Project Milestone - Just-in-time nudging for word math problems in elementary school

Anant Mittal, Roshan Ramkeesoon, Sahil Verma
E-mail: anmittal@cs.washington.edu; roshanr@uw.edu vsahil@cs.washington.edu

1 Project Description

We are interested in improving the prediction accuracy for sequential linear formulas on the MathQA dataset [1]. Given a math problem in English, we want the model to generate a string of steps to be used in order to solve the problem. At each step, the model should generate an operation (eg. add, divide) and its operands which can either consist of values from the word problem or results of previous operations. We consider the problems which are solvable by the four basic operations, i.e., add, subtract, multiply and divide. The trained model can then be incorporated into the educational tools being used to teach word math problems in elementary schools. When a student gets stuck at a particular step in the problem, the educational tool can help the student with a hint of the operation (and operands) to be used at that step. This is quite established technique in education and is termed as nudging. With a good amount of accuracy such tools can be used to provide assistance to students for just-in-time nudging [2]. We hope to present a live notebook that can provide hints on such math problems.

2 Dataset

The MathQA dataset is a dataset of around 37k math questions (categorised), with rationale and linear formulas (algorithms) and answers to multiple choice questions. It is an extension of the AQuA dataset, but included they key difference is MathQA is gathered by using a new representation language to annotate over the AQuA-RAT dataset. MathQA has provided the questions, options, rationale, and the correct options. An example datapoint is:

Question: A train running at the speed of 48 km/hr crosses a pole in 9 seconds. What is the length of the train ?

Rationale: Speed = (48 x 5/18) m/sec = 40/3 m/sec . length of the train = speed x time. Length of the train = (40/3 x 9) m = 120 m.

Options: a) 140 , b) 130 , c) 120 , d) 170 , e) 160

Correct Option is: C

Category: Physics

Linear Formula:

- $multiply(n0, const\_1000)$
- $divide(\#0, const\_3600)$
- $multiply(\#1, n1)$

3 Methods

We pose this problem as NMT task and experimented with a variety of attention based Seq2Seq architectures [3]. In our encoder and decoder we kept the LSTM bidirectional and unidirectional respectively. We calculated BLEU-4 scores over the entire test set while modifying the number of LSTM layers in the encoder. As another modification, we replaced the embedding layer with BERT embeddings and trained our model with frozen as well as by fine-tuning the BERT layer. We used the pretrained model available from Hugging Face [4], [5].

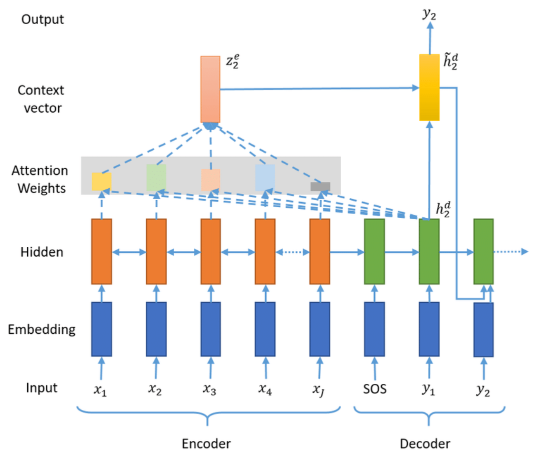

Figure 1: Diagrammatic representation of Seq2Seq architectures with attention

We tokenized the input word math problem into words, and for the output, we tokenized them into both words and characters. We start with simplest word to character model. Next we changed the tokenization of the steps to words, and hence trained a word to word model. In the above models, we were learning the embeddings. We replaced learning with BERT pre-trained models. We trained two versions, with frozen and fine-tuning BERT. Finally we also attempted to solve this using Transformers, but were unable to obtain a suitable result in time for submission.

4 Results

Figure 2 shows the loss plot while we trained our Seq2Seq model which had bidirectional single layer LSTM encoder

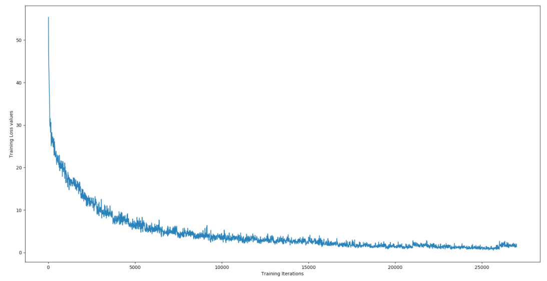

Figure 2: Train Loss Plot for Seq2Seq LSTM model with bidirectional single layer encode and unidirectional single layer decoder.

and single layer unidirectional decoder.

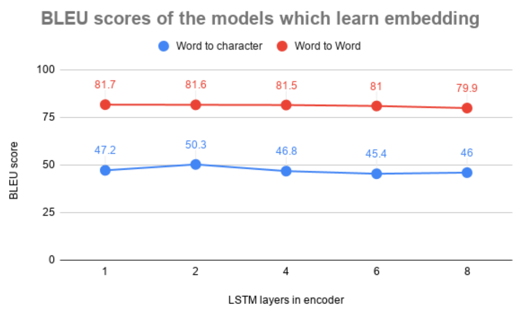

Figure 3: BLEU scores of the model which learn embedding layers.

Figure 3 shows the BLEU score for word to character and word to word based models. In both these cases, the entire model including the embedding layer was learnt. We got pretty **good** results with word to word models, with BLEU score reaching as high as **81.7** on the test set and **93.0** on the training set. Overall, we tried 5 architecture for word to character setting and 5 architectures for word to word setting.

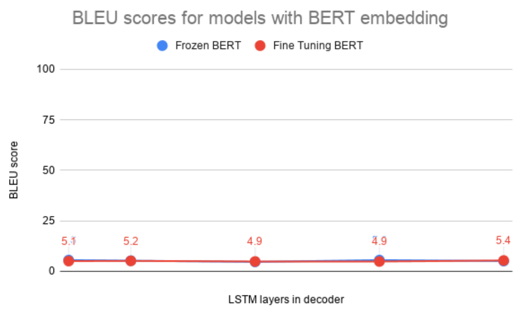

Figure 4: BLEU scores of the model which use BERT embeddings.

We thought that the word to word model from Figure 3 can be further improved if we use BERT pre-trained embeddings instead of learning them from scratch. So we tried two setting with BERT pre-trained embeddings, one in which BERT layer was frozen and another in which BERT was fine-tuned. Figure 4 shows the BLEU scores for the word to word model for the two cases, BERT layer frozen and BERT fine-tuned. Unexpectedly, both the experiments resulted in a dramatic decrease in BLEU score. We hypothesize that the task BERT models were trained on were very different from the formula prediction task we are addressing here and hence this huge gap was observed. Since the number of examples in our dataset (37k) is comparable to the size of dataset what BERT was originally trained on, therefore even fine-tuning BERT did not make lot of changes, and the BLEU score remained poor in this situation as well.

Overall, we tried 5 architectures with BERT layer frozen and 5 architectures with BERT layer fine-tuned.

Thus in total, we experimented with **20** architectures.

4.1 Sample predictions

`1. A shopkeeper sold an article offering a discount of 5% and earned a profit of 31.1%. What would have been the percentage of profit earned if no discount had been offered ?`

Ground truth:
add(n1,const_100)|subtract(const_100,n0)| multiply(#0,const_100)|divide(#2,#1)| subtract(#3,const_100)|

Prediction:
add(n1,const_100)|subtract(const_100,n0)| multiply(#0,const_100)|divide(#2,#1)| subtract(#3,const_100)|

`2. What least number must be subtracted from 3832 so that the remaining number is divisible by 5?`

Ground truth:
divide(n0,n1)|floor(#0)|multiply(n1,#1)|subtract(n0,#2)|

Prediction:
divide(n0,n1)|floor(#0)|multiply(n1,#1)|subtract(n0,#2)|s

5 Code

Our code is publicly available on GitHub. We implemented the Seq2Seq basic model which was based on a boilerplate code from Stanford NLP course (CS224N). All the architectures were obtained by modifying this code. We have organised the code into sub-directories, dependent on whether the embedding was learnt or BERT embedding and whether the output tokenization was made in character or word space.s

6 Conclusion

The attention-based seq2seq model predicting the annotated math formula as "words" using learned embeddings showed best performance of 81.7 BLEU score. The highest BLEU score for this architecture had a single LSTM encoder layer and decoder layer. Models with more LSTM layers resulted in lower BLEU score. This could be the result of overfitting the train set with larger models.

We found that predicting math formulas as characters resulted in lower BLEU scores than predicting as words. Whereas the number of unique characters in the formula vocabulary is 43, the number of unique words in the formula vocabulary is 145. In many NMT scenarios, word based models are difficult to train because of the large vocabulary of languages. However, in the context of predicting math formulas, we can take advantage of the increased semantic density of words without incurring a penalty from the large output space.

Using BERT embeddings for token embeddings in the encoder showed worse results than training new embeddings. As a result we next tried to fine-tune BERT. However, fine-tuning BERT also showed similarly poor results around a BLEU score of 5. The language of math word problems constitute a different context from the corpus BERT was trained on. The poor BLEU score could be the result of these differing contexts. In future studies it would be interesting to compare results if using a model that is trained on an unsupervised corpus of math data such as mathematics textbooks.

References

[1] A. Amini, S. Gabriel, S. Lin, R. Koncel-Kedziorski, Y. Choi, and H. Hajishirzi, "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)*. Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 2357–2367. [Online]. Available: https://www.aclweb.org/anthology/N19-1245

[2] M. T. Damgaard and H. S. Nielsen, "Nudging in education," *Economics of Education Review*, vol. 64, pp. 313–342, 2018.

[3] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," 2014.

[4] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," *CoRR*, vol. abs/1810.04805, 2018. [Online]. Available: http://arxiv.org/abs/1810.04805

[5] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, and J. Brew, "Huggingface's transformers: State-of-the-art natural language processing," 2019.