Evaluate and Switch between LaTeX and Natural Mathematical Expressions with Encoder-Decoder

Zeyun Wu
Department of Electrical and Computer Engineering
University of California, San Diego
La Jolla, CA 92093
zew038@ucsd.edu

Abstract

This paper introduces a pilot study of evaluating and translation between LaTeX and natural mathematical expressions. We implemented algorithms that generate LaTeX and natural expressions by defining rules and permutating elements. We utilized encoder-decoder networks with attention mechanism to translate LaTeX expression to natural expression with high performance. Some critiques and future directions were discussed, and the study was still preliminary.

1 Introduction

LaTeX (LaTeX) is a typesetting software system that allows users to gain scalable, systematic, and consistent presentation of contents. It is commonly used by people from all fields in various circumstances, including writing this paper. It is especially useful in formatting mathematical expressions. For instance, this is an example of well-organized presentation of LaTeX equation computing gradient during backpropagation when training neural network.

$$\nabla_{\mathbf{w}^l} L[\eta(x;\theta), y] = \frac{\partial L[\eta(x;\theta), y]}{\partial z^l} \frac{\partial z^l}{\partial \mathbf{w}^l}$$

While LaTeX is a great application for composings written documentation, during daily communication, we constantly use natural language expressions to describe our thoughts. It would be naturally interesting and useful to explore the following two major tasks: translation between LaTeX and natural expression, and evaluate whether a given pair of LaTeX and natural expressions is equivalent to each other. To fulfill these two tasks, we utilized encoder-decoder networks with attention, which is one of state-of-the-art algorithms in natural language translation.

Currently, there had been a lot of study about converting images of mathematical expression into LaTeX. However, there is no study about convertion between LaTeX and natural expressions yet. Moreover, there is no available public dataset for similar tasks. Therefore, we also developed methods to generate LaTeX and natural expression pairs.

2 Dataset

Currently, there is no public data that can be used in the tasks we would like to explore. Thus, we implemented methods to generate pairs of equivalent LaTeX and natural expressions (in English) from pre-defined rules and elements.

By default, we treated all LaTeX expressions as under math mode. We first defined limited number of possible operations including plus, minus, power, fraction, integral, etc, as well as elements (Table 1).

Preprint. Under review.

Table 1: Elements used in generating expression pairs

Types	Elements in Corresponding Type
Digit (for LaTeX)	0, 1, 2, 3, 4, 5, 6, 7, 8, 9
Number (for natural)	zero, one, two, three, four, five, six, seven, eight, nine
Variable	x, y, z

Table 2: Examples of rules for generating base and complex expression pairs. Numbers are placeholder representing elements to be filled. There can be several different natural expressions for the same LaTeX expression, which are separated by ';'.

Types	LaTeX rule	Natural rule
Base	0 \ge 1	0 is greater or equal to 1
Base	\frac{0}{1}	0 over 1; 0 divided by 1
Base	0\times 1	0 times 1; 1 multiplied by 0; product of 0 and 1
Complex	\int_{0}^ {1} h(2) \,d3	integral of h (2) from 0 to 1 with respect to 3
Complex	\lim_{0 \to 1} h(2)	limit of h (2) as 0 goes to 1

We then defined three types of rules to write expression pairs. By permutating elements in defined rules, we generated over 3000 pairs of equivalent expressions.

2.1 Define Expression Rules

There are three types of rules: base, complex, and nested. Some of the examples were listed in Table 2.

Base rules were the most basic and simple rules. They consisted of basic operations: plus, minus, multiplicatiom, fraction, power, and comparations. They had two or less parameters, and tended to have the most simple grammar in both LaTeX and natural expressions. Note there can be several different but equivalent natural expressions for one single LaTeX expression. For example, for multiplication, people would say one times two, or two multiplied by one, or product of one and two.

Complex rules generally were relatively complicated. They often had three or more parameters, and were with complex grammer. There were less number of complex rules than base rules.

Nested expressions were composed by nesting multiple base expressions or complex expressions. For base and complex rule, they were filled by elements, while nested expressions were filled by base or complex expressions. For example, we can have '\frac{x+1}{x} \ne 1' and 'x plus one over x is not equal to one' as a nested pair. Nested data was not used during training. Rather, they were used only during testing stage.

2.2 Data Generation

With defined rules, we generated expression pairs by permutating appropriate elements in corresponding placeholder positions. Generally all possible cartesian products of elements would be used to generate expression pairs. However, for positions of some rules, only one type of elemenets were legit. For example, in interal expression, the boundaries can only be filled with elements in Digit type (in LaTeX) or Number type (in natural), and the vairable can only be filled with elements in Variable type (see Table 1 and 2).

In total, we generated 3391 data pairs which would be used during training, along with several nested pairs. We shuffle all the expression pairs and split 80% of the pairs into training set, and rest into testing set. Again, nested expressions were not used during training.

3 Experiment

3.1 Tokenization

Before we can feed data into networks, we would need to define token for both LaTeX and natural languages. For natural expressions, tokens would be words separated by space. For LaTeX expressions,

Figure 1: Token cloud and word cloud plots for LaTeX and natural expressions

many symbols and space were meaningful in both semantic and syntax sense. Therefore instead of split by space, we defined token vocabulary, where the possible tokens were shown in Figure 1

We plotted token clouds for LaTeX and natural expressions, respectively (Figure 1). The font size represented the frequency that a certain token appeared in the dataset.

After tokenization, we converted expressions into indices for corresponding tokens that consisted of the expressions. Thus, we now had vector representation for each expression.

3.2 Encoder-Decoder with Attention

Since we had limited number of data, we constructed encoder-decoder networks with simple structure. The encoder had one hidden layer with 64 gated recurrent units. The decoder consisted of an attention layer, a softmax layer, and a hidden layer sized 64. The attention mechanism allowed the decoder to focus on part of the embedded information at each time.

We trained the networks with cross-entropy loss on decoder's output and teacher for 20000 randomly chosen data from training set. We recorded average loss for training and testing per 100 examples fed, respectively. During training, we changed the next input to decoder between the real target outputs from data and the output from itself with 50% probability.

4 Result

Figure 2: Training and testing loss curves versus training iteration

The loss of the network converged to small value for both training and testing (Figure 2).

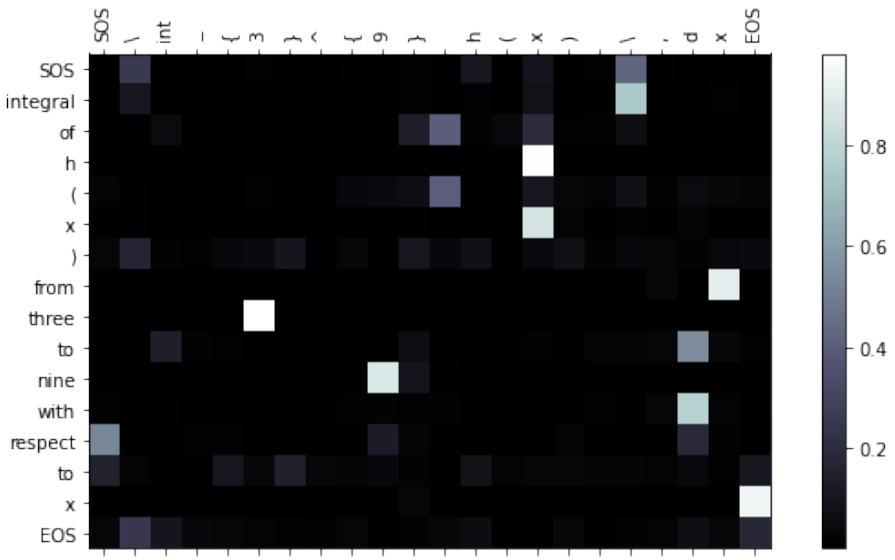

Figure 3: Heatmap of attention between input to encoder and output of decoder

To evaluate the network in translation task, we randomly sampled 20 expression pairs from testing set. The network showed great performance that it translated 17 of those samples perfectly, and the rest three with minor errors. Moreover, it can handle complex expressions correctly. We can also observe that attention was paid on correct spot from Figure 3

We also fed nested examples to examine its performance on new grammar by combining seen grammar. It failed to generate meaningful translation in this case. Instead, it generated some base expressions that were of highest frequencies.

5 Discussion

One major critique to current work would be that the dataset was generated by certain limited rules, rather than collected from real communication. This undermined the study in two senses. First, the data can be too clean and regular compared to real scenario though there were some variation in the rules. Second, it contained too less operations and syntax. At the end of the day, one can reasonably doubt that the network had more parameters than needed to simply learn the limited number of defined rule, which already known by us. Nevertheless, what we really wanted was some hidden grammar or correspondence from complicated daily language, which we might not easily discover explicitly. In the future study, on the one hand, we can enlarge our dataset by include more nested examples to add in complicity. On the other hand, we can also try to collect data from public mathematical discussion forum.

Moreover, the classifier that evaluates whether a input pair of LaTeX and natural expressions are equivalent has not been implemented yet. Once we see good traits from networks on better dataset, this implementation shall not be hard.

An additional intereting study would be to examinate embeddings of different natural expressions which corresponds to the same LaTeX expression. If the data become much more complicated, these non-bijective mapping may bring some difficulty. Ideally, we would like to see such embeddings to cluster together. Then one thing worth tring might be to force them cluster by adding contrastive loss term.

References

[1] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin. Attention Is All You Need. 2017. https://arxiv.org/abs/1706.03762.