Classifying MathML Expressions by Multilayer Perceptron

著者（英）	Yuma NAGAO, Nobutaka SUZUKI
journal or publication title	IEICE Transactions on Information and Systems
volume	E101.D
number	7
page range	1954-1958
year	2018-07
権利	(C)2018 The Institute of Electronics, Information and Communication Engineers
URL	http://hdl.handle.net/2241/00155504

doi: 10.1587/transinf.2017EDL8211

IEICE TRANS. INF. & SYST., VOL.E101–D, NO.7 JULY 2018

LETTER

Classifying MathML Expressions by Multilayer Perceptron*

Yuma NAGAO[†a)], *Nonmember and* **Nobutaka SUZUKI**[††], *Member*

SUMMARY MathML is a standard markup language for describing math expressions. MathML consists of two sets of elements: Presentation Markup and Content Markup. The former is widely used to display math expressions in Web pages, while the latter is more suited to the calculation of math expressions. In this letter, we focus on the former and consider classifying Presentation MathML expressions. Identifying the classes of given Presentation MathML expressions is helpful for several applications, e.g., Presentation to Content MathML conversion, text-to-speech, and so on. We propose a method for classifying Presentation MathML expressions by using multilayer perceptron. Experimental results show that our method classifies MathML expressions with high accuracy.

key words: *MathML, classification, multilayer perceptron*

1. Introduction

MathML is a standard markup language for describing math expressions. MathML consists of two set of elements: Presentation Markup and Content Markup. The former describes layout structures of math expressions, and is widely used to display math expressions in Web pages. On the other hand, the latter describes semantic meanings of math expressions, and is suited to automatic calculation of math expressions.

In this letter, we focus on the former and consider classifying Presentation MathML expressions, i.e., given a Presentation MathML expression e, identify the class (e.g., hypergeometric function, bessel-type function, etc.) that e should belong to. If we can identify the class of a given Presentation MathML expression automatically, it is helpful for several applications, e.g., Presentation to Content MathML conversion, text-to-speech, and so on. This is because there are a number of multi-meaning symbols and expressions in mathematics, and the class of a MathML expression can be a clue to resolve such an ambiguity. For example, consider the following two expressions.

$$C_n = \frac{1}{n+1}\binom{2n}{n} \qquad C(x) = \int_0^x \cos(t^2)\,dt$$

The left is Catalan number and the right is Fresnel integrals, and both expressions contains the same symbol "C". The symbol is expressed by `<mi>C</mi>` element in Presentation MathML for both "C" symbols. On the other hand, in Content MathML, `<ci>Catalan</ci>` element is used for "C" of the left expression while `<ci>Fresnel</ci>` element is used for "C" of the right expression. The wolfram function site [7] classifies the left expression in "Catalan" class and the right expression in "Fresnel integrals" class. Thus, the classes can be helpful to determine which element should be used for "C".

Our method classifies MathML expressions by using multilayer perceptron, which is a basic model of deep learning. The difficulty in taking such an approach is that MathML expression is tree structured data while multilayer perceptron requires "flat" vectors as input, and it is not clear how to convert MathML expressions into vectors suitable for classification by multilayer perceptron. To address this, our method converts a Presentation MathML expression into a vector which is based on binary branch vector [2] and our dimensionality reduction method. Experimental results show that our method classifies MathML expressions with high accuracy.

Related Work

Kim et al. [3] propose a classification method for Presentation MathML expressions. They extract features from MathML expressions and classify them by using support vector machine (SVM). They use labels of nodes and contiguous sequence of leaf nodes as a feature, in which parent-child relationships between elements are not considered. Moreover, SVM requires proper features which must be extracted according to characteristics of data manually. On the other hand, multilayer perceptron (and other deep learning models) does not require such manual feature extractions.

Besides MathML classification, several studies on semantic extraction from math expressions [4]–[6] have been made. [4] extracts semantic meanings of math identifiers from math expressions and texts surrounding the expressions. [5] proposes a method for specifying meanings of math identifiers. The method uses classes of math expressions as features. On the other hand, we estimate the classes of math expressions without using texts surrounding the expressions. [6] proposes a method for classifying math documents.

Manuscript received September 26, 2017.
Manuscript revised March 1, 2018.
Manuscript publicized April 4, 2018.

[†]The author is with Graduate School of Library, Information and Media Studies, University of Tsukuba, Tsukuba-shi, 305–8550 Japan.
[††]The author is with Faculty of Library, Information and Media Science, University of Tsukuba, Tsukuba-shi, 305–8550 Japan.

[*]This paper is based on [1].

a) E-mail: ynagao@klis.tsukuba.ac.jp

DOI: 10.1587/transinf.2017EDL8211

2. Proposed Method

Our method classifies Presentation MathML expressions by the following steps.

1. Convert Presentation MathML expressions into binary trees.
2. Generate binary branch vectors from the binary trees obtained in step 1.
3. Reduce the size of the binary branch vectors obtained in step 2.
4. Train multilayer perceptron by using the vectors obtained in step 3, and classify MathML expressions by using the multilayer perceptron.

In the following, we give the details of the above steps.

Step 1: Binarization of MathML Expression

MathML expressions are represented as unranked ordered trees. We convert them into full binary trees.

An *unranked ordered tree* is denoted $T = (N, E, Root(T))$, where N is the set of nodes, E is the set of edges, and $Root(T)$ is the root node of T. By $(u, v) \in E$ we mean an edge from parent node u to child node v.

A *full binary tree* (*binary tree*, for short) is an ordered tree such that every node has either zero or two child nodes. By $B(T)$, we mean the binary tree of an unranked ordered tree T. Here, $B(T)$ is obtained from T by the well-known binarization method for unranked ordered tree, as follows.

1. Initially, $B(T)$ consists of the same set of nodes as T and no edges.
2. For each node u in T, do the following.
    a. Let $v_1, v_2, \ldots, v_n$ be the child nodes of u. Add an edge (v_{i-1}, v_i) to $B(T)$ for every $2 \leq i \leq n$.
    b. Add an edge (u, v_1) to $B(T)$.
3. Insert empty nodes ϵ so that every internal node in $B(T)$ has exactly two child nodes.

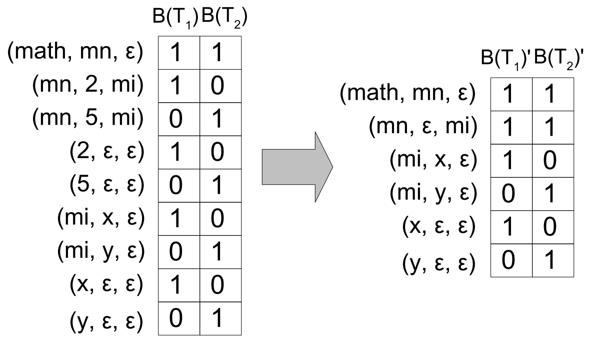
For example, $B(T_1)$ and $B(T_2)$ in Fig. 1 (center) are the binary trees of T_1 and T_2 in Fig. 1 (left), respectively.

Steps 2: Vectorization of Binary Tree by Binary Branch Vector

In this step, we vectorize binary trees as binary branch vectors [2].

A *binary branch* is a one level subtree of a binary tree. A binary branch is denoted (u, u_l, u_r), where u is the *root* of the binary branch, u_l is the left child of u, and u_r is the right child of u. For example, $B(T_1)$ in Fig. 1 (center) consists of five binary branches (math, mn, ϵ), (mn, 2, mi), (2, ϵ, ϵ), (mi, x, ϵ), and (x, ϵ, ϵ). Let S be a set of binary trees and $T \in S$. Then the *binary branch vector* of T, denoted $BRV(T)$, is defined as $BRV(T) = (b_1, b_2, \ldots, b_{|\Gamma|})$, where Γ is an ordered set of binary branches in S, $|\Gamma|$ is the size of Γ, and b_i is the number of occurrences of the ith binary branch in $B(T)$.

For example, consider binary trees $B(T_1)$ and $B(T_2)$ in Fig. 1 (center). The binary branch vectors of $B(T_1)$ and $B(T_2)$ are shown in Fig. 2 (left). In the figure, the two columns of the array represent binary branch vectors of $B(T_1)$ and $B(T_2)$, respectively, i.e., $BRT(B(T_1)) = (1, 1, 0, 1, 0, 1, 0, 1, 0)$ and $BRT(B(T_2)) = (1, 0, 1, 0, 1, 0, 1, 0, 1)$. Each cell in the array represents the number of occurrences of a binary branch. For example, binary branch (mn, 2, mi) occurs once in $B(T_1)$ but zero times in $B(T_2)$.



	B(T₁)	B(T₂)
(math, mn, ε)	1	1
(mn, 2, mi)	1	0
(mn, 5, mi)	0	1
(2, ε, ε)	1	0
(5, ε, ε)	0	1
(mi, x, ε)	1	0
(mi, y, ε)	0	1
(x, ε, ε)	1	0
(y, ε, ε)	0	1

	B(T₁)'	B(T₂)'
(math, mn, ε)	1	1
(mn, ε, mi)	1	1
(mi, x, ε)	1	0
(mi, y, ε)	0	1
(x, ε, ε)	1	0
(y, ε, ε)	0	1

Fig. 2 Dimensionality reduction of binary branch vectors.

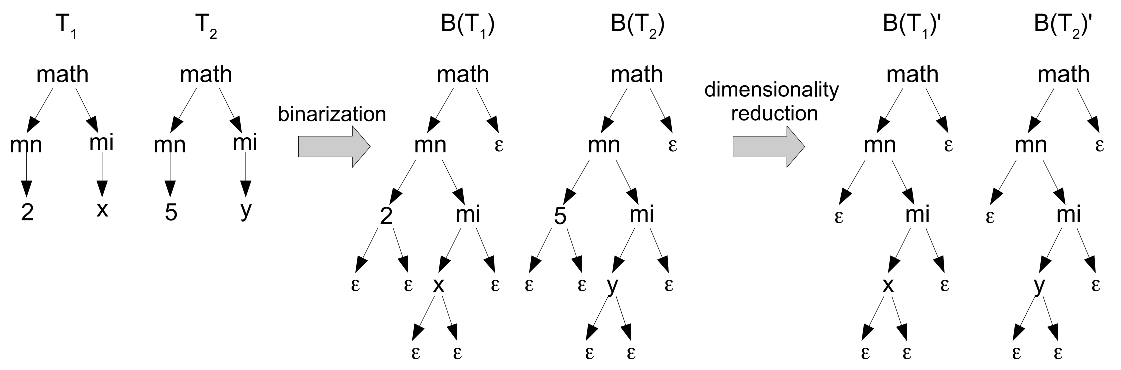


Fig. 1 Tree binarization and dimensionality reduction.

Step 3: Dimensionality Reduction of Binary Branch Vector

Each element of a binary branch vector is the number of occurrences of a binary branch in a given set S of binary trees. Therefore, as S becomes larger, the size of Γ increases accordingly. Actually, the size of Γ tends to become considerably large since MathML expressions have substructures which are similar but have different numeric values. However, in most cases, classes of MathML expressions can be captured by their structural features (tree structures and element names) rather than numerical values. Thus, by dropping such numerical values we can effectively reduce the size of binary branch vectors.

In our method, we reduce the sizes of binary branch vectors by erasing the text of `mn` elements. In Presentation MathML, `mn` elements represent numeric numbers. We give an example of dimensionality reduction. Consider the binary trees $B(T_1)$ and $B(T_2)$ in Fig. 1 (center). We replace the texts of `mn` elements (2 and 5) with ϵ. As the result, binary branches (mn, 2, mi) and (mn, 5, mi) are converted into the same binary branch (mn, ϵ, mi) and two binary branches (2, ϵ, ϵ) and (5, ϵ, ϵ) are dropped (Fig. 1 (right)). Figure 2 shows an example of binary branch vectors before and after this dimensionality reduction. As shown in the figure, the size of binary branch vector is reduced from 9 to 6.

Steps 4: Classification of MathML Expressions by Multilayer Perceptron

Our method classifies vectors obtained in step 3 by a multilayer perceptron.

A multilayer perceptron is composed of an input layer, any number of hidden layers and an output layer. Our model consists of an input layer, two hidden layers, and an output layer. The number of units in the input layer is same as the size of vectors of a given dataset, obtained in step 3. Each hidden layer has 512 units and activated by ReLU. The output layer has the same number of units as MathML classes of a given dataset and activate by softmax. We use Adam as the optimizer to minimize cross entropy error.

3. Experiments

In this section, we present our experimental results.

3.1 Dataset

In our experiments, we used two datasets: the Wolfram Function Site and MREC.

The Wolfram Function Site:

We collected HTML files exhaustively and extracted Presentation MathML expressions from the wolfram function site [7]. We obtained 307,676 expressions and 14 classes. We used 70% of them for training data and used 30% for test data.

MREC (Mathematical REtrieval Collection):

MREC [8] is a dataset of scientific papers in arxiv.org translated to XML. The math expressions in the papers are written in Presentation MathML.

MREC contains too short expressions (e.g., “x”, “Δ”) for which classification is meaningless. Therefore, we removed such MathML expressions. Specifically, we removed MathML expressions that have no `mi` elements or less than two elements except `math` and `mrow` elements from the dataset. Note that `math` is the root element of a MathML expression and that `mrow` elements affect neither the display nor the meaning of any MathML expression.

The MREC dataset has originally 34 classes, and we chose classes containing enough number of expressions. Specifically, we chose 20 classes having more than 100,000 expressions. The hierarchy of the 20 classes is shown in Fig. 3 (the two slanted classes *Mathematics* and *Physics* are “empty” classes having no their own documents). As shown in the figure, the hierarchy consists of three tiers: “top-class”, “sub-class”, and “sub-sub-class”. Since some classes are very similar to or overlapping each other, we merged such classes and obtain the 12 classes. Specifically, we merged classes by the following criteria.

- Sub-sub classes and their parent are merged ((2) and (9)).
- Classes having the same field name with supplemental words are merged ((7) and (10))

We extracted 100,000 MathML expressions from each of the 12 classes, which constitutes the final dataset. We used 70% of them for training data and used 30% for test data.

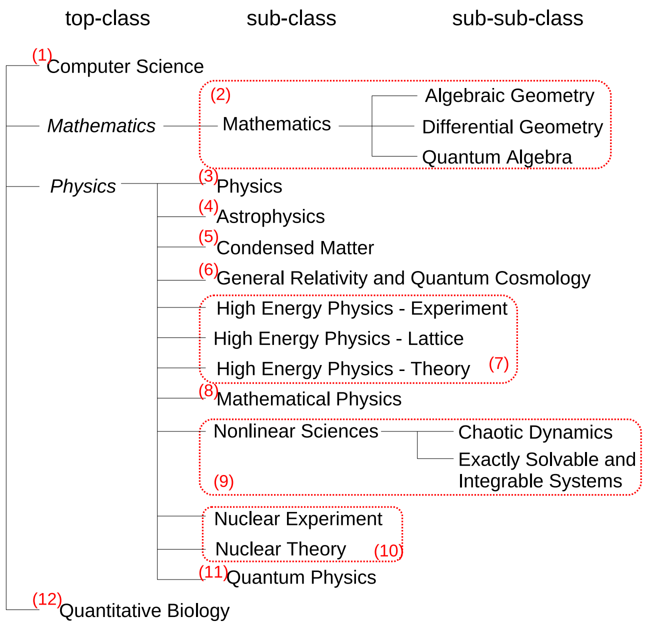


Fig. 3 Class hierarchy of 20 MREC classes.

3.2 Experimental Results

We compared our method, our method without dimensionality reduction, and the SVM-based method proposed by Kim et al. [3]. They define five features: labels of nodes (Tag), texts of `mo` elements which represent operators (Operator), texts of `mi` elements which represent identifiers (Identifier), bigram of plain text in expressions (String Bigram), and bigram of identifier and operator (I&O). To classify Presentation MathML expressions, they compared several combinations of the features as the inputs of SVM with liner kernel. In their experimental results, the combination of Tag, Operator, String Bigram, and I&O shows the highest accuracy. However, the expressions in our datasets contain few plain texts that can be used as the String Bigram feature. Therefore, we use Tag, Operator, Identifier, and I&O, which marked the second highest accuracy in their experiment. We adjusted penalty parameter C of SVM ($C = 2^{-10}, 2^{-9}, \ldots, 2^{10}$) since the value of C is not given in their paper. Besides the SVM-based method, we also tested our method without the dimensionality reduction.

Tables 1 and 2 show the results. Table 1 shows the accuracies of the three methods for the wolfram function site dataset. The accuracy of the SVM-based method is the value for $C = 2^8$, which brings the highest accuracy among the values of C. The result shows that any of methods achieve high accuracy. Further, our method slightly outperforms the SVM-based method. The possible reason why such high accuracies are obtained is that the dataset is highly "clean", that is, it has well-formed structures (e.g., order of variables, operators, etc.) and unified notations of identifiers.

Table 2 shows the accuracies of the three methods for the MREC dataset. The accuracy of the SVM-based method is the value for $C = 2^0$, which brings the highest accuracy among the values of C. For this dataset, our method clearly outperforms the SVM-based method. On the other hand, any of the accuracies are lower than those of the wolfram function site. This is because the expressions of the MREC dataset are much less "clean" than these of the wolfram function site dataset. For example, in the MREC dataset, the notation of identifiers are not unified, because the expressions are written by various authors. The result means that our method is much robust and effective to less "clean" expressions than the SVM-based method. Another interesting result for this dataset is that the accuracy of our method significantly drops without dimensionality reduction. A possible reason for this is that binary branches with numerical values act as "noises" that prevent our model from classifying math expressions correctly. Detailed reason is still under investigation and left as a future work.

Finally, let us present the sizes of input vectors. For the wolfram function site dataset, the size of binary branch vector is reduced from 558,777 to 2,184 by the dimensionality reduction method. For the MREC dataset, the size of binary branch vector is reduced from 72,019 to 39,833.

Consequently, the combination of multilayer perceptron and dimensionality reduced input vectors can be helpful to improve the accuracy of classification of MathML expressions.

Table 1 Accuracy for the wolfram function site dataset.

Method	Accuracy
Our method	**99.35**
Our method (without dimensionality reduction)	99.24
SVM ($C = 2^8$)	99.03

Table 2 Accuracy for the MREC dataset.

Method	Accuracy
Our method	**83.54**
Our method (without dimensionality reduction)	33.75
SVM ($C = 2^0$)	28.16

4. Conclusion

In this letter, we proposed a method for classifying Presentation MathML expressions based on multilayer perceptron. Experimental results showed that our method can classify MathML expressions with higher accuracy than the SVM-based method.

As a future work, we would like to investigate the effectiveness of our dimensionality reduction method, in terms of accuracy of classification. Furthermore, we need to consider comparing our model with other complex models (e.g., Tree-LSTM [9]). We also have to tune hyperparameters (number of units and layers, etc.) of our model, since it is not clear whether the hyperparameters used in our experiment are optimum or not.

References

[1] Y. Nagao and N. Suzuki, "Classification of MathML expressions using multilayer perceptron," Proc. 2017 ACM Symposium on Document Engineering, DocEng '17, pp.133–136, 2017.

[2] R. Yang, P. Kalnis, and A.K.H. Tung, "Similarity evaluation on tree-structured data," Proc. 2005 ACM SIGMOD International Conference on Management of Data, SIGMOD '05, pp.754–765, 2005.

[3] S. Kim, S. Yang, and Y. Ko, "Classifying mathematical expressions written in MathML," IEICE Trans. Inf. & Syst., vol.E95-D, no.10, pp.2560–2563, Oct. 2012.

[4] M. Schubotz, A. Grigorev, M. Leich, H.S. Cohl, N. Meuschke, B. Gipp, A.S. Youssef, and V. Markl, "Semantification of identifiers in mathematics for better math information retrieval," Proc. 39th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '16, pp.135–144, 2016.

[5] M.-Q. Nghiem, G.Y. Kristianto, and A. Aizawa, "Using MathML parallel markup corpora for semantic enrichment of mathematical expressions," IEICE Trans. Inf. & Syst., vol.E96-D, no.8, pp.1707–1715, Aug. 2013.

[6] T.T. Nguyen, K. Chang, and S.C. Hui, "Adaptive two-view online learning for math topic classification," Proc. 2012 European Conference on Machine Learning and Knowledge Discovery in Databases, ECML PKDD '12, Lecture Notes in Computer Science, vol.7523, pp.794–809, Springer, Berlin, Heidelberg, 2012.

[7] Wolfram Research, "The Wolfram Function Site," http://functions.wolfram.com/

[8] M. Líška, P. Sojka, M. Růžička, and P. Mravec, “Web interface and collection for mathematical retrieval: Webmias and mrec,” Towards a Digital Mathematics Library, ed. P. Sojka and T. Bouche, Bertinoro, Italy, pp.77–84, July 2011.

[9] K.S. Tai, R. Socher, and C.D. Manning, “Improved semantic representations from tree-structured long short-term memory networks,” CoRR, vol.abs/1503.00075, 2015.