Mathematical Formula Search using Natural Language Queries

Seon YANG, Youngjoong KO
Department of Computer Engineering, Dong-A University, 604-714, Republic of Korea
youngjoong.ko@gmail.com

***Abstract*—This paper presents how to search mathematical formulae written in MathML when given plain words as a query. Since the proposed method allows natural language queries like the traditional Information Retrieval for the mathematical formula search, users do not need to enter any complicated math symbols and to use any formula input tool. For this, formula data is converted into plain texts, and features are extracted from the converted texts. In our experiments, we achieve an outstanding performance, a MRR of 0.659. In addition, we introduce how to utilize formula classification for formula search. By using class information, we finally achieve an improved performance, a MRR of 0.690.**

***Index Terms*—Information retrieval, Formula search, MathML, Natural language query, Classification.**

I. Introduction

Math search is a new area of research with many enabling technologies but also many challenges [1]. Until recently, mathematical formulae had been saved as images on the Web. By this reason, many users had difficulties in directly accessing such formulae. In a recent decade, however, there has been much interest in formula search. One of the initiatives that are created to promote the accessible publication of mathematical contents is MathML (Mathematical Markup Language, http://www.w3.org/Math). Since MathML helps web documents to easily include mathematical contents, the number of documents containing MathML expressions is rapidly increasing.

In this paper, we propose how to search MathML formulae when given plain words as a query. We focus on natural language queries like those of the conventional Information Retrieval (IR). Of course, it is reasonable to use the queries written in MathML for searching for the data written in MathML. However, most people do not know about MathML. If only those who know MathML well are allowed to retrieve MathML formulae, most people cannot even have the opportunity of formula search. Although there are some mathematical formula input tools available, most of the users are not familiar with the tools.

Therefore, the main contribution of this paper is that the proposed method allows natural language queries. An example query is as follows. We assume that a user is searching for a certain formula. He or she only remembers some part of the formula.

- Target formula: $x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$
- Part that the user remembers: $\sqrt{b^2 - 4ac}$
- Example query: "*root b squared minus 4ac*"

As shown above, there is no grammar, no tool, and no math-only symbol such as root symbol; users can enter a query as if they speak the formula.

There are two opposite approaches for our purpose: 1) converting natural language queries into MathML expressions, and 2) converting MathML data into plain texts. The former is relatively simple but it is seriously dependent on the query translation performance. The latter requires more work but can alleviate the requirement of high query conversion performance. The latter has one more advantage. Since the data consist of plain texts, it can be easily combined to many traditional IR techniques. It is the reason why this paper focuses on the latter.

We first convert MathML data into plain sentences that we call 'math-sentence' hereafter.

- Definition (math-sentence): A *math-sentence* consists of plain words. It expresses a mathematical formula. For example, "*sigma i from 1 to n i cubed*" is a math-sentence that indicates $\sum_{i=1}^{n} i^3$.

Through this conversion, a wide range of mathematics vocabulary is constructed. In other words, we build up important math lexicons, e.g., a math operator lexicon, synonyms and polysemy.

Next, features are extracted from the math-sentences. Some factors such as identifiers, operators and their order are employed as features to effectively reflect the characteristics of the math contents. These features are used for indexing and ranking. As an additional experiment, we classify formulae into several classes for performance improvement and search space reduction. Finally, we achieved a MRR of 0.690.

The remainder of the paper is organized as follows: Section 2 briefly introduces the related work. Section 3 describes the proposed features and scoring scheme in detail. Section 4 presents experimental results and Section 5 discusses additional experiments. Section 6 describes the error analysis. Finally, Section 7 summarizes and concludes this paper.

II. Related Work

The main subject of our study is math search. The research of Altamimi and Youssef [1] is directly related to our study. They studied to recognize mathematical symbols and structures. Their challenge is the creation and implementation of a math query language that enables the

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

Digital Object Identifier 10.4316/AECE.2014.04015

general users to express their information needs intuitively yet precisely. They presented such a language and detail its features. Their math query language offers an alternative way to describe mathematical expressions that is more consistent and less ambiguous than conventional mathematical notation.

Youssef already discussed the roles of math search in [2]. His research addressed that the key question is what roles the Math-aware fine-grain search can play in mathematics. This research presented the short-term goals and state of the art of math-aware fine-grain search. Afterwards, it focused on how math search can help advance and manage mathematical knowledge, and discussed what needs to be done to fulfill those roles. He also presented query-relevant hit-summary generation methods [3].

Miner and Munavalli aimed at a search system that can be effectively and economically deployed [4]. They produced good results with a large portion of the mathematical content freely available on the Web. Their basic concept was to linearize mathematical notation as a sequence of text tokens, which are then indexed by a traditional text search engine. Their approach is to query for a weighted collection of significant sub-expressions, where weights depend on expression complexity, nesting depth, expression length, and special boosting of well-known expressions.

Miller and Youssef proposed augmenting presentation MathML for math search [5]. Adeel *et al*. presented the "Math GO!" system to search and present the mathematical information encoded in mathematical expressions [6]. Their approach used the concept of template based math block identification, vector representation, searching from mathematical topic based clusters and relevance ranking

Misutka and Galambos studied how to search for mathematical formulae in real-world mathematical documents, but still offering an extensible level of mathematical awareness [7]. They exploited the advantages of full text search engine and stored each formula not only once but in several generalized representations. Because it was designed as an extension, any full text search engine could adopt it. Zhao *et al*. reported on the user requirements study and preliminary implementation phrases in creating a digital library that indexes and retrieves educational materials on math [8]. They first reviewed the current approaches and resources for math retrieval, and then reported on the interviews of a small group of potential users to properly ascertain their needs. Yokoi and Aizawa proposed a similarity search method for mathematical equations that are particularly adapted to the tree structures expressed by MathML [9].

Ion explored the mathematics in the Web [10]. Kamali and Tompa proposed the mathematics retrieval [11-14], and Kamali *et al*. proposed an approach for recognizing and classifying math queries using large scale search logs [15]. Zanibbi and Blostein surveyed the recognition and the retrieval of mathematical Expressions [16]. Nghiem *et al*. explored the problem of semantic enrichment of mathematical expressions [17]. Do and Pauwels proposed an ontology alignment by introducing MathML [18].

Kohlhase *et al*. introduced MathWebSearch version 0.5 [19], and Kohlhase and Rabe presented two denotational semantics for OpenMath [20]. Lange *et al*. reimplemented the MSC (Mathematics Subject Classification) [21]. Their focus was concentrated on turning it into the new MSC authority. Sojka and Liska discussed the mathematics retrieval [22-23].

The research of Ferreira and Freitas [24] is also closely related to our study. Their goal is to convert MathML expressions into representations of audio version in English and Portuguese. They reviewed the problem of speaking mathematics and presented the tool AudioMath.

Related to the IR with class information, Liu and Croft showed that cluster-based retrieval can perform consistently across collections of realistic size [25] and Jain and Wadekar studied classification-based IR methods [26].

For our formula classification, we implemented the system proposed by Kim *et al*. [27]. The authors classified math equations into twelve classes. They conduct experiments using five types of features, tags, operators, identifiers, string bigrams, and "identifier & operator" bigrams. In Section 5, we will describe their approach in detail.

III. Formula Search with Natural Language Queries

A. Converting Formulae into Math-sentences

We convert each math formula in our corpus into a corresponding math-sentence. We observe a variety of math representations and analyze the rules of speaking formulae. Besides, we mainly refer to MathPlayer (http://www.dessci.com/en/products/mathplayer). Finally, we could implement a conversion system, which we call *MConv-sys*. While doing this, we could build some important lexicons. Among them, we briefly introduce the identifier lexicon and the operator lexicon as follows:

- Identifier lexicon: An identifier is expressed with the tag <mi> in MathML. It includes all variables, e.g., "a," "b," "x," "y," and some promised symbols, e.g., "cos" and "log."

- Operator lexicon: An operator is expressed with the tag <mo> in MathML.

Table 1 and Table 2 list identifier examples and operator examples, respectively.

TABLE 1. EXAMPLE IDENTIFIERS

Identifier	in MathML	MConv-sys
a	<mi>a</mi>	*a*
α	<mi>α</mi>	*alpha*
Ψ	<mi>Ψ</mi>	*psi*
cosθ	<mi>cos</mi><mi>θ</mi>	*cosine theta*

TABLE 2. EXAMPLE OPERATORS

Identifier	in MathML	MConv-sys
∫	<mo>∫</mo>	*integral*
∩	<mo>∩</mo>	*intersection*
ϕ	<mo>∅</mo>	*empty set*
Σ	<mo>∑</mo>	*sigma*

As a result, we could construct a corpus of math-sentences using MConv-sys. An example math-sentence is as follow:

• Example formula:

$$\int e^{\frac{i}{2}\int_0^t (p(s)dq(s)-q(s)dp(s))-i\int_0^t h(p(s),q(s))ds} dW_v(p,q)$$

• MathML expression:

```
<math xmlns="http://www.w3. ...">
 <semantics>
  <mstyle displaystyle='true'>
  <mrow>
   <mo>&#x222B;</mo>
   <mrow>
   <msup>
    <mi>e</mi><mrow>
    <mfrac>
        <mi>i</mi><mn>2</mn>
    </mfrac>
    <mstyle displaystyle='true'>
     <mrow>
     <msubsup>
         <mo>&#x222B;</mo> <mn>0</mn>
         <mi>t</mi></msubsup>
     <mrow><mo stretchy='false'>(</mo>
         <mi>p</mi>
     <mo stretchy='false'>(</mo>
         <mi>s</mi>
          ...
```

• Math-sentence (generated by MConv-sys):
"*integral e superscript i over 2 integral from 0 to t open parenthesis p open parenthesis s ...*"

B. Features and Scoring Scheme

We define two types of features, I&N and O&S, for indexing and calculating ranking scores as follows:

- **I&N (Identifiers & Numbers):** We first extract identifiers and numbers from each math-sentence. According to our observations, the order of these two elements in formulae is relatively consistent. By this reason, we employ identifiers and numbers as important features. In addition, the pattern of I&Ns is used to select candidate formulae.
- **O&S (Operators & Structures):** The other words except identifiers and numbers are also used importantly. With the exception of a few stop words, only operators and some other tokens which represent the structure of the formula are left. We define them as O&S.

TABLE 3. FEEATURES FROM AN EXAMPLE FORMULA

Math-sentence ($D = b^2 - 4ac$)	"*D equal b squared minus 4ac*"
Identifiers	D, b, a, c
Numbers	2, 4
I&Ns (with their relative positions)	**D_0, b_1, 2_2, 4_3, a_4, c_5 (count:6)**
I&N pattern	i-i-n-n-i-i
Stop word	"of"
Operators	"equal" in front of b_1, "minus" in front of 4_3
Structures	"squared" in front of 4_3
O&Ss (with their relative positions)	**"$equal_1$," "$squared_3$," "$minus_3$" (count:3)**

Indexing and ranking are both performed by using these features. We first extract all math-sentences of which I&N patterns contain the query's I&N pattern, and regard the extracted math-sentences as relevant candidates. We then calculate the score of each candidate as given in Formula (1), (2), and (3).

$$Score_{I\&N} = \frac{I\ \&\ N\ count(matched)}{I\ \&\ N\ count(indexed\ data)} \quad (1)$$

$$Score_{O\&S} = \frac{O\ \&\ S\ count(matched)}{O\ \&\ S\ count(indexed\ data)} \quad (2)$$

$$Final\ Score = Score_{I\&N} + Score_{O\&S} \quad (3)$$

Suppose that the formula "$D = b^2 - 4ac$" (given in Table Table 3) is an indexed data, and a user enter a query that indicates "$a^2 + 4bc$" as shown in Table 4. The I&N pattern of the indexed data (i-i-n-n-i-i) contains that of the query (i-n-n-i-i), so the formula is extracted as a candidate. After all the candidates are extracted, we rank the candidates. Since three I&N tokens (2, 4, c) and one O&S token "squared" are exactly matched in both value and relative position, the number of matched element of I&N is 3 and that of matched element of O&S is 1.

TABLE 4. EXAMPLE OF RANKING SCORE CALCULATION

QUERY	User's input ($a^2 + 4bc$)	"*a squared plus 4bc*
	I&N	a_0, 2_1, 4_2, b_3, c_4
	I&N pattern	i-n-n-i-i
	O&S	"$squared_2$," "$plus_2$"
DATA	I&N	D_0, b_1, 2_2, 4_3, a_4, c_5 **(count:6)**
	I&N pattern	i-i-n-n-i-i
	matched I&N	2, 4, c **(count: 3)**
	$Score_{I\&N}$	**3/6**
	O&S	"$equal_1$," "$square_3$," "$minus_3$" **(count:3)**
	matched O&S	"square" **(count: 1)**
	$Score_{O\&S}$	**1/3**
	Final Score	**3/6 + 1/3 = 5/6**

Finally, the score of the data, $Score_{IN\&OS}$, becomes 3/6 + 1/3 = 5/6 by Formula (3). In this way, we extract candidates and rank the candidates by their scores.

IV. EXPERIMENTAL RESULTS AND ANALYSIS

A. Dataset

Our corpus consists of 1,800 formulae. First, the real math formulae were extracted from math manuals that have a high school level or more. Next, they were written again in MathML by a tool of MathType (http://www.dessci.com/en/products/mathtype). At last, MConv-sys converted the MathML formulae into math-sentences.

For evaluation, 10 assessors tested our system. As an evaluation measure, we employ a well-known IR measure, MRR, which is the average of the reciprocal ranks of results for a sample of queries (Q) as shown in the following Formula (4):

$$MRR = \frac{1}{|Q|}\sum_{i=1}^{Q}\frac{1}{rank_i} \quad (4)$$

The assessors were given 200 formulae at random. (Thus, |Q| is 200 in this study.) It is assumed that those formulae are their target ones.

B. Evaluation

The math-sentences in our corpus consist of plain words, so we were able to use the traditional IR techniques. We employed TFIDF and Okapi-BM25 as the traditional weighting schemes. We regard these methods our baselines. Table 5 shows performance comparison with the baselines. Here, IN&OS indicates the method using I&Ns and O&Ss, the features introduced in Section 3.

TABLE 5. RESULTS OF THE THREE SCHEMES

Weighting scheme	MRR
TFIDF	.329
Okapi-BM25	.363
IN&OS	***.508***

As shown in Table 5, IN&OS shows the higher performance than the other two methods. We found that the performances of TFIDF and Okapi-BM25 are very dependent on operators, whereas that of IN&OS is relatively dependent on identifiers and numbers.

To compensate for these biased phenomena, we did experiments of linear combination as given in Formula (5):

$$\alpha \cdot Score_{IN\&OS} + (1-\alpha) \cdot Score_{other},\ 0 \le \alpha \le 1 \qquad (5)$$

where $Score_{IN\&OS}$ denotes the score calculated by I&Ns and O&Ss, and $Score_{Other}$ denotes the score generated by TFIDF or Okapi-BM25. Fig. 1 shows the performance comparison with eleven α values.

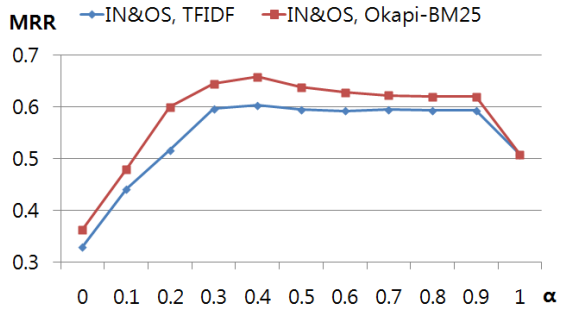

Figure 1. Performance comparison by α value

As shown in Fig. 1, the overall performance was increased by the linear combination of IN&OS and other scheme. In particular, we achieved the best performance with α of 0.4. Table 6 summarizes the results of the combined system.

TABLE 6. REEESULTS OF THE COMBINED SCHEMES

Weighting scheme	MRR
IN&OS only	.508
IN&OS, TFIDF (α = 0.4)	.604
IN&OS, Okapi-BM25 (α = 0.4)	***.659***

We achieved an outstanding performance, a MRR of 0.659, by combining IN&OS with Okapi-BM25. As we can see in the above results, the proposed IN&OS can be effectively combined with the traditional IR techniques and lead to noticeable performance improvements.

C. Other evaluation

We evaluated our method using other evaluation measure, P@n (precision at n). In this study, P@n indicates the proportion of the top-n formulae that are relevant. We first chose five short formulae (or subformulae) and generated corresponding queries as shown in Table 7.

TABLE 7. LIST OF THE FIVE SHORT TEST QUERIES

No	formula	Query
1	$x^2 + ax + 1$	"*x squared plus a x plus 1*"
2	$A \cup B \cap C = D$	"*A union B intersection C equal D*"
3	$sinx + cosx$	"*sine x plus cosine x*"
4	$\sum^{m}_{k=1} k+1$	"*sigma k from 1 to m k plus 1*"
5	$P(A\|B)$	"*P open parenthesis A bar B close parenthesis*"

Using the above five queries, we retrieved relevant formulae. We gave the top ten retrieved formulae for each query to the assessors. The formulae were labeled as "Relevant" or "Not relevant" by the assessors. For each formula, if six or more assessors answered that it is relevant, the formula was finally classified into "Relevant." For example, if only one formula is classified into "Relevant" among top ten formulae, P@10 becomes to 0.1. Table 8 shows the experimental results using P@5 and P @10.

TABLE 8. EVALUATION RESULTS WITH P@N

No	P@5	P@10
1	.40	.30
2	.60	.40
3	.60	.40
4	.60	.40
5	.80	.70
Ave	**0.60**	**0.44**

As shown in Table 8, the overall value of P@5 is higher than that of P@10. This fact means that formulae that are closer to the query were successfully ranked in a higher ranking. That is, our system could perform well even been given very short queries.

V. DISCUSSION: FORMULA SEARCH WITH CLASS INFORMATION

In this section, we classify math formulae with the method proposed by Kim *et al*. [27]. The purpose of this classification task is to improve the search performance. First, twelve classes are defined referring to the math textbooks.

- 12 classes: "*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* ," "*12) Probability*"

Next, five types of features are defined as follows:

1. **Tag:** The first feature is a tag itself. It is a basic unit that represents a mathematical structure. For example, <mo> indicates that the following token is an operator. Tags are surrounded by angle parentheses such as <mi> (identifier), <mo> (operator), <mroot> (root), and <mfrac> (fraction), etc.

2. **Operator:** As explained Section 3, an operator is expressed with the tag <mo> in MathML. Most operators play an important role in classifying formulae. It is true that some operators such as '+' and '=' occur in almost all classes. On the other hand, there are many operators that mainly occur in one class. Some examples are as follow:

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

3. **Identifier:** As explained in Section 3, an identifier is expressed with the tag <mi> in MathML.

4. **String**: A string is expressed with the tag <mtext> in MathML. We consider every word bigram as our fourth feature. Although string-contained MathML formulae do not occur frequently, they often contain crucial information.

5. **"Identifier & operator" bigram (I&O):** An I&O 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 a formula $\vec{OA}\cdot\vec{OB}=|\vec{OA}|\cdot|\vec{OB}|\cos\theta$ would belong to the "*11) Vector*" class. On the other hand, the "cosine" identifier implies that the formula 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&O features. In the case of the above formula, the "vertical line & cosine" bigram feature plays an important role in resolving that kind of ambiguity problems.

We investigated all the possible combination cases of the five feature types to find the best combination. We evaluated this classification method on 5-fold cross validation using SVM with the linear kernel and the TFIDF scheme. Table 9 lists the accuracy values for each feature combination.

TABLE 9. RESULTS OF FORMULA CLASSIFICATION

# of feature types / Rank		Feature combinations	Accuracy
One	1	Operators	.768
	2	I&Os	.703
Two	1	Tags + operators	.805
	2	Tags + I&Os	.787
Three	1	Tags + operators + I&Os	.933
	2	Operators + identifiers+ I&Os	.920
Four	**1**	**Tags + operators + strings + I&Os**	**.947***
	2	Tags + operators + identifiers + I&Os	.935
All five	-	Tags + operators +identifiers + strings + I&Os	.921

In Table 9, * indicates statistically significant improvement over other values according to t-test at $p < 0.05$ level (http://www.graphpad.com) [28].

After we classified all the formulae in our corpus, we added the class information to the queries. Thus the search space for a query is limited to its class. Finally, we could obtain improved performance by about 3%. Table 10 summaries our final results.

TABLE 10. FINAL RESULTS

Weighting scheme	MRR
IN&OS only	.508
IN&OS, Okapi-BM25	.659
IN&OS, Okapi-BM25, Class information	***.690***

As shown in Table 10, we finally achieved an outstanding performance, a MRR of 0.690, using the class information additionally. It should be noted that the use of class information can provide fast processing time to formula search because it reduces the search space.

VI. ERROR ANALYSIS

Although the proposed method showed an outstanding performance, several challenges remain. We analyzed some major sources of the remaining errors as follows.

1. **When a user only remembers separated parts in an expression:**

   Assume that the following case.

• Target formula: $x=\frac{-b\pm\sqrt{b^2-4ac}}{2a}$

• Three short parts that a user remembers: b, $\sqrt{\ }$, 2a

In this case, user may enter a query as "*b*, *root*, *2a*," However, our system does not recognize three separated parts. The system will consider the query as one sub-formula, $b\sqrt{2a}$. It is required to enhance our algorithm to prepare this case

2. **When a user remembers the wrong order of math tokens:**

Since I&N pattern is one of the major features, the order of math tokens in a query is very important. If a user enters a query in the wrong order, e.g., "*root 4ac minus b squared*" ($\sqrt{4ac-b^2}$), the system would generate "n-i-i-i-n" as I&N pattern and may bring wrong results. This case should also be considered in the future system enhancement.

3. **Length normalization:**

Most of the existing IR systems are known to perform length normalization. We also perform length normalization; the denominator values in formula (1) an (2) are almost determined by the length of a formula. In the following case,

for example, formula-1 would get the higher rank compared to formula-2.

• query: "*root b squared minus 4ac*" ($\sqrt{b^2-4ac}$)

• formula-1: $D=\sqrt{b^2-4ac}$

• formula-2: $x=\frac{-b\pm\sqrt{b^2-4ac}}{2a}$

This result is considered reasonable from the perspective of both document retrieval and formula search. However, there are cases where a user wants to find a very big formula with only a small part of the expression. Thus, one of the approaches we have been examining is to give users an option, *whether to perform length normalization or not,* so that for the users to easily find the formulae with a sub-expression that perfectly corresponds with the query. If the user chooses no length normalization, formula-1 and formula-2 will have the same ranks.

VII. Conclusion

This paper has presented how to retrieve MathML formulae using natural language queries. We first implemented MConv-sys which converts MathML formulae into plain sentences. We changed MathML data into math-sentences using MConv-sys. We then extracted features, I&Ns and O&Ss from the math-sentences, and retrieved formulae using the features. We also used conventional weighting schemes and combined them to our proposed weighting scheme. In the discussion section, we conducted the formula classification task to improve the search performance. We achieved an improved performance by about 3%. These results show that the proposed method can be effectively used for the real world math information retrieval.

There are several interesting directions for the future work. The first is to enhance our system to solve the problems described in the error analysis section. The second is to build a larger collection is an important plan, too. We already have started to collect a large volume of mathematical documents. The third is to conduct experiments by converting queries into MathML expressions not converting formula data into math-sentences. The fourth is related to improve the performance. We will combine our method (IN&OS) to the state-of-the-art IR techniques beyond TFIDF or Okapi-BM25. In addition, we will conduct experiments by applying unsupervised clustering techniques, e.g., topic models, to our system.

References

[1] M. E. Altamimi and A. Youssef, "A Math Query Language with an Expanded Set of Wildcards," *Mathematics in Computer Science*, vol. 2, no. 2, pp. 305-331, 2008.

[2] A. Youssef, "Roles of Math Search in Mathematics," in *Proc. Mathematical Knowledge Management*, pp. 2-16, 2006.

[3] A. Youssef, "Relevance Ranking and Hit Description in Math Search," *Mathematics in Computer Science*, vol. 2, no. 2, pp. 333-353, 2008.

[4] R. Miner and R. Munavalli, "An Approach to Mathematical Search Through Query Formulation and Data Normalization," in *Proc. Mathematical Knowledge Management*, pp.342-355, 2007.

[5] B. R. Miller and A. Youssef, "Augmenting Presentation MathML for Search," in *Proc. Mathematical Knowledge Management*, pp. 536-542, 2008.

[6] M. Adeel, H. S. Cheung and S. H. Khiyal, "Math GO! Prototype of A Content Based Mathematical Formula Search Engine," *Journal of Theoretical and Applied Information Technology*, vol. 4, no. 10, pp. 1002-1012, 2008.

[7] J. Misutka and L. Galambos, "Extending Full Text Search Engine for Mathematical Content," in *Proc. Towards a Digital Mathematics Library*, pp. 55-67, 2008.

[8] J. Zhao, M. Kan and Y. L. Theng, "Math Information Retrieval: User Requirements and Prototype Implementation," in *Proc. Joint Conference on Digital Libraries*, pp. 187-196, 2008.

[9] K. Yokoi and A. Aizawa, "An Approach to Similarity Search for Mathematical Expressions using MathML," in *Proc. Towards a Digital Mathematics Library*, pp. 27-35, 2009.

[10] P. D. F. Ion, "Mathematics and the World Wide Web," in *Proc. Towards a Digital Mathematics Library*, pp. 230-245, 2013.

[11] S, Kamali and F. W. Tompa, "Structural Similarity Search for Mathematics Retrieval," in *Proc. Intelligent Computer Mathematics*, pp. 246-262, 2013.

[12] S, Kamali and F. W. Tompa, "Retrieving Documents with Mathematical Content," in *Proc. Special Interest Group on Information Retrieval*, pp. 353-362, 2013.

[13] S. Kamali and F. W. Tompa, "A new mathematics retrieval system," in *Proc. Conference on Information and Knowledge Management*, pp. 1413-1416, 2010.

[14] S. Kamali and F. W. Tompa, "Improving Mathematics Retrieval," in *Proc. Towards a Digital Mathematics Library*, pp. 37-48, 2009.

[15] S, Kamali, J. Apacible and Y. Hosseinkashi, "Answering Math Queries with Search Engines," in *Proc. companion on World Wide Web*, pp. 43-52, 2012.

[16] R. Zanibbi and D. Blostein, "Recognition and Retrieval of Mathematical Expressions," *International Journal on Document Analysis and Recognition*, vol. 15, pp 331-357, 2012.

[17] M. Nghiem, G. Y. Kristianto and A. Aizawa, "Using MathML Parallel Markup Corpora for Semantic Enrichment of Mathematical Expressions," *IEICE Transactions*, vol. 96-D, no. 8, pp. 1707-1715, 2013.

[18] C. Do and E. J. Pauwels, "Using MathML to Represent Units of Measurement for Improved Ontology Alignment," in *Proc. Towards a Digital Mathematics Library*, pp. 310-325, 2013.

[19] M. Kohlhase, B. Matican and C. Prodescu, "MathWebSearch 0.5: Scaling an Open Formula Search Engine," in *Proc. Artificial Intelligence and Symbolic Computation*, pp. 342-357, 2012.

[20] M. Kohlhase and F. Rabe, "Semantics of OpenMath and MathML3," *Mathematics in Computer Science*, vol. 6, pp 235-260, 2012.

[21] C. Lange, P. Ion, A. Dimou, C. Bratsas, W. Sperber, M. Kohlhase and I. Antoniou, "Bringing Mathematics to the Web of Data: The Case of the Mathematics Subject Classification," in *Proc. European Semantic Web Symposium*, pp. 763-777, 2012.

[22] P. Sojka and M. Liska, "Indexing and Searching Mathematics in Digital Libraries - Architecture, Design and Scalability Issues," in *Proc. Mathematical Knowledge Management*, pp. 228-243, 2011.

[23] P. Sojka and M. Liska, "The Art of Mathematics Retrieval," in *Proc. ACM Symposium on Document Engineering*, pp. 57-60, 2011.

[24] H. Ferreira and D. Freitas, "Audio-Math: Towards Automatic Readings of Mathematical Expressions," in *Proc. Human Computer Interaction International*, 2005.

[25] L. Liu and W. B. Croft, "Cluster-Based Retrieval Using Language Models," In *Proc. Special Interest Group on Information Retrieval*, pp. 186-193, 2004.

[26] Y. K. Jain and S. Wadekar, "Classification-based Retrieval Methods to Enhance Information Discovery on the Web," *International Journal of Managing Information Technology*, vol. 3, no. 1, pp. 33-44, 2011

[27] S. Kim, S. Yang and Y. Ko, "Classifying Mathematical Expressions Written in MathML," *IEICE Transactions on Information and Systems*, vol. E95-D, no. 10, pp. 2560-2563, 2012.

[28] P. Refaeilzadeh, L. Tang and H. Liu, "Cross-validation," *Encyclopedia of Database Systems*. pp. 532-538, 2009.