Feature Extraction and Clustering-based Retrieval for Mathematical Formulas

Kai Ma
School of Computer Engineering
Nanyang Technological University,
Singapore 639798, Singapore
maka0001@ntu.edu.sg

Siu Cheung Hui
School of Computer Engineering
Nanyang Technological University,
Singapore 639798, Singapore
asschui@ntu.edu.sg

Kuiyu Chang
School of Computer Engineering
Nanyang Technological University,
Singapore 639798, Singapore
askychang@ntu.edu.sg

***Abstract*—Mathematical formulas or expressions are essential for presenting scientific knowledge in many research documents in academic areas such as physics and mathematics. Searching for related mathematical formulas is an important but challenging problem as formulas contain both structural and semantic information. Such information is hidden inside the mathematical expressions of the formulas. To support effective formula search, it is necessary to extract the structural and semantic features from the mathematical presentation of the formulas faithfully. In this paper, we propose an effective approach for formula feature extraction. To evaluate the proposed approach, the extracted features are tested with three popular clustering algorithms, namely K-means, Self Organizing Map (SOM), and Agglomerative Hierarchical Clustering (AHC), for formula retrieval. The performance of the clustering-based retrieval is measured based on a dataset of 881 formulas and promising results have been achieved.**

Keywords-feature extracction; formula search; clustering; information retrieval

I. INTRODUCTION

In many scientific documents, formulas are commonly used for presenting the theoretical foundation of the underlying knowledge mathematically. Searching for related formulas is an important but challenging problem as formulas contain both structural and semantic information. Such information is hidden inside the mathematical expressions of the formulas. Current popular search engines such as Google and Yahoo! work well on text-based Web information. However, they are not always effective when it comes to certain types of data such as formulas. Unlike textual data [1, 2, 3], formulas contain both semantic and structural information which cannot be extracted easily by conventional search engines. For example, in the formula e^{sinx}, it contains semantic meanings for the symbols *e* and *sin* that represent the exponential and trigonometric functions respectively. In addition, it also contains structural information as the *sin* function is structurally related to the exponential function as a power. As such, extracting semantic information alone is certainly not effective for supporting formula search.

In this paper, we propose an effective feature extraction approach which extracts both semantic and structural information as well as constant and variable information from mathematical formulas. The extracted features are evaluated using three clustering algorithms, namely K-means [4], Self Organizing Map (SOM) [5], and Agglomerative Hierarchical Clustering (AHC) [6]. The rest of the paper is organized as follows. Section 2 gives the related works on feature extraction for formula search. Section 3 presents the proposed feature extraction approach. Section 4 discusses the feature-based clustering algorithms for formula retrieval. Section 5 presents the performance of the clustering-based algorithms for formula retrieval. Finally, Section 6 concludes the paper.

II. RELATED WORK

One of the formula search engines available on the Web is developed by WolfFram Research [7] to support mathematical search. In this search engine, it mainly extracts semantic features based on formula operators and functions. In other experimental formula search systems, conventional text retrieval approaches are usually adopted. In these systems, formulas are transformed into a sequence of key terms representing the semantic meanings for each mathematical symbol. For example, Youssef [8] wrapped math symbols to textual data. Similarly, Munavalli and Miner [9] encoded math formulas into a sequence of textual data called math fragments. Although most semantic information in formulas is retained, structural information is generally missing in such approaches. As formulas are highly symbolic and structured, these unique characteristics have made common text processing techniques inefficient.

In [10], Samarasinghe and Hui proposed a search engine for math documents. It extracts both formula and textual features from math documents for retrieval. The approach is based on the clustering of both formula and text features that are obtained by preprocessing the math documents. For formula extraction, regular expressions are defined in order to extract the semantic and structural information. However, it is infeasible to define regular expressions for all the commonly used mathematical formulas. In addition, the regular expressions which are currently defined manually are also quite inefficient. As such, the approach is unable to

extract both semantic and structural information from formulas effectively. In the next section, we discuss our proposed feature extraction approach which aims to extract both semantic and structural information as well as number constant and variable information effectively.

III. FORMULA FEATURE EXTRACTION

A. MathML

For representing formulas, one of the most common approaches is to use standardized representation format such as LaTeX, ASCIIMath [11] and Mathematical Markup Language (MathML) [12]. LaTeX is an extensively used markup language in the scientific community for document authoring. Formulas are represented as a sequence of text characters in this format. For example, the formula given in Figure 1 is represented by $[\int5e^{2x-1}\,dx]$. This representation merely focuses on addressing the formatting issues of a formula and produces a visually appealing formula. The drawback of this format is that both formula semantic and structural meanings are difficult to extract. Similar to LaTeX, ASCIIMath also formats formulas as text characters and focuses on producing nice-looking math formulas on Web pages. The formula shown in Figure 1 is formatted as `int5e^(2x-1)dx` in ASCIIMath, which is much simpler than that of LaTeX. Moreover, the ASCIIMath format is rather easy to store.

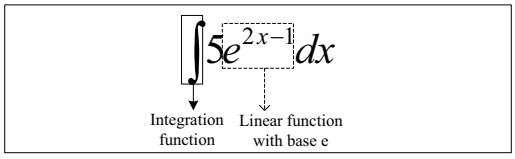

Figure 1.The semantic and structural information in a formula.

MathML is a standard markup language released as a recommendation by W3C. It is an XML application for describing mathematical notation. It captures both its structure and content. With MathML, math expressions are not only able to be represented semantically, but also can be displayed easily on the browser. All MathML elements fall into one of the three categories: presentation elements, content elements and interface elements. The MathML presentation markup represents all mathematical notations as visual terms with semantic meaning, and more importantly, all of which are structurally organized.

The layout schemata of MathML are grouped into several classes. One group of elements is concerned with scripts, which contains elements such as *msub*, *munder*, and *mmultiscripts*. Another group focuses on more general layout, which includes *mrow*, *mstyle*, and *mfrac*. A third group deals with tables. The *maction* element is in a category by itself, and allows coding of various kinds of actions on notation. Table I shows an example formula $\int 5e^{(2x-1)}dx$ encoded in MathML. In the figure, the *mo*, *mn*, *mi* and *msup* tags stand for the operators ∫ and -, the number constants *5, 2* and *1*, the identifiers *x* and *dx*, and the power operator respectively. In addition, the tag *mrow* is used to control the displaying format. In this research, we use ASCIIMath format to store the formulas, while MathML format is used for feature processing.

TABLE I. MATHML OF THE FORMULA $\int 5e^{(2x-1)}dx$.

MathML	Meaning
<math>	root (all starts with <math>)
<mo> ∫ </mo>	∫ operator (mo)
<mn> 5 </mn>	number constant 5 (mn)
<msup>	power (superscript)
<mi> e </mi>	identifier e (mi)
<mrow>	format tag
<mn> 2 </mn>	number constant 2 (mn)
<mi> x </mi>	identifier x (mi)
<mo> - </mo>	operator - (mo)
<mn> 1 </mn>	number constant 1 (mn)
</mrow>	
</msup>	
<mrow>	format tag
<mi> dx </mi>	identifier dx (mi)
</mrow>	
</math>	

B. Formula Information

As we have mentioned in Section 1, formulas are highly symbolic and structured. That is, formula elements contain not only semantic meanings, but also structural meanings. For example, in Figure 1, the symbols ∫ and *e* have their semantic meanings as *integration function* and *exponential function* respectively. And the sub-formula *(2x-1)* is structurally related to *e*, seen as superscript. In fact, this relationship is known as the *base*. Due to this reason, formulas can also be considered as two-dimensional information objects [13]. In general, there are two types of information contained in a formula, which are:

- *Semantic Information*. It refers to the semantic meaning in a formula and this information focuses more on describing each element independently.
- *Structural Information.* It refers to the structural meaning in the formula, that is, the structural relationships among those formula elements.

Table II gives some examples for both types of information. In fact, these two types of information are very important in determining the meaning of a formula. As shown in Table II, all elements in mathematical expressions have semantic information, for example, ∫ means the *integration function*, and *sec* means the *secant function*. While for those annexed functions, especially for number constants and variables, they are often related to other elements. So this information can be better described by structural information. For example, the variable *x* and number constant *5* in *sec5x* are related to the *sec* function, their structural information can be described as "*x* and 5 are in the *secant function*".

C. Proposed Approach

Conventional retrieval systems use terms extracted from documents as features to index document collection. However, this approach cannot be applied directly to

TABLE II. EXAMPLES FOR SEMANTIC & STRUCTRUAL INFORMATION.

Mathematical Expression	Semantic Information	Structural Information
$\int \sec x dx$	integration, secant function, variable x	secant function in integration, x in secant function
$\sec 5x$	secant function, variable x, number constant 5	x in secant function 5 in secant function
$e^{\sec x}$	exponential function, secant function, variable x	secant function base on exp., x in secant function

formulas. For example, the formula $\int 5e^{(2x-1)}dx$ can be tokenized as ∫, *5*, *e*, (, *2*, *x*, -, *1*,), *d* and *x*. These tokens cannot be used to index the formula because the semantic and structural information of the formula is not fully preserved. To enable further processing on formulas, such as formula search, we have to find an effective approach to extract both types of information from formulas.

To tackle this problem, we propose an approach for extracting semantic and structural information from formulas. The proposed approach consists of five major steps which are shown in Figure 2: MathML Conversion, DOM Tree Construction, Semantic Feature Extraction, Structural Feature Extraction, and Number Constant and Variable Feature Extraction.

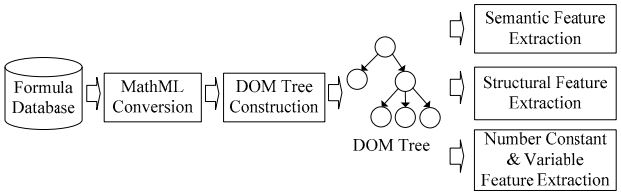

Figure 2. Proposed feature extraction approach.

MathML Conversion

In this research, both textual data and formulas of mathematical documents are stored in text-based format. We use the ASCIIMath format to represent formulas. However, one of the major drawbacks is that it is not easy to obtain the semantic meaning of formulas from such format, since all formula elements are represented as a sequence of characters. To extract semantic features, we convert the formulas from the ASCIIMath format into MathML using a converter. The conversion process is illustrated in Figure 3.

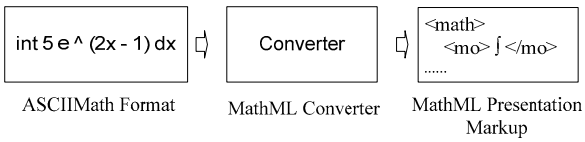

Figure 3. MathML conversion process.

DOM Tree Construction

Although MathML is organized as a well-formed structure, it is still in form of a sequence of text items which cannot be processed easily. For solving this problem, the Document Object Model (DOM) tree of the given MathML is constructed. Algorithm 1 illustrates the process of constructing a DOM tree.

We take a part of the MathML given in Table 1 as an example to illustrate the DOM tree construction process which is shown in Figure 4. Firstly, we create the root of the DOM tree by the first tag <*math*> in Figure 4(a) and set it as Current Parent (CP) (in step 1 of Figure 4(b)). As the next tag <*mo*> represents the starting tag of a new node, correspondingly we create a new tree node by this tag as *mo*, then add it as a child to the root *math*, and set this newly created node as CP. As the following tag </*mo*> indicates it is an end tag, its content ∫ is then added as the *inner text* to the CP *mo* as "*mo [∫]*", and set CP's parent as CP. This process stops until it reaches the end tag of the root </*math*>.

Algorithm 1 *DOM_Tree_Construction*

Input:
MathML - A formula in *MathML* format
Output:
DOM tree - A tree structure representing a formula
Process:
1: Create the *root* of the *DOM tree* by the first tag of *MathML* and set it as *Current Parent* (CP).
2: Starting from the next tag, check whether it is the end tag or not. If it is not an end tag, create a new node by the name of this MathML tag and add it as a *child* to CP and set it as CP, otherwise add its *content* as *inner text* of CP and set CP's *parent* as CP.
3: Recursively create and add nodes to the *DOM tree* following steps 1 and 2 until the end tag of the *root* is reached.
4: Return *DOM tree*.

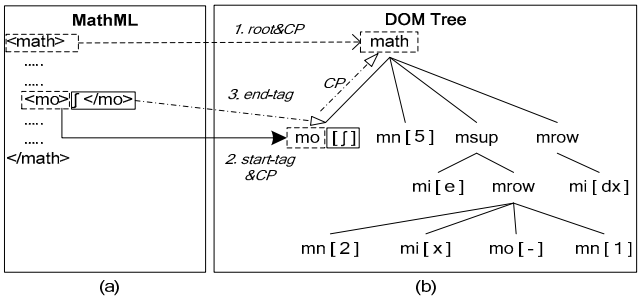

Figure 4. DOM tree construction.

Semantic Feature Extraction

This step aims to extract the semantic features from a DOM tree. It only focuses on extracting semantic features from formula operators and functions, which determine the essential underlying meanings of a formula. The extraction of number constant and variable feature extraction will also be conducted, which will be discussed later. Given a DOM tree, we traverse the entire tree in preorder. Algorithm 2 illustrates the process of semantic feature extraction.

For example, to extract the semantic features from the DOM tree shown in Figure 4(b), according to Algorithm 2, we begin with the node *mo [∫]*. As it satisfies the condition in step 2 of Algorithm 2, the symbol ∫ is extracted as semantic feature representing the integration function (see Figure 5(a)). Likewise, for the node *mi [e]*, after checking its

Algorithm 2 *Semantic Feature Extraction*

Input:
DOM Tree - A *tree* structure representing a formula
Output:
{*Semantic_feature*} - A set of semantic features of a formula
Process:
1: Traverse all tree nodes in *DOM Tree* in preorder except the *root*.
2: If the name of the node equals to *mo*, add its *inner text* to {*Semantic_ feature*}.
3: Else if the name of the node equals to *mi*, although it means *identifier*, we still have to check whether there are any *functions* in it or not. If it has, extract it from the *inner text* of this node and add it to {*Semantic_ feature*}.
4: Else check whether there is a node named *mi* among its children (operand carried operator is meaningless, e.g. *sin2*, *ln3*). If there is, add the name of this node to {*Semantic_ feature*} except for format node named *mrow*.
5: Return {*Semantic_ feature*}.

inner text, as the symbol *e* represents the exponential function, we extract it as semantic feature. To extract semantic feature for the node *msup*, according to the condition stated in step 4 of Algorithm 2, we found one of its children *mi [x]* is such a node containing a variable. Thus, the *msup* is extracted as semantic feature (see Figure 5(b)). As a result, the resultant semantic features for the DOM tree given in Figure 4(b) are {∫, *msup* , *e*, -}.

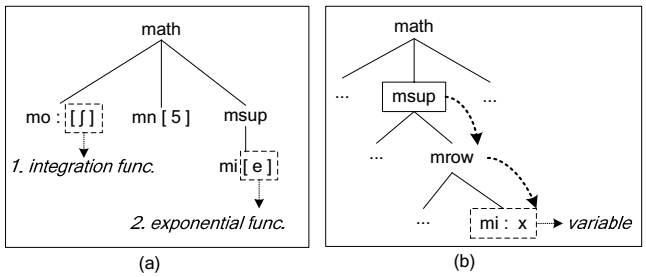

Figure 5. Semantic feature extraction.

Structural Feature Extraction

This step aims to extract the structural features from a given DOM tree. Structural information may reflect the actual meaning of a formula element. For example, consider two formulas $f^2(x)$, $ff(x)$. Although both formulas contain the same function *f*, the meaning is quite different from each other. For *f* in $f^2(x)$, it is under *square*, which means it belongs to the function *square*. While for $ff(x)$, both functions annex together, which means the inner *f* belongs to the outside function. In fact, this process, which is shown in Algorithm 3, is similar to the semantic feature extraction.

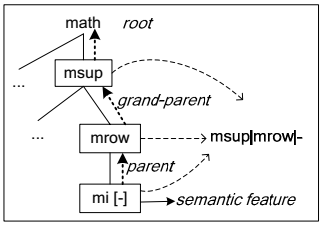

Figure 6. Structural feature extraction.

Algorithm 3 *Structural Feature Extraction*

Input:
DOMTree - A *tree* structure representing a formula
Output:
{*Structural_feature*} - A set of structural features of a formula
Process:
1: Traverse all nodes in *DOM Tree* in preorder except the *root*.
2: If the node contains *semantic feature* and its *depth* is greater than one (*depth*=1, means no relations to others), traverse its *parent*, *grand-parent*, etc. until it reaches the *root*. Then, combine the *name* of these nodes with their *semantic features* together as a feature term and add it into {*Structural_feature*}.
3: Return {*Structural_feature*}

For example, consider the extraction of the structural feature of the node *mi[-]* (its *depth* = 3 > 1) of the DOM Tree given in Figure 4(b). Firstly, we obtain its semantic feature "-". Then, traverse its parent *mrow* and grand-parent *msup* until the root *math* is reached. We then combine *mrow* and *msup* (as prefix) with the semantic feature to form the structural feature *msup*|*mrow*|- ("|" is a separator). Figure 6 illustrates the above structural feature extraction process. As a result, for the DOM tree given in Figure 4(b), the two structural features extracted are {*e*|*msup*, *msup*|*mrow*|-}.

Number Constant and Variable Feature Extraction

Unlike *operators* or *functions*, both number constants and variables are not representative, so their semantic features may not be meaningful. In this case, it is better to describe such information with structural features. However, it is still different from the structural feature extraction process for *operators* or *functions*. Therefore, we separate it as an independent feature extraction process. The number constant and variable feature extraction algorithm is based on the structural feature extraction algorithm given in Algorithm 3. The only difference is that the values of the variable and number constant are represented as common terms like *var* and *cn* (stands for variable and number constant) rather than their exact values. It is obvious that the exact value representations are not meaningful, e.g. *12* or *x*, for formula search purpose.

For example, the number constant and variable feature extraction process for *2* and *x* in the DOM tree given in Figure 4(b) can be described as follows. Firstly, we extract their structural features as *msup*|*mrow*|*5* and *msup*|*mrow*|*x*. Then, the extracted values are replaced with *var* and *cn*. As a result, the extracted features for the number constant *5* and variable *x* are {*msup*|*mrow*|*cn*, *msup*|*mrow*|*var*}.

IV. Formula Clustering & Retrieval

In this section, we incorporate the formula features extracted using our proposed approach into three clustering algorithms, namely K-means, SOM, and AHC for clustering and retrieval. This serves the purpose for evaluating the performance of the proposed feature extraction approach under clustering-based retrieval.

Figure 7 shows the formula clustering and retrieval approach which consists of two stages: clustering and retrieval. The clustering stage comprises the following three steps:

- *Feature Extraction*. This step uses the proposed feature extraction approach presented in Section III. The semantic features, structural features, and number constant and variable features are extracted from the formulas.
- *Transformation.* This step converts all formula features obtained in the feature extraction step into feature vectors, which are then normalized in preparation for training the clusters based on different clustering algorithms. For obtaining the value in each dimension of a feature vector, we use the *tf-idf* (term frequency × inverse document frequency) weighting scheme. Since the *tf-idf* is biased to those documents which contain more terms [15], the *tf-idf* feature vectors need to be normalized before feeding them into clustering algorithms. As such, we normalize the *tf-idf* vectors using *cosine normalization*, i.e. the normalized vector $\vec{V}_{norm}$ to its original vector $\vec{V}(w_1, w_2, ..., w_M)$ is given by:

$$\vec{V}_{norm} = \frac{1}{\sqrt{w_1^2 + w_2^2 + ... + w_M^2}} \vec{V} = \frac{1}{\sqrt{w_1^2 + w_2^2 + ... + w_M^2}} (w_1, w_2, ..., w_M)$$

Figure 8 shows an example for feature extraction and transformation processes. As shown in the figure, after the formula feature extraction process, four types of features are obtained together with their occurrences in the formula $\int 5e^{2x-1}dx$. In the transformation step, if a total of *N* distinct feature terms are extracted from the whole formula collection, then each formula can be represented with a feature vector that consists of *N* dimensions. The index column in the figure represents the position of each feature term in the *N* dimensional feature vector. The *tf-idf* value for each feature term is then calculated representing the corresponding value of the keyword at the i^{th} index in the feature vector. Finally, after cosine normalization, the normalized feature vector is obtained.
- *Cluster Model Generation.* In this step, the normalized formula vectors are trained using the clustering algorithms K-means, SOM, and AHC. After training, the generated cluster information is then stored in the knowledge base for subsequent retrieval.

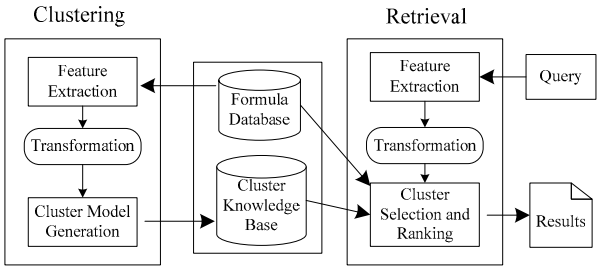

Figure 7. Formula clustering and retrieval.

While for the retrieval stage, it comprises the following three steps:

- *Query Feature Extraction*. In this step, formula features are extracted from the query formula.
- *Query Transformation.* Similar to the Transformation step in the clustering stage, the generated query vector is transformed into a normalized vector.
- *Cluster Selection and Ranking.* This step firstly selects the nearest cluster to the normalized query vector using *Euclidean distance*. Then, the ranking step is carried out to rank all formulas within this cluster by calculating the *cosine similarity* between the query vector $\vec{V}(q)$ and other formula vector $\vec{V}(f)$ using the following formula:

$$\text{sim}(q,f) = \frac{\vec{V}(q) \cdot \vec{V}(f)}{|\vec{V}(q)| \cdot |\vec{V}(f)|},$$

where the numerator represents the dot product (also known as the inner product) of the vectors $\vec{V}(q)$ and $\vec{V}(f)$, while the denominator is the product of their *Euclidean lengths*.

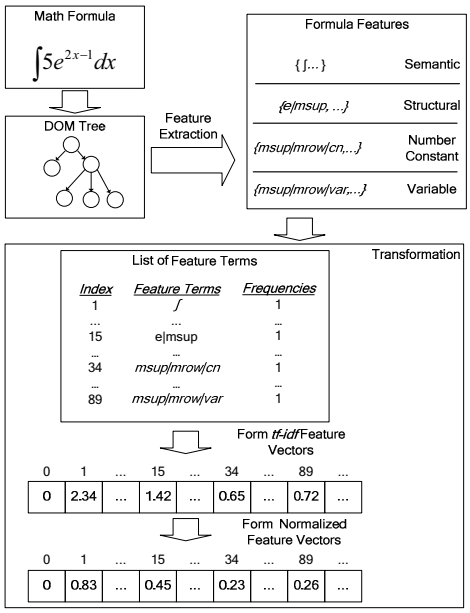

Figure 8. An example of feature extraction and transformation.

V. PERFORMANCE RESULTS

In this section, we present the performance results of the clustering-based formula retrieval based on K-means, SOM and AHC. As there is no benchmark formula dataset available for either training the clustering algorithms for the retrieval model or testing the clustering results, we have conducted the performance evaluation by collecting a dataset of 884 formulas that are extracted from various mathematical books with different types of formulas, such as functions, inequalities, logarithmic functions, exponential functions, trigonometry, differentiation, integration, etc. Table III gives some sample formula data that are used in the experiment. As this set of formulas covers a wide range of topics with different semantic and structural features, we believe that they should be effective to be used for

evaluating the performance of the proposed clustering-based formula retrieval.

The experiment is conducted as follows: 20 training samples and 20 test samples are created to test the training and retrieval accuracy for each clustering algorithm. The measurement of the retrieval accuracy is carried out by a metric called Average Precision at Seen Relevant Documents [14]. This metric favors retrieval systems that retrieve relevant formulas in higher ranking. It is a good indicator for the usability of formula matching, since users typically spend more time in examining top ranked results than those lower ones. In this experiment, we analyze both top 5 and top 10 formulas ranked in the result set to evaluate the retrieval accuracy.

TABLE III. EXAMPLE FORMULAS IN THE DATASET.

Formula	Formula Type	Total
$f(x)=\frac{1}{8}(28+30x+41x^2)$	Function	36
$x^2+2x-3<0$	Inequalities	61
$\ln(1+x)$	Logarithmic Function	18
$y=1-e^{-2x}$	Exponential Function	27
$6\sec\theta-5\tan\theta=12$	Trigonometry	81
$\int_0^1 \frac{2x-1}{\sqrt{4x^2+4x+2}}dx$	Integration	224
$\frac{dy}{dx}=2x+2xy^2$	Differentiation	21
$P(X\le 2)=0.765$	Probability	35
$\sum_{r=1}^{n} r(r+4)=\frac{1}{6n(n+1)}$	Summation	22

TABLE IV. PERFORMANCE RESULTS.

		K-means	AHC	SOM
@Top5	Training Accuracy	98.00%	98.00%	99.00%
	Testing Accuracy	90.00%	91.00%	92.00%
@Top10	Training Accuracy	96.00%	95.00%	98.50%
	Testing Accuracy	82.00%	85.00%	86.00%

Table IV shows the performance comparison of the three algorithms based on the 20 training samples and 20 test samples respectively. The three clustering algorithms have achieved very high training performance. The test training data are mostly retrieved as the first ranked result. For the 20 test samples, the average accuracy of SOM has achieved 92% and 86% respectively at top 5 and top 10 retrieval. As can be seen, SOM has performed better than the other two algorithms.

VI. CONCLUSTION

In this paper, we have proposed an approach for semantic and structural feature extraction from formulas, in which the semantic features, structural features, and number constant and variable features are extracted. Then, we apply the extracted features into clustering-based formula retrieval algorithms based on K-means, SOM, and AHC for performance evaluation. The experiment for the clustering-based formula retrieval is then conducted and promising performance results have been achieved. As for future work, we intend to conduct further experiments with a much larger dataset of testing samples. In addition, we will also develop a hybrid semantic search approach based on text and formulas for math document retrieval. As such, scientific mathematical documents can be indexed and retrieved by hybrid search using textual contents and formulas.

REFERENCES

[1] J. Misutka and L. Galambos, "Mathematical Extension of Full Text Search Engine Indexer," Proc. International Conference on Telecom Technology and Applications, pp. 1-6, 2008.

[2] B.R. Miller and A. Youssef, "Technical Aspects of the Digital Library of Mathematical Functions," in Annals of Mathematics and Artificial Intelligence, Springer Netherlands, pp. 121-136, 2003.

[3] M. Shatnawi and A. Youssef, "Equivalence Detection using ParseTree Normalization for Math Search," Proc. International Conference on Digital Information Management, pp. 643-648, 2007.

[4] H. Zhang, T.B. and M.S. Lin, "An Evolutionary Kmeans Algorithm for Clustering Time Series Data," Proc. International Conference on Machine Learning and Cybernetics, pp. 1282-1287, 2004.

[5] T. Kohonen, Self-organizing Maps, Springer, 1995.

[6] M. Nanni, "Speeding-Up Hierarchical Agglomerative Clustering in Presence of Expensive Metrics," in Advances in Knowledge Discovery and Data Mining, Springer Berlin, pp. 378-387, 2005.

[7] Formula Search, Available at: http://functions.wolfram.com/formulasearch/.

[8] A. Youssef, "Search of Mathematical Contents: Issues and Methods," Proc. International Conference on Intelligent & Adaptive Systems and Software Engineering, pp. 100-105, 2005.

[9] R. Munavalli and M.R. MathFind, "A Math-aware Search Engine," Proc. Annual International ACM SIGIR Conference on Research and development in information retrieval, pp.735-735, 2006.

[10] S.H. Samarasinghe and S.C. Hui, "Mathematical Document Retrieval for Problem Solving," Proc. International Conference on Computer Engineering and Technology, pp.583-587, 2009.

[11] ASCIIMath, Available at: http://www1.chapman.edu/~jipsen/asciimath.html.

[12] Mathematical Markup Language, Available at http://www.w3.org/TR/MathML2.

[13] M. Kohlhase. "Markup for Mathematical Knowledge," An Open Markup format for Mathematical Documents, Ver. 1.2, Lecture Notes in Computer Science, pp. 13-23, Springer Berlin, 2006.

[14] R.B. Yates and B.R. Neto, Modern Information Retrieval, Addison – Wesley Longman Publishing Co. Inc., Boston, MA, 1999.

[15] C.D. Manning, P. Raghavan and H. Schütze, Introduction to Information Retrieval, Cambridge University Press, 2008.