Classifying Mathematical Expressions Written in MathML

Shinil Kim (non-member), Seon Yang (Overseas student membership), Youngjoong Ko (non-member)

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 the math expressions. For example, “plus” to a symbol “+” and “equation” to “=.” 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.

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.

| 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              | | \neg p | | Proposition |
| difference       | | r \sim r' | | Equation |

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 1 (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 Section 4 presents the experimental results. Section 5 is devoted to the discussion, and finally Section 6 concludes.

---

1 It is the most representative math reference in Korea.
2. Related work

We have not found any research on automatically classifying math equations. To the best of our knowledge, there is no reported study yet on this problem. Although [1] is closely related to our final goal, converting MathML expressions into audio versions, 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 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 conclude 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.

| Tag     | Example |
|---------|---------|
| <mi>    | identifier A |
| <mo>    | operator = |
| <mrow>  |           |
| <msup>  | superscript A^{c} |

Although string-contained MathML equations do not occur frequently, they often contain crucial information such as the string “벡터[hek-teo]: vector”. Hence, we consider syllable bigrams in each word as our fourth feature. For example, the equation in Table 4 has nine

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
∫ (integral) → “10) Integral calculus” class
∥ (parallel, slanted) → “11) Vector” class

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.

| Table 4 | String-contained MathML expression |
|---------|-----------------------------------|
| Equation: | $X - \vec{c} \parallel R \iff \text{중심 } \vec{c} \text{, 반지름 } r \text{ 인 원의 벡터방정식}$ |
| Here, “중심 $\vec{c}$, 반지름 $r$ 인 원의 벡터방정식 [jung-sim $\vec{c}$, ban-ji-reum $r$ in won-eui bek-teo-bang-sang-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>&RightArrow;</mo> | |
| <mi>&DoubleLeftRightArrow;</mi> | |
| <mtext>중심</mtext> | |
| <mover><mi>c</mi></mover> | |
| <mtext>반지름</mtext> | |
| <mover><mi>A</mi></mover> | |
| <mtext>원의 벡터방정식</mtext> | |

4. Conclusions

We have proposed a similarity search method for math equations that are particularly adapted to the tree structures expressed by MathML. For the first feature, we use the tags that follow the equation; for the second feature, we use the operators; for the third feature, we use the identifiers; for the fourth feature, we use the string bigram. Although we have not done any performance measurement yet, we have implemented an algorithm for this purpose. In our future work, we will apply this algorithm to a database of math equations to evaluate the performance.
syllable bigrams: “중심[jung-sim],” “반지[ban-ji],” “지름[ji-reum],” “원[on],” “원의[won-uei],” “벡터[beke-teo],” “테두리[teo-bang],” “방정[jang-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. 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.

| Form | Extracted I&Os |
|------|----------------|
| id/id | “OA,OB,” “OB,OA,” “OB,cos,” “cos,θ” |
| id/op | “→OA,” “→OA,” “→OB,” “OB,=,” “OA,” |
| op/op | “,” “→,” “=,” “→,” “,” “,” |

As will be described in the experiment section, I&Os are helpful for improving overall performance².

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.

| Feature Combination | Acc (%) |
|---------------------|---------|
| Operators           | 76.75   |
| I&Os                | 69.75   |
| Identifiers         | 64.75   |

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 the case of all features. We first experimented using each feature separately. Operators showed the highest performance, an accuracy of 76.75%. The other four features showed the performance of 47.25 ~ 69.75%.

² We experimented 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”) are employed simultaneously.
When we combined two of five features, top three cases showed same performance of 80.50%. Interestingly, the three cases were composed of two of three features: tags, operators or I&Os. The other seven cases showed relatively low performance. At this stage, we carefully predicted that tags + operators + I&Os would perform well. In the experiments combining three features, the combination, tags + operators + I&Os, showed the highest performance of 93.25%.

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

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 actually belongs to “9) Differential calculus”, while it was classified into “4) Math Function.” We think that an I&O of “f,(” is the cause of this error.

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

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.”

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

Therefore, we think that it could be necessary to give 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 is 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 future work.

6. Conclusion

In this paper, we study the problem of classifying math equations. 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, since this paper used small data of total 400 equations, we have been collecting a considerable number of equations for a sufficiently large corpus. We will continue to study for our final goal, converting MathML equations into audio version ones.

References

[1] Helder Ferreira and Diamantino Freitas. 2005. Audio-Math: Towards Automatic Readings of Mathematical Expressions. In Proceedings of Human Computer Interaction International.

[2] Junsoo Shin and Harksoo Kim. 2010. An Equation Retrieval System Based on Weighted Sum of Heterogenous Indexing Terms. The Korean Institute of Information Scientists and Engineers, 37(10):723-801.

[3] Muhammad Adeel, Hui Siu Cheung and Sikandar Hayat Khiyal. 2008. Math GO! Prototype of A Content Based Mathematical Formula Search Engine. Journal of Theoretical and Applied Information Technology, 4(10):1002-1012.

[4] Keisuke Yokoi and Akiko Aizawa. 2009. An Approach to Similarity Search for Mathematical Expressions using MathML. Towards a Digital Mathematics Library, 27-35.

[5] Jozef Misutka and Leo Galambos. 2008. Extending Full Text Search Engine for Mathematical Content. Towards a Digital Mathematics Library, 55-67.