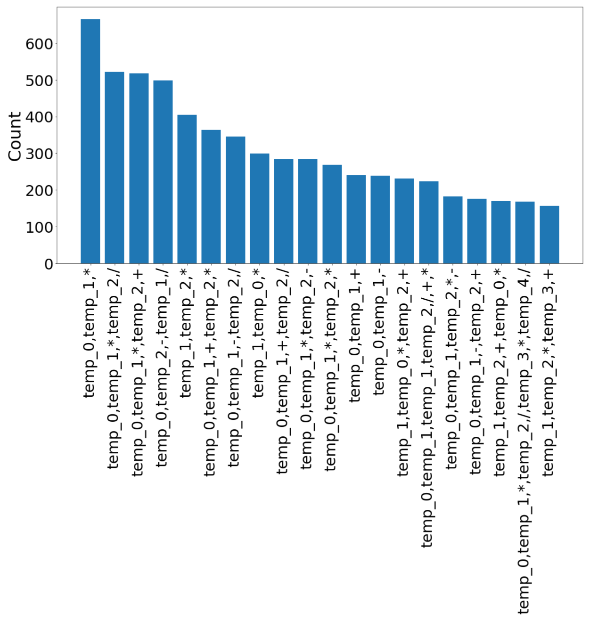


Figure 4: Math23K

ter filtering out the multi-variable equations from Dolphin-S (a subset of Dolphin 18K) we obtained a dataset with 1571 rows, which we used for model development. On the other hand, Math23K is a rich dataset with 23162 single variable Chinese math word problems. We used Google translate to debug and understand this dataset. Table 1 briefly describes the data summary statistics. Figures 1 and 2 show some sample data from our datasets. The translated text in Figure 2 is just for demonstration purposes. We are not using it for our model. Figures 3 and 4 show the distribution of top 10 templatized postfix equations for both the datasets. We elaborate further on equation templates in subsection 3.2.

After filtering single variable equations, we performed some pre-processing steps on problem text like removal of unnecessary text (urls, tags, etc.), stop words, whitespace and punctuation. The cleaned text now contains useful words and numbers for word problems. Next, we pre-process equations to follow the format:

$$x = val1 < op > val2 < op > val3$$

For example, an equation like $1 + 3 = n$ is converted to $n = 1 + 3$. Then we perform one pass of stemming on problem text. For example, $halved$ will be converted to $half$. Later, we convert all numbers contained in the problem text from word form to their mathematical form. We also do a sanity check for equations being complied to the problem text. We then filter out complex problems where the number of literals in problem text doesn't comply to the number of literals in the equation. This will eliminate problems like ”What is the sum of the distinct prime factors of 2007 ” with equation ”x = 3 + 223”. These sorts of problems require an external knowledge base and is not within the scope of our project.

3.2 Method

Our main goal is to develop a robust language agnostic model that can handle large scale, multilingual datasets and afford developers a high level of interpretability during equation template prediction without compromising accuracy. Within the sub-domain of DL-based techniques for math word problem solving, we focus on implementing a template-based solution using Recursive Neural Networks (RNN). Literature suggests that RNN-based solutions are better equipped to handle larger, more diverse datasets, such as Math23K and Dolphin18K (Wang et al., 2017). Previous statistical models don't fare well on datasets of this magnitude, and alternative DL-based approaches i.e. the stand-alone seq2seq model lack interpretability. Our model, as derived from (Wang et al., 2019) leverages RNN in conjunction with the seq2seq model and expression trees to facilitate answer generation.

The T-RNN model is implemented in two stages. The first stage involves equation template prediction. This method of templatization is not new to the field and has been introduced in prior papers, where models learn the mapping between an input problem's equation structure and candidate template. The uniqueness of our implemented approach stems from quantity representation and template space reduction. In our structural template, we represent each i^{th} detected quantity from the question text with common variables like $temp0$ and $temp1$, and convert the equation template to a postfix expression. In order to provide a contextual overview of the different templates within the datasets we've chosen, we plotted Figures 3 and 4 that show the distribution of different templates in their respective dataset. Additionally, we seek to encapsulate operators as abstract representations to reduce the number of equation templates in the template space and allow for efficient template prediction. This is illustrated in subsection 3.1. We then apply the first stage seq2seq model as referenced by (Wang et al., 2019) to predict the equation template represented by the expression tree.

The expression tree contains leaf nodes, which represent quantities whose values can be inferred, and inner nodes, which represent unknown operators. Consequently, this leads to the second stage of the model, which involves answer generation, and, more specifically, utilizing RNN to infer the unknown operator nodes. This bottom-up approach readily leverages the suffix expression of the tree structure. This stage includes two layers. In the first layer, we utilized Bidirectional LSTM with a self attention mechanism to design a quantity embedding network. In the second layer, we implemented RNN, as contributed by (Wang et al., 2019), to predict unknown operators using the derived quantity features and structural templates. The RNN recursively deploys the softmax function to estimate the probability of an operator until all operators of inner nodes have been successfully predicted.

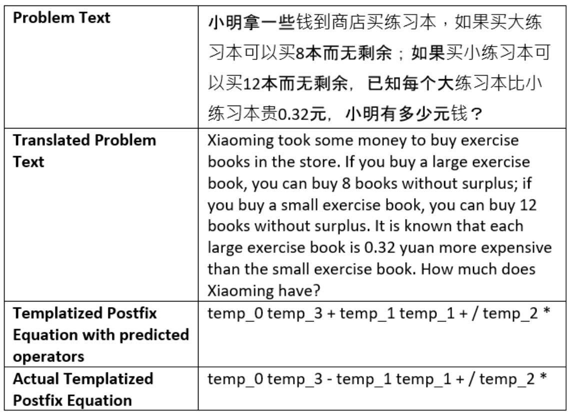
The final component of our project involves integrating a language agnostic embedding module into our T-RNN model. Our model for the language-agnostic math problem solver is motivated by the works of (Artetxe and Schwenk, 2019) and (Yang et al., 2019). One such architecture, LASER, uses a language agnostic BiLSTM encoder and an auxiliary decoder to learn a classifier on top of resulting embeddings (Artetxe and Schwenk, 2019). This is a singular encoder that can be applied to downstream tasks without finetuning and retains the capacity to handle multiple languages, such that semantically similar sentences across languages are close in the embedding space.

To the best of our knowledge, this is the first exploration of universal language agnostic sentence embeddings in the space of RNN-enabled, template-based automatic math word problem solvers. However, we will develop a novel baseline model to compare our model with.

3.3 Preliminary Results

At this point in our project work, we have implemented the second stage involving the answer generation module. Figure 5 shows the output of our model for an example problem from Math23k. We have also developed a baseline model which predicts operators randomly given a template equation structure. We use this baseline model to evaluate the success of our model.

In order to evaluate our model, we use two performance metrics - Integrated accuracy and Element wise Accuracy. In integrated accuracy, an output is considered a positive hit if all the predicted operators in the equation match the annotated equation entirely whereas the element wise accuracy metric is calculated operator wise. Equations 1 and 2 show the formulae to calculate accuracies.



Problem Text	小明拿一些钱到商店买练习本，如果买大练习本可以买8本而无剩余；如果买小练习本可以买12本而无剩余，已知每个大练习本比小练习本贵0.32元，小明有多少元钱？
Translated Problem Text	Xiaoming took some money to buy exercise books in the store. If you buy a large exercise book, you can buy 8 books without surplus; if you buy a small exercise book, you can buy 12 books without surplus. It is known that each large exercise book is 0.32 yuan more expensive than the small exercise book. How much does Xiaoming have?
Templatized Postfix Equation with predicted operators	temp_0 temp_3 + temp_1 temp_1 + / temp_2 *
Actual Templatized Postfix Equation	temp_0 temp_3 - temp_1 temp_1 + / temp_2 *

Figure 5: Sample Output

$$accuracy\_e = \frac{\sum_{i=1}^{n} match\_e(p_i, c_i)}{\sum_{i=1}^{n} l_i} \qquad (1)$$

$$accuracy\_i = \frac{\sum_{i=1}^{n} match\_i(p_i, c_i)}{n} \qquad (2)$$

Here, $accuracy\_e$ refers to Element-wise accuracy, $accuracy\_i$ refers to Integrated accuracy, n is the total number of examples, p_i is the predicted operator list for the i^{th} example and c_i is the ground truth operator list for the i^{th} example, l_i is the number of operators in the equation of i^{th} example. Subroutine $match\_e(p, c)$ returns the total number of matching operators in lists p and c and $match\_i(p, c)$ returns 1 if all the operators of both the input lists match, otherwise it returns 0.

We assume that the tree structure of the post-fix template equation is available along with the input text. In the context of our problem framework, we assume that the first stage of the model has been implemented, and proceed to analyze our performance in the second stage. With this in mind, we evaluated our model using the above metrics. Preliminary results are shown in tables 2 and 3. To clarify notation, e refers to element-wise accuracy and i refers to integrated accuracy.

Our baseline model gave an element wise accuracy of 0.25 and integrated accuracy of 0.07 on test set. We see that our model has performed far

train_e	train_i	val_e	val_i	test_e	test_i
0.89	0.85	0.92	0.87	0.95	0.93

Table 2: Dolphin-S Performance Results

train_e	train_i	val_e	val_i	test_e	test_i
0.95	0.89	0.93	0.84	0.92	0.85

Table 3: Math23k Performance Results

better than this for both the datasets. We attribute these high accuracies to the fact that the structure of equation is available, which reduces the problem to predicting the operation to be performed between a pair of operands.

To train the model, we chose batch size as per dataset sizes - 128 for Math23k and 64 for Dolphin-S. We used training and validation loss curves to determine the hyper parameters that avoid overfitting of the model. Figures 6, 7, 8 and 9 show the loss and accuracy curves obtained when the model is trained with the following best set of hyper parameters. Embedding layer of size 100, hidden layer of size 160, 2 RNN layers in encoding and 20 epochs are the common hyper parameters.

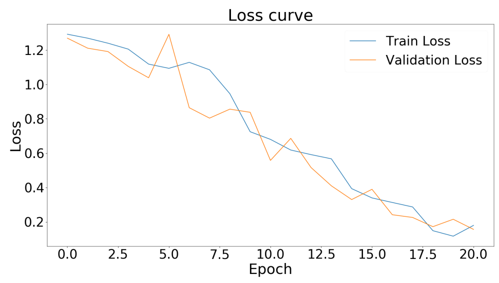


Figure 6: Dolphin-S Loss curve

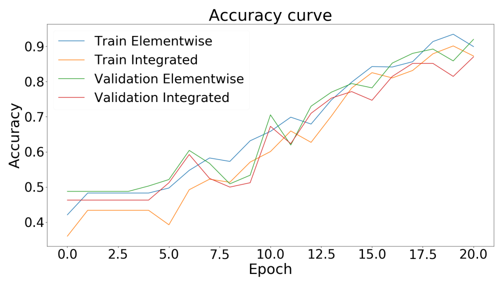


Figure 7: Dolphin-S Accuracy curve

3.4 Work Division

Every member started with the literature review and then shared ideas to come up with the plan and approach. Sheryl attempted to reach out to researchers from the Tencent AI Lab to acquire the Math23K dataset. Parallelly, Shrija and Ramya began exploring resources for dataset and collected them. Ramya, Sheryl and Shrija then worked on developing data preprocessing methods. While Shrija and Ramya were developing the second component of the T-RNN model, Sheryl worked on the templatization of equations. All three then combined everything and helped fill the gaps. Post this everyone tried different hyper parameter tuning and came up with the best results. All group members were involved in putting together this report.

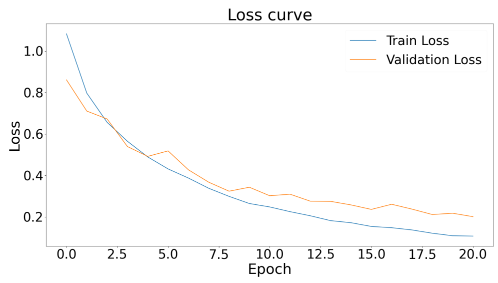


Figure 8: Math23K Loss curve

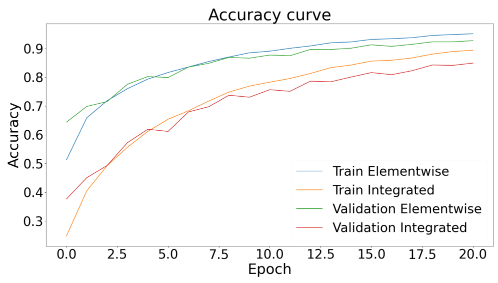


Figure 9: Math23K Accuracy curve

4 Plan for Completion

4.1 Future Tasks

We have a much better idea of the direction we are headed in for our project, now that we have obtained and cleaned the data, and know how we plan to proceed with the T-RNN model. In the short term, we expect to hone in on the completion of our model and building out the final components. If we have encouraging results, we plan to publish it in a reputed conference in the long term.

In the next stage of our project implementation, we revert back to the first component of the T-RNN model, structure prediction. Here, we cross-apply the seq2seq model to transform the input problem text into the tree structure, followed by suffix expression, in order to predict the tree-structure template. This serialization in the form of suffix expression will be critical for storing and representing the math equation. We also aim to apply equation normalization to reduce the space of equation templates. This will be directed at reducing order duplication and bracket duplication. We anticipate the seq2seq model transforming the problem text into suffix expression, which is then naturally processed with stack operations. Bi-LSTM serves as the encoder and LSTM serves the decoder. The output of the seq2seq model will then be passed to a sequence-to-sequence attention layer to more precisely capture the relationship between question words and the corresponding equation template. We plan on terminating the algorithm if an invalid equation template is generated. However, based on the literature, we are confident that the rate of invalid template generation is low (0.7% according to (Wang et al., 2019)).

Once we've implemented the entire T-RNN model, we will move on to incorporate the language agnostic word embedding layer, which introduces our novel contribution to the existing work in this field. As described in Subsection 3.2, we will build a joint model with the use of LASER, a language agnostic BiLSTM encoder and auxiliary decoder (Artetxe and Schwenk, 2019). We may also draw inspiration from USE, a multi-task dual-encoder framework (Yang et al., 2019). We also plan on developing a combined dataset with Dolphin-S and Math23K to test our language agnostic model and its ability to learn multilingual word problem/sentence embeddings.

We are working with subsets of Math23K and Dolphin-S that pertain to single variable equations, so we expect our complete model's performance to surpass the baseline model performance. Since this problem is a novel one, we are introducing a new baseline which will generate random equations/answer within the range of data. Moreover, since the available language agnostic embeddings are evaluated on retrieval tasks, we are not sure how they will perform on our MWP task. We expect that our accuracy may decrease relative to our current model accuracy, given that we will be predicting equation templates along with the operator. However, our short-term aim is to improve upon the accuracy of our baseline model. We will perform a thorough analysis of the accuracy loss/gain in the final report.

We intend to evaluate results with the same performance metrics we've analyzed at this stage of our progress. We will use both integrated accuracy and element-wise accuracy to assess the reliability of our results.

We have allocated the next two weeks for completing the design of the T-RNN model and integrating the language agnostic sentence embedding layer into the system. We will then immediately move to testing once we have completed all the components of our proposed model. We are reserving the third week for finalizing training/testing procedures, comparing against the baseline, analyzing results, and putting together our final project report.

4.2 Project Changes

At this point, we are getting progressively comfortable with remote collaboration given that our team members are located in different timezones. We do not anticipate changing our model design or architecture, and have a strong sense that we will complete it on time.

4.3 Work Division - Moving Forward

Now that we have a streamlined idea as to where our project is headed, we have assigned specific tasks to each of our group members. In the next half of the project, Shrija will build the equation normalization component of the template prediction stage. Sheryl will be responsible for cross-referencing and extrapolating the seq2seq model from (Wang et al., 2019). Ramya will integrate the above implementation with the language agnostic embedding layer and pass the output to an additional attention layer. To confirm the veracity of our results, we expect all group members to train and test the model locally against the baseline. We will then regroup to evaluate our results and compare performance across models and languages. All members will collaborate in synthesizing our experiment results and putting together the final report.

References

Mikel Artetxe and Holger Schwenk. 2019. Massively multilingual sentence embeddings for zero-shot cross-lingual transfer and beyond. *Transactions of the Association for Computational Linguistics*, 7:597–610.

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.

Lei Wang, Dongxiang Zhang, Jipeng Zhang, Xing Xu, Lianli Gao, Bing Tian Dai, and Heng Tao Shen. 2019. Template-based math word problem solvers with recursive neural networks. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence (Volume 33: No. 01)*, pages 7144–7151.

Yan Wang, Xiaojiang Liu, and Shuming Shi. 2017. Deep neural solver for math word problems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 845–854.

Yinfei Yang, Daniel Cer, Amin Ahmad, Mandy Guo, Jax Law, Noah Constant, Gustavo Hernandez Abrego, Steve Yuan, Chris Tar, Yun-Hsuan Sung, et al. 2019. Multilingual universal sentence encoder for semantic retrieval. *arXiv preprint arXiv:1907.04307*.