MIT 6.883 Meta Learning Due: Nov 16, 2020 11:59pm

Exercise 2: Learning to Solve Math Word Problems

Instructor: Prof. Iddo Drori Teaching Assistant: Zhongxia "Zee" Yan

The goal of this exercise is to gain practical experience with using Transformers and graph neural networks (GNN's) for solving math word problems. You will be filling in an implementation of one such method [16] using Hugging Face Transformers library [15] and the Deep Graph Library (DGL) [14]. A zip file for this exercise consists of exercise2.ipynb, demo.ipynb, and util.py (see a preview in exercise2.html), as well as the data in data/.

Please submit your best model prediction on the in-class Kaggle site and any written responses and code on Canvas Exercise 2 by Monday, November 16th. The exercise is in pairs, and each pair should submit one submission with both persons' names at the top of the exercise. You may either handwrite your responses or use LaTeX. For the coding part, you may submit your code in either the original .ipynb format, or submit it in .py format. The TA will moderate questions regarding the exercise on Piazza (please do not contact the papers authors for help).

1 Introduction

Very large Transformer models [3, 11, 10, 1, 4] have recently been applied to natural language processing, computer vision, and other tasks, resulting in significant performance gains across most common benchmarks, including question answering. Recent work on question answering demonstrates that Transformer models excel on benchmarks such as SuperGLUE [12]. However, while large Transformers perform with nearly 70% accuracy on topics such as foreign policy, Psychology, and Marketing, few-shot Transformer performance on STEM related topics such as Chemistry, Physics or Math are currently slightly above random [5] (we expect this to quickly change). Recent work [2, 16] targeting these tasks enhance the Transformer with a GNN to predict an expression tree whose evaluation solves these problems. Specifically, in this exercise you will complete and improve upon such a method, given an implementation, and fill in the missing lines of code.

2 Math Word Problems (MWP)

A math word problem described by sentences of text may be represented by an expression or equation. Here is an example math word problem:

Alice has 4 bags with the same amount of marbles in them, totaling 12 marbles. Bob has 3 bags with the same amount of marbles in them, totaling 18 marbles. How many more marbles does Bob have in each bag?

and solution: $x = (18/3) - (12/4) = 3$.

Several math word problem datasets have been introduced to assess the performance of recent Transformer models. In this exercise, you will use the MAWPS dataset [6], which contains 2,373 problems. Larger and recent datasets such as Math23K [13] and APE210K [17] are also widely used.
Please refer to demo.ipynb for examples using the MAWPS dataset.

3 Method

For this exercise, you are given an implementation based on *Graph-to-tree Learning for Solving Math Word Problems* [16], which uses multiple other building blocks. The basic implementation consists of a Seq2Seq

model, GNN, and expression-tree decoder. Feel free to read the paper before starting the exercise.

3.1 Base Seq2Seq Model

The base Seq2Seq model maps each token index in the input sentence to a hidden vector representation; this representation reflects the semantics of the token as well as its relation to other tokens in the question. The reference paper uses a BiLSTM model for this component, and here you will use custom Transformer implementation, or a pre-trained T5 encoder model as the base model.

3.2 Graph Neural Network

The graph neural network component takes in the graph relationships of the input tokens to perform graph convolution on the output of the base model. The two graphs used are (i) an undirected quantity cell graph and (ii) a directed quantity comparison graph. To generate the quantity cell graph, each quantity in the input is connected with attributes describing that quantity; to generate the quantity comparison graph, an edge connects quantity i to j iff $n_i > n_j$.

3.3 Expression Tree Decoding

Given the output of the GNN, the decoder layer generates each token of the expression conditioned on all previously generated tokens. Note that the decoder is fully implemented for you, so you may use the existing code, or optionally modify the decoder.

4 Assignment and Grading

1. Given the adjacency matrices of relationships between tokens in a math word problem, complete the given implementation of the graph neural network which uses DGL. Benchmark the performance of your code with at least three different combinations of hyperparameter settings.
2. Replace the given custom Transformer implementation with a Transformer using the Hugging Face library. Specifically, finetune a pre-trained T5 Transformer model to improve the results in the previous section. Again, use the Hugging Face transformers library to work with the T5 model, and we recommend that you start with the T5-Small model (60M parameters). Benchmark the performance with at least five different combinations of hyperparameter settings. Report your result, including accuracy, for each set of hyperparameter.
3. *Instead of 1 and 2*, you may choose to implement the generic Graph2Tree [8] based on the implementation available online [7] using the Hugging Face and DGL libraries.
4. Improve your performance in the previous sections either by a generic modifications such as ensembling or data augmentation; or by modifying the GNN, Transformer, or decoder; or combining components: i.e. using just a Transformer followed by GNN to output probabilities over expression templates.

Sign-up for the in-class Kaggle site as a team named foo_bar, where foo@mit.edu and bar@mit.edu are your MIT usernames. Submit your results on the in-class Kaggle site [9]. The exercise is evaluated by 5/10 points for your submitted code and document, and 5/10 points based on your Kaggle submission, score and leaderboard ranking.

References

[1] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu,

Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020.

[2] Kunlong Chen, Weidi Xu, Xingyi Cheng, Zou Xiaochuan, Yuyu Zhang, Le Song, Taifeng Wang, Yuan Qi, and Wei Chu. Question directed graph attention network for numerical reasoning over text. *arXiv preprint arXiv:2009.07448*, 2020.

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[4] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2020.

[5] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

[6] Rik Koncel-Kedziorski, Subhro Roy, Aida Amini, Nate Kushman, and Hannaneh Hajishirzi. 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, 2016.

[7] Shucheng Li, Lingfei Wu, Shiwei Feng, Fangli Xu, Fengyuan Xu, and Sheng Zhong. Graph-to-Tree implementation. `https://github.com/IBM/Graph2Tree`, 2020.

[8] Shucheng Li, Lingfei Wu, Shiwei Feng, Fangli Xu, Fengyuan Xu, and Sheng Zhong. Graph-to-tree neural networks for learning structured input-output translation with applications to semantic parsing and math word problem. *EMNLP*, 2020.

[9] Fall 2020 MIT Meta Learning course. Learning to solve math word problems leaderboard. `https://www.kaggle.com/c/mitmetalearningmawps`, 2020.

[10] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

[11] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020.

[12] Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. Superglue: A stickier benchmark for general-purpose language understanding systems. In *Advances in Neural Information Processing Systems*, pages 3266–3280, 2019.

[13] Lei Wang, Dongxiang Zhang, Jipeng Zhang, Xing Xu, Lianli Gao, Bing Tian Dai, and Heng Tao Shen. Template-based math word problem solvers with recursive neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7144–7151, 2019.

[14] Minjie Wang, Lingfan Yu, Da Zheng, Quan Gan, Yu Gai, Zihao Ye, Mufei Li, Jinjing Zhou, Qi Huang, Chao Ma, et al. Deep graph library: Towards efficient and scalable deep learning on graphs. *arXiv preprint arXiv:1909.01315*, 2019.

[15] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771, 2019.

[16] Jipeng Zhang, Lei Wang, Roy Ka-Wei Lee, Yi Bin, Yan Wang, Jie Shao, and Ee-Peng Lim. Graph-to-tree learning for solving math word problems. Association for Computational Linguistics, 2020.

[17] Wei Zhao, Mingyue Shang, Yang Liu, Liang Wang, and Jingming Liu. Ape210k: A large-scale and template-rich dataset of math word problems. *arXiv preprint arXiv:2009.11506*, 2020.