IEICE TRANS. INF. & SYST., VOL.E95–D, NO.10 OCTOBER 2012

LETTER

Classifying Mathematical Expressions Written in MathML

Shinil KIM[†], *Nonmember*, **Seon YANG**[†], *Student Member*, *and* **Youngjoong KO**[†a)], *Nonmember*

SUMMARY In this paper, we study how to automatically classify mathematical expressions written in MathML (Mathematical Markup Language). It is an essential preprocess to resolve analysis problems originated from multi-meaning mathematical symbols. We first define twelve equation classes based on chapter information of mathematics textbooks and then conduct various experiments. Experimental results show an accuracy of 94.75%, by employing the feature combination of tags, operators, strings, and "identifier & operator" bigram.

key words: *mathematical expressions, MathML, classification*

1. Introduction

Although diverse and rich knowledge is available from Web documents, complicated math equations are often not available to be analyzed. Most Web documents containing math materials have utilized images and this fact has led to accessibility problems in reading of documents containing math expressions for visually impaired persons [1]. MathML (Mathematical Markup Language) is one of the initiatives that are created to promote the accessible publication of math contents over the Web. Since employing MathML allows Web documents to easily describe math expressions without using images, visually impaired persons have the increased possibility to acquire the math contents from the Web.

Our final goal is to convert math expressions written in MathML into audio version ones. As the Text-To-Speech technology is able to create audio version of plain text data, our work can help the visually impaired persons, in particular, the blind students, directly hear math expressions from the Web by only converting equations written in MathML into plain text expressions. This will be very useful in the current educational situation where web-based educational materials are increasing. The first step of our proposed method is to convert each math symbol into an appropriate word. For this, we extract all the math symbols from Korean high school math textbooks. We then give a name to each symbol, e.g., "plus" to a symbol '+' and "equal" to a symbol '=.' However, many ambiguous cases are observed during this conversion process. For example, the vertical line '|' is commonly used in math to express absolute value, whereas it also expresses a determinant such as |A| ('A': a matrix), a vector distance such as $|\vec{v}|$, and so on. If this ambiguity problem could be effectively resolved, |A| in the Matrix chapter can be translated into "determinant A". However, if impossible, it should be translated into "vertical line, A, vertical line" by just reading its original symbols. This translation makes it much more difficult to understand the math equation.

Table 1 Examples of multi-meaning symbols.

Symbol & Meaning		Example	Chapter
→	if / then	$p \rightarrow q$	*Proposition*
	from / to	$f: X \rightarrow Y$	*Function*
|	determinant	|A|	*Matrix*
	vector distance	$\|\vec{v}\|$	*Vector*
~	not	$\sim p$	*Proposition*
	difference	$r \sim r'$	*Equation*

As a key clue to solve this problem, we develop math equation classification based on the subjects of a textbook's chapters. Table 1 lists some multi-meaning symbols and their chapter information which can play an important role to identify the multi-meaning math symbols.

In this paper, we aim to automatically classify math expressions written in MathML into a dozen chapter-based classes. We define twelve classes referring to the math textbooks, especially the series of "수학의 정석 * (The Mathematical Manual)"; 1) Set & Proposition, 2) Equation, 3) Inequality, 4) Math Function, 5) Matrix, 6) Arithmetic progression, 7) Logarithm, 8) Trigonometric function, 9) Differential calculus, 10) Integral calculus, 11) Vector, and 12) Probability. We first define five feature types, and then classify MathML expressions using Support Vector Machine (SVM). Experimental results in 5-fold cross validation show the overall accuracy of 94.75%.

The remainder of the paper is organized as follows. Section 2 briefly describes the related work. Section 3 explains various feature types and Sect. 4 presents the experimental results. Section 5 is devoted to the discussion, and finally Sect. 6 concludes.

2. Related Work

We have not found any research on automatically classifying math equations. To the best of our knowledge, there is no re-

Manuscript received November 24, 2011.
Manuscript revised April 14, 2012.

[†]The authors are with the Department of Computer Engineering, Dong-A University, 840 Hadan 2-dong, Saha-gu, Busan, 604–714, Korea.
a) E-mail: youngjoong.ko@gmail.com

DOI: 10.1587/transinf.E95.D.2560

*It is the most representative math reference in Korea.

ported study yet on this problem. Although [1] is closely related to our final goal, converting MathML expressions into audio version ones, they did not classify math equations.

There are some other MathML-based studies we referred. [2] studied to retrieve math documents including various equations. [3] created a search system enable user to search for math formula contents. [4] proposed a similarity search method for math equations that are particularly adapted to the tree structures expressed by MathML. [5] tried to find for a math formula in real-world math documents, but still offering an extensible level of math awareness.

3. Definition of Feature Types

This section describes the five feature types. MathML has about 30 tags which all begin with *m* and include a token element, e.g. ⟨mn⟩5⟨/mn⟩ - numbers; ⟨mo⟩+⟨/mo⟩ - operators; ⟨mi⟩x⟨/mi⟩ - identifiers; ⟨mrow⟩ - a row; ⟨mfrac⟩ - fractions. After investigating the characteristics of these tags, we concluded that the following five features best represent the math equations.

3.1 First Feature: Tag

The first feature is a tag itself which is a basic unit that represents a math structure. For example, a tag ⟨mo⟩ means that an operator follows the tag. Table 2 shows an example equation. Tags are surrounded by angle parentheses such as ⟨mi⟩, ⟨mo⟩, and ⟨msup⟩. And Table 3 shows some examples.

3.2 Second Feature: Operator

The second feature is an operator which is expressed with the tag ⟨mo⟩. Most operators play an important role in classifying equations. It is true that some operators such as '+' and '=' occur in almost all chapters. On the other hand, there are many operators that occur in only one chapter as follows:

∅ (empty set) → "1) Set & Proposition" class
∫ (intetral) → "10) Integral calculus" class
// (parallel, slanted) → "11) Vector" class

Table 2 Examples of MathML equation.

Equation: $A \cap A^C = \phi$
<mi>A</mi> <mo>∩</mo> <msup><mi>A</mi><mi>c</mi></msup> <mo>=</mo> <mo>∅</mo>

Table 3 Examples of tags.

Tag		Example
<mi>: identifier	A	<mi>A</mi>
<mo>: operator	=	<mo>=</mo>
<msup>: superscript	A^c	<msup><mi>A</mi> <mi>c</mi></msup>

3.3 Third Feature: Identifier

The third feature is an identifier expressed with the tag ⟨mi⟩. An identifier includes all variables, e.g., 'a,' 'A,' and some promised symbols, e.g., "sin," "log." For example, "sin(θ)" is converted into "⟨mi⟩sin⟨/mi⟩⟨mo⟩(⟨mo⟩⟨mi⟩θ⟨/mi⟩⟨mo⟩)⟨/mo⟩." Since the promised symbol could be a good feature for equation classification, we determine to employ the identifier features.

3.4 Fourth Feature: String Bigram

Mathematical expressions occasionally include plain string as shown in Table 4.

Although string-contained mathML equations do not occur frequently, they often contain crucial information such as the string "벡터 [bek-teo]: vector". Hence, we consider syllable bigrams in each word as our fourth feature. For example, the equation in Table 4 has nine syllable bigrams: "중심 [jung-sim]," "반지 [ban-ji]," "지름 [ji-reum]," "인 [in]," "원의 [won-eui]," "**벡터 [bek-teo]**," "터방 [teo-bang]," "방정 [bang-jeong]," "정식 [jeong-sik]."

3.5 Fifth Feature: "Identifier & Operator" Bigram

The fifth feature is an "identifier & operator" bigram (hereafter, I&O). It is represented as one of the following three forms: "id/id," "id/op," and "op/op" ("id": identifier and "op": operator). This feature can compensate for each of the operators and identifiers. For example, the operator features such as "right arrow" or "vertical line" imply that an equation would belong to the "11) Vector" class. On the other hand, the "cosine" identifier implies that the equation would be assigned to the "8) Trigonometric function" class.

Table 4 String-contained MathML expression.

Equation:

$|\vec{x} - \vec{c}| = r \Leftrightarrow$ *중심* $\vec{c}$*, 반지름 r 인 원의 벡터방정식*

Here, "중심 $\vec{c}$, 반지름 r 인 원의 벡터방정식 [jung-sim $\vec{c}$, ban-ji-reum r in won-eui bek-teo-bang-jung-sik]" means "A vector equation of a circle with a center $\vec{c}$ and a radius r."

```
<mo>&mid;</mo>
<mover><mi>x</mi><mo>&RightArrow;</mo> </mover>
<mo>-</mo>
<mover> <mi>c</mi><mo>&RightArrow;</mo> </mover>
<mo>&mid;</mo>
<mo>=</mo>
<mi>r</mi> <mo>&DoubleLeftRightArrow;</mo>
<mtext>중심</mtext><mover> <mi>c</mi>
<mo>&RightArrow;</mo> </mover>
<mo>,</mo> <mtext>반지름</mtext>
<mi>r</mi> <mtext>인 원의 벡터방정식</mtext>
```

Table 5 I&Os extracted from $\overrightarrow{OA} \cdot \overrightarrow{OB} = |\overrightarrow{OA}| \cdot |\overrightarrow{OB}| \cos\theta$.

Form	Extracted I&Os
id/id	"OA,OB," "OB,OA," "OB,cos," "cos,θ"
id/op	"→,OA," "OA,·," "→,OB," "OB,=," ",OA,|," "OB,|," "|,cos"
op/op	"·,→," "=,|," "|,→," "|,·," "·,|"

Many similar cases are found in our corpus. After various experiments, we observe that these ambiguity problems can be considerably reduced via I&Os. In the case of the above equation, the "vertical line & cosine" bigram feature plays an important role in classifying that. Table 5 lists I&Os extracted from an example equation.

As will be described in the experiment section, I&Os are helpful for improving overall performance[†].

4. Experiments

We first defined twelve classes and then selected a total of 400 equations (30 ∼ 40 equations per each class) from the Korean high school math textbooks according to their importance. We manually converted these equations into MathML expressions, and then extracted features. Table 6 shows the number of distinct features per class in the collected equations.

As a learning technique, we employed SVM using the linear kernel. We used the TF/IDF scheme for feature weighting and used "accuracy" for an evaluation measure.

Table 7 shows the experimental results. We investigated all the possible combination cases of feature types. We first experimented using each feature type separately. Operators showed the highest performance, an accuracy of 76.75%. The other four feature types showed the performance of 47.25 ∼ 69.75%.

When we combined two of five feature types, top three cases showed same performance of 80.50%. Interestingly, these three cases were composed of two types among three feature types: tags, operators or I&Os. The other seven cases showed relatively low performances. At this stage, we carefully predicted that tags + operators + I&Os could perform well in the experiments for three types combinations by the experimental results of two types combinations. Actually, the combination, tags + operators + I&Os, showed the highest performance of 93.25%

Next, we attempted to combine all the four types. Among five cases, tags + operators + strings + I&Os showed the best performance of 94.75%. However, the experiment using all the five types achieved the lower performance reduced by 2.5%. We finally conclude that tags + operators + strings + I&Os is the most effective combination for math equation classification.

Table 6 Number of distinct features per class.

Class \ Feature	Tag	Op	Id	Str	I&O
1) Set & Proposition	9	16	18	62	114
2) Equation	9	15	16	6	124
3) Inequality	8	18	18	126	159
4) Math Function	10	29	39	61	252
5) Matrix	9	17	25	26	140
6) Arithmetic progression	8	24	25	66	216
7) Logarithm	10	23	15	119	152
8) Trigonometric function	11	25	20	50	183
9) Differential calculus	7	22	16	72	129
10) Integral calculus	9	26	24	219	181
11) Vector	11	16	56	88	62
12) Probability	16	34	24	110	252

Table 7 Results of top 3 combinations according to each number of combined feature types; performances with more than 93% are in bold.

# of feature types / top 3 rank		Feature Combination	Acc (%)
One	1.	Operators	76.75
	2.	I&Os	69.75
	3.	Identifiers	64.75
Two	1.	Tags + operators	80.50
	2.	Tags + I&Os	80.50
	3.	Operators + I&Os	80.50
Three	1.	**Tags + operators + I&Os**	**93.25**
	2.	Operators + identifiers + I&Os	92.00
	3.	Tags + operators + identifiers	91.25
Four	1.	**Tags + operators + strings + I&Os**	**94.75**
	2.	**Tags + operators + identifiers + I&Os**	**94.00**
	3.	Tags + operators + identifiers + strings	92.25
All		Tags + operators +identifiers + strings + I&Os	92.50

5. Discussion: Error Analysis

This section briefly introduces an error analysis. After investigating misclassified equations, we could find that some features are seriously biased. For example, Eq. (1) has the class label of "9) Differential calculus", while it was classified into "4) Math Function" by the proposed method. We think that an I&O feature of "f,(" is the cause of this error.

$$y'_{x=1} = lim_{\Delta\to 0}\frac{f(1+\Delta x) - f(1)}{\Delta x} \quad (1)$$

[†]We conducted several experiments with various sub-combinations of I&Os. We could achieve the best performance when all of the three forms ("id/id," "id/op," and "op/op") were employed simultaneously.

We also found that some equations have ambiguity problems. For example, Eq. (2) is possible to have two classes, "9) Differential calculus" and "10) Integral calculus."

$$\frac{d}{dx}\int_1^x (t^3 - t^2)dt = \left(\frac{1}{4}x^4 - \frac{1}{3}x^3 + \frac{1}{12}\right)' = x^3 - x^2 \quad (2)$$

Therefore, we think that it could be necessary for the proposed method to assign multiple classes for one equation. In addition, a kind of feature selection technique could be applied to solve the case of Eq. (1). That is, the classification strength of features can be measured by the feature selection technique and it can be utilized for math classification. In Eq. (1), "f,(" occurs twice but "lim" should be considered as a more important feature. However, simple the TF/IDF scheme cannot reflect this fact. These are regarded as the future work.

6. Conclusion

In this paper, we study the problem of classifying math equations. As far as we know, this is the first study on the problem. Our experimental results showed that the proposed method can be effectively used for math equation classification.

We plan to conduct more experiments for obtaining better performance. We also consider a method which gives multiple classes for one equation as mentioned in the discussion section. In addition, we have been collecting a considerable number of equations for a larger dataset. We will continue to study for our final goal, converting MathML equations into audio version ones.

Acknowledgement

This work was supported by the Dong-A University research fund.

References

[1] H. Ferreira and D. Freitas, "Audio-math: Towards automatic readings of mathematical expressions," Proc. Human Computer Interaction International, 2005.

[2] J. Shin and H. Kim, "An equation retrieval system based on weighted sum of heterogenous indexing terms," The Korean Institute of Information Scientists and Engineers, vol.37, no.10, pp.723–801, 2010.

[3] M. Adeel, H.S. Cheung, and S.H. Khiyal, "Math GO! Prototype of a content based mathematical formula search engine," J. Theoretical and Applied Information Technology, vol.4, no.10, pp.1002–1012, 2008.

[4] K. Yokoi and A. Aizawa, "An approach to similarity search for mathematical expressions using MathML," Towards a Digital Mathematics Library, pp.27–35, 2009.

[5] J. Misutka and L. Galambos, "Extending full text search engine for mathematical content," Towards a Digital Mathematics Library, pp.55–67, 2008.