Can Neural Networks Learn Symbolic Rewriting? [*]

Bartosz Piotrowski[1,2], Chad E. Brown[1], Josef Urban[1], and Cezary Kaliszyk[3]

[1] Czech Technical University, Prague,
[2] University of Warsaw, Poland
[3] University of Innsbruck, Austria

Introduction

Neural networks (NNs) turned out to be very useful in several domains. In particular, one of the most spectacular advances achieved with use of NNs has been natural language processing. One of the tasks in this domain is translation between natural languages – neural machine translation (NMT) systems established here the state-of-the-art performance. Recently, NMT produced first encouraging results in the *autoformalization task* [6, 5, 4, 12] where given an *informal* mathematical text in LaTeX the goal is to translate it to its *formal* (computer understandable) counterpart. In particular, the NMT performance on a large synthetic LaTeX-to-Mizar dataset produced by a relatively sophisticated toolchain developed for several decades [1] is surprisingly good [12], indicating that neural networks can learn quite complicated algorithms. This inspired us to pose a question: *Can NMT models be used in the formal-to-formal setting?* In particular: *Can NMT models learn symbolic rewriting?*

The answer is relevant to various tasks in automated reasoning. For example, neural models could compete with symbolic methods such as inductive logic programming [10] (ILP) that have been previously experimented with to learn simple rewrite tasks and theorem-proving heuristics from large formal corpora [11]. Unlike (early) ILP, neural methods can however easily cope with large and rich datasets, without combinatorial explosion. Our work is also an inquiry into the capabilities of NNs as such, in the spirit of works like [3].

Data

To perform experiments answering our question we prepared two data sets – the first consists of examples found in ATP proofs in a mathematical domain (AIM loops), whereas the second is a synthetic set of polynomial terms – they are described below.

The AIM data set: The data consists of sets of ground and nonground rewrites that came from `Prover9` proofs of theorems about AIM loops produced by Veroff [7]. Many of the inferences in the proofs are paramodulations from an equation and have the form

$$\frac{s = t \qquad u[\theta(s)] = v}{u[\theta(t)] = v}$$

where s, t, u, v are terms and θ is a substitution. For the most common equations $s = t$, we gathered corresponding pairs of terms $\big(u[\theta(s)], u[\theta(t)]\big)$ which were rewritten from one to another with $s = t$. We put the pairs to separate buckets (depending on the corresponding $s = t$): in total 8 buckets for ground rewrites (where θ is trivial) and 12 for nonground ones. These constituted training sets for our experiments. Some examples from these data sets are presented in TPTP format in Table 1.

The polynomial data set: This is a synthetically created data set where the examples are pairs of equivalent polynomial terms. The first element of each pair is a polynomial in an

[*]Supported by the ERC Consolidator grant no. 649043 AI4REASON, and COST Action EUTypes CA15123, STSM no. 42217

Table 1: Examples in the AIM data set.

Rewrite rule:	Before rewriting:	After rewriting:
`b(s(e,v1),e) = v1`	`k(b(s(e,v1),e),v0)`	`k(v1,v0)`
`o(V0,e) = V0`	`t(v0,o(v1,o(v2,e)))`	`t(v0,o(v1,v2))`
`k(V0,k(V1,V2)) = k(V1,k(V0,V2))`	`l(k(v1,k(v0,v2)),k(v0,v2),v3)`	`l(k(v0,k(v1,v2)),k(v0,v2),v3)`

Table 2: Examples in the polynomial data set.

Before rewriting:	After rewriting:
`(x * (x + 1)) + 1`	`x ^ 2 + x + 1`
`(2 * y) + 1 + (y * y)`	`y ^ 2 + 2 * y + 1`
`(x + 2) * ((2 * x) + 1) + (y + 1)`	`2 * x ^ 2 + 5 * x + y + 3`

arbitrary form and the second element is the same polynomial in a normalized form. The arbitrary polynomials are created randomly in a recursive manner from a set of available (non-nullary) function symbols, variables and constants. First, one of the symbols is randomly chosen. If it is a constant or a variable it is returned and the process terminates. If a function symbol is chosen, its subterm(s) are constructed recursively in a similar way. Several data sets of various difficulty were created by varying the number of available symbols and the length of the polynomials. Each data set consists of 300000 examples, see Table 2 for examples.

Experiments

Several experiments were conducted using an established NMT implementation [8] with parameters inherited from [12]. The training terms were given to NMT as linear sequences of symbols. First, NMT models were trained for each of the 20 rewrite rules in the AIM data set. It turned out that the models, as long as the number of examples was greater than 1000, were able to learn the rewriting task with high accuracy – reaching 90% on separated test sets. On the joint set of all rewrite rules (consisting of 41396 examples) the performance was also good – 83%. This means that the task of applying single rewrite step seems relatively easy to learn by NMT. Then experiments on more challenging but also much larger data sets for polynomial normalization were performed. Depending on the difficulty of the data, accuracy on the test sets achieved in our experiments varied between 70% and 99%. Some of the results are shown in Table 3.

Conclusions and future work

NMT is not typically applied to symbolic problems, but surprisingly, it performed very well for both described tasks. The first one was easier in terms of complexity of the rewriting (only one application of a rewrite rule was performed) but the number of examples was quite limited. The second task involved more difficult rewriting – multiple different rewrite steps were performed to construct the examples. Nevertheless, provided many examples, NMT could learn normalizing polynomials.

This motivates us to extend our work in two directions. Firstly, more interesting and difficult rewriting problems for NMT need to be provided for better delineation of the strength of the NMT models. Secondly, we are going to develop and test new kinds of NMT models tailored for

Table 3: Choosen results of experiments with polynomials. (Characteristic of formulas concerns input polynomials.)

Function symbols	Constant symbols	Number of variables	Maximum length	Accuracy on test
$+, *$	$0, 1$	1	30	99.28%
$+, *$	$0, 1$	3	50	88.20%
$+, *$	$0, 1, 2, 3, 4, 5$	5	50	83.47%
$+, *, \hat{}$	$0, 1, 2$	3	50	71.81%

the problem of comprehending symbolic expressions. Specifically, we are going to implement an approach based on the idea of treeNN, which may be another effective approach for this kind of tasks [3, 9, 2].

References

[1] Grzegorz Bancerek, Adam Naumowicz, and Josef Urban. System description: XSL-based translator of Mizar to LaTeX. In Florian Rabe, William M. Farmer, Grant O. Passmore, and Abdou Youssef, editors, *Intelligent Computer Mathematics - 11th International Conference, CICM 2018, Hagenberg, Austria, August 13-17, 2018, Proceedings*, volume 11006 of *Lecture Notes in Computer Science*, pages 1–6. Springer, 2018.

[2] Saikat Chakraborty, Miltiadis Allamanis, and Baishakhi Ray. Tree2tree neural translation model for learning source code changes. *CoRR*, abs/1810.00314, 2018.

[3] Richard Evans, David Saxton, David Amos, Pushmeet Kohli, and Edward Grefenstette. Can neural networks understand logical entailment? In *International Conference on Learning Representations*, 2018.

[4] Cezary Kaliszyk, Josef Urban, and Jirí Vyskocil. Automating formalization by statistical and semantic parsing of mathematics. In Mauricio Ayala-Rincón and César A. Muñoz, editors, *Interactive Theorem Proving - 8th International Conference, ITP 2017, Brasília, Brazil, September 26-29, 2017, Proceedings*, volume 10499 of *Lecture Notes in Computer Science*, pages 12–27. Springer, 2017.

[5] Cezary Kaliszyk, Josef Urban, and Jiří Vyskočil. Learning to parse on aligned corpora (rough diamond). In Christian Urban and Xingyuan Zhang, editors, *Interactive Theorem Proving - 6th International Conference, ITP 2015, Nanjing, China, August 24-27, 2015, Proceedings*, volume 9236 of *Lecture Notes in Computer Science*, pages 227–233. Springer, 2015.

[6] Cezary Kaliszyk, Josef Urban, Jiří Vyskočil, and Herman Geuvers. Developing corpus-based translation methods between informal and formal mathematics: Project description. In Stephen M. Watt, James H. Davenport, Alan P. Sexton, Petr Sojka, and Josef Urban, editors, *Intelligent Computer Mathematics - International Conference, CICM 2014, Coimbra, Portugal, July 7-11, 2014. Proceedings*, volume 8543 of *LNCS*, pages 435–439. Springer, 2014.

[7] Michael K. Kinyon, Robert Veroff, and Petr Vojtechovský. Loops with abelian inner mapping groups: An application of automated deduction. In Maria Paola Bonacina and Mark E. Stickel, editors, *Automated Reasoning and Mathematics - Essays in Memory of William W. McCune*, volume 7788 of *Lecture Notes in Computer Science*, pages 151–164. Springer, 2013.

[8] Minh-Thang Luong, Eugene Brevdo, and Rui Zhao. Neural machine translation (seq2seq) tutorial. *https://github.com/tensorflow/nmt*, 2017.

[9] Ning Miao, Hengliang Wang, Ran Le, Chongyang Tao, Mingyue Shang, Rui Yan, and Dongyan Zhao. Tree2tree learning with memory unit, 2018.

[10] Stephen Muggleton and Luc De Raedt. Inductive logic programming: Theory and methods. *The Journal of Logic Programming*, 19:629–679, 1994.

[11] Josef Urban. Experimenting with machine learning in automatic theorem proving. Master's thesis, Faculty of Mathematics and Physics, Charles University, Prague, 1998. English summary at `https://www.ciirc.cvut.cz/~urbanjo3/MScThesisPaper.pdf`.

[12] Qingxiang Wang, Cezary Kaliszyk, and Josef Urban. First experiments with neural translation of informal to formal mathematics. In Florian Rabe, William M. Farmer, Grant O. Passmore, and Abdou Youssef, editors, *11th International Conference on Intelligent Computer Mathematics (CICM 2018)*, volume 11006 of *LNCS*, pages 255–270. Springer, 2018.