Towards Mathematical Expression Understanding

Minh-Quoc Nghiem, Giovanni Yoko, Yuichiroh Matsubayashi and Akiko Aizawa
The Graduate University for Advanced Studies
National Institute of Informatics
nqminh,giovanni,y-matsu,aizawa@nii.ac.jp

Abstract

Mathematical expressions are an important means of scientific communication, used not only for numerical calculation and theorem proving but also for clarifying concept definitions and ensuring formal operations are not ambiguous. In this paper, we address the problem of understanding the meaning of mathematical expressions. In our approach, we formulate the problem as a task of translation from Presentation MathML to Content MathML expressions. We propose a machine learning-based approach that combines automatic extraction of fragment rules and statistical machine translation. Experimental results showed the potential of our approach for understanding the meaning of mathematical expressions.

1 Introduction

The digitization of mathematical and scientific content and its applications are attracting increasing interest in the scientific community. One of the most important tasks in this field is to make the computer *understand* the *meaning* of mathematical expressions. This enables automatic calculations so that the computer can solve mathematical problems. It also enables semantic searches for mathematical expressions by understanding the intent of the searcher and the contextual meaning of mathematical terms improve search accuracy. Moreover, this can enhance the accessibility for the visually impaired, for example, the computer can produce automatically synthesized speech output so people can correctly visualize the mathematical expressions.

As is the case with understanding language, understanding a mathematical expression is a non-trivial task. There are three main challenges to this task. The first is that there may be many concepts for a specific mathematical notation and vice versa. Correct interpretation of a notation must rely on an understanding of the context in which the notation appears and the context in which the whole mathematical expression appears. The second challenge is implicit grouping or invisible operators. For example, people can omit the multiplier operator and usually do so. The computer has to decide whether an operator is omitted or not. The last challenge is that notations tend to be introduced and used as and when needed. The author can reuse the notation later for different purposes. A computer, therefore, has to keep track of a notation definition in a document and recognize whether the notation appears again in the document or not. Of the above three main challenges, we address the first and second by proposing a framework based on machine learning methods.

In our task, we use Presentation MathML for displaying a mathematical expression and Content MathML to describe the semantic meaning of that expression. Our task then becomes translating from Presentation MathML to Content MathML. In this paper, we propose a machine learning-based approach that combines automatic extraction of fragment rules and statistical machine translation (SMT). We use a part of the data from the Wolfram Functions site for training and evaluation. The purpose of this study is to investigate the feasibility of machine learning-based approaches to the problem of understanding mathematical expressions. Experimental results show that our proposed system significantly outperforms a fundamental rule-based system.

The remainder of this paper is organized as follows: In Section 2, we give a brief overview of MathML and related approaches for understanding mathematical expressions, while in Section 3 we

present our proposed method. We then describe the experimental setup and results in Section 4. Section 5 concludes the paper and gives avenues for future work.

2 Related Work

Since mathematical formulas contain both mathematical symbols and structures, a special markup is required for their representation. Until recently, images have been used to represent mathematical formulas on the web. This type of display does not need any markup language to decode the formulas, but it is hard to process them. A way of dealing with mathematical formulas in this format is to convert them to another text-based format, as seen in InftyReader [2].

For scientific documents, TEX has been used to encode mathematical formulas. TEX is popular in academia, especially in mathematics since TEX provides a text syntax for mathematical formulas. The formula is printed in a way a person would write by hand, or typeset the equation. In Wikipedia, a formula is displayed in both image and TEX formats.

The best known open markup format for representing mathematical formulas for the web is MathML [4], and recommended by the W3C math working group, it provides a standard way of representing mathematical expressions. It is an XML application for describing mathematical notations and encoding mathematical content within a text format. MathML has two types of encoding, content-based encoding which is called Content MathML, dealing with the meaning of formulas, and presentation-based encoding which is called Presentation MathML, dealing with the display of formulas. In addition to MathML, there is another open markup format called OpenMath, but it is not as widely used.

Figure 1 shows an example of mathematical equation encoded in TEX, content-based MathML and presentation-based MathML.

For understanding mathematical expressions, Grigole et al. [3] proposed an approach based on the surrounding text of mathematical expressions. The main idea of this approach is to use the surrounding text for disambiguation which is based on word sense disambiguation and lexical similarity. First, a local context C (5 nouns preceding a target mathematical expression) is found in each sentence. For each noun, the system identifies a Term Cluster (TC) (derived from the OpenMath Content Dictionary) with the highest semantic similarity according to a similarity metric. The similarity scores obtained were weighted, summed up, and normalized by the length of the considered context. The assigned interpretation is the TC with the highest similarity score. The approach was evaluated on 451 manually annotated mathematical expressions and the best result was 68.26 $F_{0.5}$ score. To deal with the meanings of mathematical formulas, Nghiem et al. [7] proposed an approach for extracting the names or descriptions of the formulas using natural language text surrounding them. The most accurate extraction result was 68.33 percent.

There are two other projects that deal with the semantic meaning of mathematical expressions. The first is the SnuggleTeX project [5], which provides a free and open-source Java library for converting fragments of LaTeX to XML including Content MathML. The other project is Lamapun [6]. This project investigates semantic enrichment, structural semantics and ambiguity resolution in mathematical corpora. SnuggleTeX and Lamapun use rule based methods for disambiguation and translation. Rule based methods normally cost much human effort, therefore we propose a machine learning method.

3 The Approach

To translate mathematical expressions from the Presentation MathML to Content MathML format, a list of translation rules is required. Building these translation rules by hand is a large undertaking. Our task is inherently domain specific therefore we propose an approach which is based on machine learning

$ax^2 + bx + c$		
TeX format	**Content MathML**	**Presentation MathML**
a x ^ 2 + b x + c	<apply> <plus/> <apply> <times/> <ci>a</ci> <apply> <power/> <ci>x</ci> <cn>2</cn> </apply> </apply> <apply> <times/> <ci>b</ci> <ci>x</ci> </apply> <ci>c</ci> </apply>	<mrow> <mi>a</mi> <mo>.</mo> <msup> <mi>x</mi> <mn>2</mn> </msup> <mo>+</mo> <mi>b</mi> <mo>.</mo> <mi>x</mi> <mo>+</mo> <mi>c</mi> </mrow>

Figure 1: Example of different markup

methods that combine the fragment rule extraction and SMT. The target dataset are the mathematical expressions from the Wolfram Functions Site[1]. This site was created as a resource for educational, mathematical, and scientific communities. It contains the world's most encyclopedic collection of information about mathematical functions. All formulas on this site are available in both Presentation MathML and Content MathML format.

The framework of the system is shown in Figure 2. In the training phase, we use GIZA++ [1] for alignment between Presentation MathML terms and Content MathML terms. Based on the aligned data, we use an algorithm (described in the next section) to extract fragment rules. We then apply these rules to the data to obtain the fragmented data. The fragmented data are then used by SMT for the training step to obtain a translation model. In the running phase, given a set of Presentation MathML expressions to be translated, we apply the fragment rules to them to obtain the fragmented Presentation MathML expressions. We then apply a translation model to translate these fragmented expressions to Content MathML expressions and rebuild the trees to obtain the final translated Content MathML expressions. The purpose of the fragment rules is to simplify input trees (reduce the length and cross reordering) before they are used by SMT training.

[1] http://functions.wolfram.com/

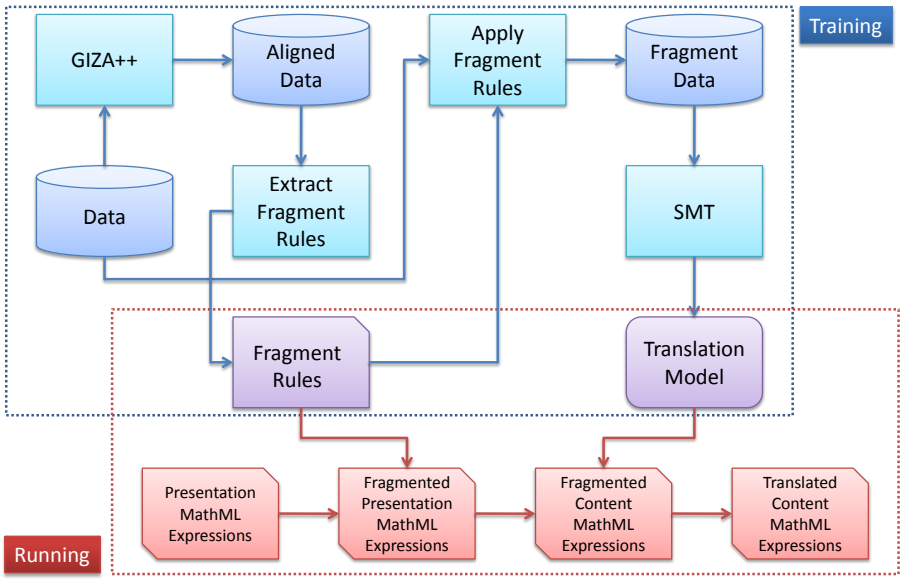

Figure 2: Framework of the proposed approach

3.1 Extracting Fragment Rules

Since there are many long mathematical expressions in real-world data and translating long and complex sentences has been a critical problem in machine translation, this step tries to break the expression down into smaller parts. Instead of translating the whole Presentation MathML tree, the system first divides the big tree into smaller trees. For the next step, the system translates these small trees from Presentation MathML to Content MathML and then groups them all together to form a complete Content MathML tree. Figure 3 illustrates this process. The purpose of this step is to reduce the length of the tree that is the input to the SMT system. It also reduces the number of cross alignments, hence reduces the number of reordering operations for the SMT system. Since the reordering complexity of SMT is exponential, this step greatly simplifies the translation step.

To separate a tree into fragments, the system first extracts all the possible fragment rules given a tree depth. After this step, a list of rules is generated, each of which is accompanied by its frequency. The rules that are not common are then filtered out (i.e. the number of appearances is less than a threshold). The purpose of the threshold is to filter out the rules created by uncommon or error data that could make the rule table smaller. Table 1 shows 5 example fragment rules and their frequency in our experimental corpus. The number in the rule indicates the order of alignment between the Presentation and Content MathML sub-trees.

The system then applies these rules to the corpus to fragment the mathematical expressions into smaller expressions. Based on the data alignment (output of GIZA++), a simple heuristic is applied to keep the parts of the Presentation MathML side aligned with the parts of the Content MathML side. It then extracts the rules from these fragmented expressions and applies them again to the corpus. The iteration stops when no new rule is added. The pseudo code of the algorithm is described in Algorithm 1.

3.2 Translation Model Generation

Once we have acquired a list of fragment rules, we apply them to the training corpus. Each mathematical expression is now fragmented to smaller expressions. After applying fragment rules, these small expres-

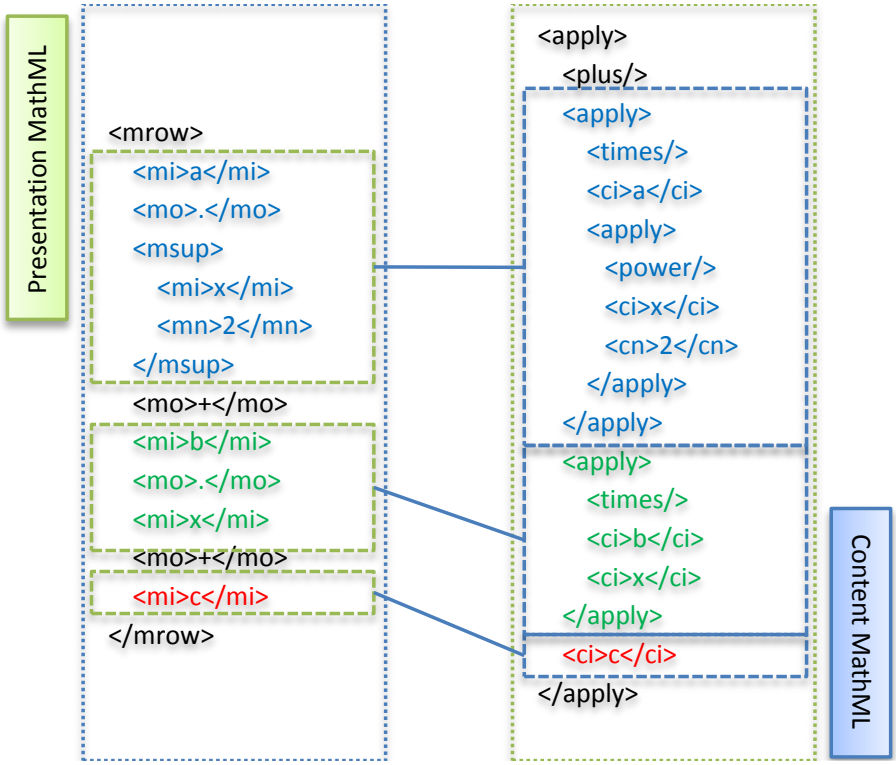

Figure 3: Fragmenting the tree of the expressions $ax^2 + bx + c$ to smaller parts

Table 1: Examples of fragment rules

Rule	**No.**
MROW ([1]MROW MO (|) [2]MROW) → APPLY (CI (condition) [1]APPLY [2]APPLY)	71
MROW ([1]MSQRT MO (==) [2]MROW) → APPLY (EQ [1]APPLY [2]APPLY)	29
MROW ([1]MSQRT (MO (.) [2]MSQRT) → APPLY (TIMES [1]APPLY [2]APPLY)	18
MROW ([1]MROW MO (∨) [2]MROW) → APPLY (OR [1]APPLY [2]APPLY)	13
MROW ([1]MROW MO (∧) [2]MROW) → APPLY (AND [1]APPLY [2]APPLY)	9

sions are fed into the statistical machine translation training framework. The SMT framework includes an aligner which automatically pairs up the relevant Presentation and Content MathML terms. It then learns the translation rules and their probability and finally generates a translation model. The result of this step is a translation model with a rule table that consists of all possible translation rules.

3.3 Translation from a Presentation MathML expression to a Content MathML expression

A Presentation MathML expression is translated to a Content MathML expression by the following steps:

Algorithm 1 Extract Fragment Rule

Input: a set of training MathML files parallel markup M
Output: a list of fragment rules R

$R \leftarrow \emptyset$
$A \leftarrow Alignment(M)$
repeat
  for all $m \in M$ **do**
    $r \leftarrow ExtractRule(m, A)$
    $R \leftarrow R \cup \{r\}$
  end for
  $R \leftarrow FilterRule(R, \theta)$
  $M \leftarrow ApplyRule(R, M)$
until $NewRule(R) = 0$
return R

- First, we fragment the Presentation MathML tree into sub-trees using fragment rules. The expressions are fragmented until they cannot be fragmented further by the fragment rules.
- Second, we translate the fragmented Presentation MathML sub-trees into Content MathML trees by using the decoder of the SMT system.
- Finally, we build the Content MathML tree from the translated sub-trees and the fragment rules obtained in the first step.

The translation algorithm is described in Algorithm 2.

4 Experimental Setup and Results

4.1 Data

We conducted an initial evaluation of the approach on all the mathematical expressions from the section "Elementary Functions" on the Wolfram Function site. To compare the results of small and large data, we also carried out the experiment on the "Sqrt" section of the "Elementary Functions". This section contains 220 mathematical expressions. After filtering out some error expressions (expressions that are in a Content MathML format but contain Presentation MathML notations), we have a total of 213 expressions in this sub-section.

We conducted two experiments. The first used the whole "Sqrt" section. The second used 2,000 random expressions from the "Elementary Functions" section. The data was divided into 20 parts, 18 for training, a part for developing (tuning parameters for the SMT system) and a part for testing. In total we had 22 and 199 expressions for testing.

4.2 Baseline and Evaluation Method

For comparison, we built another SMT system that used the original data from the Wolfram function set, i.e. the data without applying fragment rules. The evaluation metric was the Translation Error Rate [8] (TER). This is an error metric for machine translation that measures the number of edits required to change a system output into one of the references. For this metric, the smaller, the better.

Algorithm 2 Translate Presentation to Content MathML tree

Input: a Presentation MathML tree tp
a list of fragment rules R
a translation model TM
Output: a Content MathML tree tc

```
L ← ∅
TP ← {tp}
while TP ≠ ∅ do
  for all t ∈ TP do
    TP ← TP \ {t}
    if CanNotApplyRule(t, R) then
      L ← L ∪ {t}
    else
      TP = TP ∪ ApplyRule(t, R)
    end if
  end for
end while
L' ← ∅
for all l ∈ L do
  L' ← L' ∪ Translate(l, TM)
end for
tc ← RebuildTree(L', R)
return tc
```

4.3 Results

In our experiments, we set a threshold of 5 to filter uncommon rules (we archived the best result with this threshold). Table 2 presents the data statistics and Table 3 presents the results of SMT, SMT + Fragment Rules and SnuggleTeX on the test data.

Table 2: Data statistic

Data	Sqrt Section	Elementary Functions Section
Training data	169	1,589
Testing data	22	199
Total data	213	1,788
Fragment rules extracted	42	68

Table 3: Translation Error Rate of the systems

TER	Sqrt Section	Elementary Functions Section
SMT	10.25	60.35
SMT + Fragment Rules	13.18	51.62

For small and specific data ("Sqrt" section), both systems achieved excellent results but the SMT system had better results. The reason for the excellent results was that the data in this section was all about one topic. In larger and sparser data, both systems had many errors. By using fragment rules, the error rate was reduced by 10 percent. This is a significant improvement. There are two main reasons

for the poorer result of the SMT system. The first is there are many long mathematical expressions in the Wolfram function data while translating long and complex sentences has been a critical problem in machine translation. The second is that we need long-distance reordering for translating but SMT systems are not very good at this. The TER score of SnuggleTeX on both sets of data is 86.56 percent. This is because when SnuggleTeX cannot translate, it outputs error codes instead of the translated Content MathML trees. Another reason for the high error rate is SnuggleTeX uses different symbols with the Wolfram Functions.

5 Conclusions

In this paper, we discussed the problem of understanding mathematical expressions and our task was to translate from Presentation MathML to Content MathML. Despite being able to apply this only to specific mathematical notations in this paper, our preliminary experimental results show that our approach based on the statistical machine translation method, as well as using fragment rules, has the potential for translating a Presentation MathML expression to a Content MathML expression.

Since this is a first attempt to translate Presentation to Content MathML using a machine learning method, there is room for further improvement. Possible improvements are: (1) Increasing the training data so the system can cover more mathematical notations; (2) Expanding the work by incorporating the surrounding information of mathematical expressions, for example definitions or other mathematical expressions.

By combining the automatic extraction of fragment rules and statistical machine translation, our approach has shown promising results. The experimental results confirm that this approach is helpful to the understanding of mathematical expressions. However, this is only a first step; many important issues remain for future studies. Currently, our system deals only with a sub-part of mathematical notations. In future work, we should also consider expanding it to cover all mathematical notations.

References

[1] Franz Josef Och, Hermann Ney. “A Systematic Comparison of Various Statistical Alignment Models”, Computational Linguistics, volume 29, number 1, pp. 19-51 March 2003.

[2] Masakazu Suzuki, Toshihiro Kanahori, Nobuyuki Ohtake and Katsuhito Yamaguchi. “An integrated OCR software for mathematical documents and its output with accessibility”, ICCHP ’04, LNCS, vol. 3118, pp 648–655, 2004.

[3] Mihai Grigore, Magdalena Wolska and Michael Kohlhase. ”Towards Context-Based Disambiguation of Mathematical Expressions”, The Joint Conference of ASCM 2009 and MACIS 2009: Asian Symposium on Computer Mathematics and Mathematical Aspects of Computer and Information Sciences, pp. 262-271, December 2009.

[4] World Wide Web Consortium. “Mathematical markup language”, http://www.w3.org/Math/, 2011.

[5] David McKain. ”SnuggleTeX”, http://www2.ph.ed.ac.uk/snuggletex/, 2011.

[6] Deyan Ginev, Constantin Jucovschi, Stefan Anca, Mihai Grigore, Catalin David and Michael Kohlhase. “An Architecture for Linguistic and Semantic Analysis on the arXMLiv Corpus”, Applications of Semantic Technologies (AST) Workshop at Informatik 2009, 2009.

[7] M.Q. Nghiem, K. Yokoi, Y. Matsubayashi, and A. Aizawa. “Mining coreference relations between formulas and text using Wikipedia”, 2nd Workshop on NLP Challenges in the Information Explosion Era (NLPIX), pages 69-74, 2010.

[8] Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul, “A Study of Translation Edit Rate with Targeted Human Annotation”, Proceedings of Association for Machine Translation in the Americas, 2006.