A Structure Based Approach for Mathematical Expression Retrieval

P. Pavan Kumar*, Arun Agarwal, and Chakravarthy Bhagvati

Dept. of Computer and Information Sciences
University of Hyderabad, Hyderabad 500 046, India
pavan.ppkumar@gmail.com,
{aruncs,chakcs}@uohyd.ernet.in

Abstract. Mathematical expression (ME) retrieval problem has currently received much attention due to wide-spread availability of MEs on the World Wide Web. As MEs are two-dimensional in nature, traditional text retrieval techniques used in natural language processing are not sufficient for their retrieval. In this paper, we have proposed a novel structure based approach to ME retrieval problem. In our approach, query given in LaTeX format is preprocessed to eliminate extraneous keywords (like \displaystyle, \begin{array} etc.) while retaining the structure information like superscript and subscript relationships. MEs in the database are also preprocessed and stored in the same manner. We have created a database of 829 MEs in LaTeX form, that covers various branches of mathematics like Algebra, Trigonometry, Calculus etc. Preprocessed query is matched against the database of preprocessed MEs using Longest Common Subsequence (LCS) algorithm. LCS algorithm is used as it preserves the order of keywords in the preprocessed MEs unlike *bag of words* approach in the traditional text retrieval techniques. We have incorporated structure information into LCS algorithm and proposed a measure based on the modified algorithm, for ranking MEs in the database. As proposed approach exploits structure information, it is closer to human intuition. Retrieval performance has been evaluated using standard precision measure.

Keywords: Mathematical expressions, retrieval, longest common subsequence, structure information.

1 Introduction

The availability of mathematical literature on the web has been currently increased due to rigorous research work in the scientific and engineering disciplines. As such, there is an imperative need to provide mathematical expression (ME) retrieval systems for the internet users. For example, a researcher wants to search MEs which are similar in structure to a given expression or a reviewer of a research paper, needs to detect plagiarism of math notation. In these cases,

* Corresponding author.

C. Sombattheera et al. (Eds.): MIWAI 2012, LNCS 7694, pp. 23–34, 2012.

search and retrieval systems for MEs are needed so that users can retrieve similar mathematical structures to a given ME query.

Traditional text retrieval techniques used in natural language processing are not sufficient for ME retrieval due to two-dimensional nature of MEs. ME retrieval methods should also take ME structure (spatial relationships like superscript, subscript etc.) into consideration. Very few works are reported in the literature on this problem and a majority of them are based on text retrieval techniques. In this paper, we have proposed a novel structure based approach to ME retrieval problem.

In our approach, query in LaTeX format [7] is taken and preprocessed to eliminate extraneous keywords like \displaystyle, \begin{array} etc. The preprocessed query still retains the structure information. MEs in the database are also preprocessed and stored in the same manner. As standard datasets are not available, we have created a database of 829 MEs in LaTeX form as part of our work on ME recognition [13]. Those MEs are collected from different mathematical documents and cover various branches like Algebra, Trigonometry, Geometry, Calculus etc. A given query is preprocessed and matched against the preprocessed ME database using *Longest Common Subsequence* (LCS) [3] algorithm (a naive string matching algorithm). LCS algorithm is used as it preserves the order of keywords in the preprocessed MEs unlike *bag of words* approach [15] in the traditional text retrieval techniques. We have incorporated structure information into LCS algorithm and based on the modified algorithm, a measure has been proposed for ranking MEs in the database. As proposed approach exploits structure information, it is called as *structure-based* approach which is closer to human intuition. Retrieval performance is evaluated using standard precision measure [12].

The paper is organized as follows. Existing works on ME retrieval problem are discussed in Section 2. Proposed approach is presented in Section 3. Experimental results are discussed in Section 4 and Section 5 concludes the paper.

2 Related Work

Zanibbi et al. [18] have presented a survey on ME retrieval methods. Zhao et al. [21] have implemented a prototype for math retrieval and reported on the user requirements in creating a digital library that indexes and retrieves math content. Miner et al. [10] has built math-aware search engine on the top of text-based one, in which MathML [9] query is converted into text-encoded math query terms. Kohlhase and Sucan [6] have used substitution tree indexing [4] in ME retrieval by adding all sub-expressions in the document database to the substitution tree. They have performed retrieval using backtracking search over variable bindings in the tree. In [5,17], set-based measures based on subtree matching between two MathML trees are used.

In [20], content-based image retrieval [12] approach has been followed, where ME images are segmented into connected components and matched using contour and density features. In [19], queries take the form of a handwritten expression image, for which X-Y tree is generated for them. Retrieval in their approach

is based on dynamic time warping on feature vectors computed using X-Y trees. These tree matching based approaches are computationally intensive.

Adeel et al. [1] have used vector space model on MathML representation. In their approach, keywords or terms like matrix, root etc, are generated using template matching rules for the MathML expression and then standard text retrieval techniques based on *term frequency (tf)* and *inverse document frequency (idf)*, are applied. Term frequency gives the number of occurrences of a term in an expression. Inverse document frequency for a term gives the number of MEs containing that term, in the database and hence its computation needs to scan all the database MEs. A vector of dimension determined by the number of terms, is created for the query where each dimension holds normalized *tf*idf* value (product of *tf* and *idf*) for the corresponding term. Matching is performed using dot product of query and database vectors.

Some other systems for ME indexing and retrieval like Springer LaTeX search [16], Lucene library [8] etc, are also based on vector space model. In [11], MEs are indexed based on *n-grams* computed from MathML representations and retrieval is performed using Lucene library. In vector space model based methods, all the database vectors have to be recomputed if the database is updated, as *idf* values depend on the number of database expressions. They are also computationally intensive as *idf* for each term in the query has to be computed. Some approaches [1,2] cluster database vectors to speed up retrieval process by considering only cluster centroids which may reduce the retrieval performance as complete information has not been used.

3 Proposed Approach

As mentioned earlier, in the proposed structure-based approach, which is shown in the form of block diagram in Fig. 1, query in LaTeX format is preprocessed to eliminate extraneous keywords. This preprocessed query still retains the structure information and the same preprocessing is performed on all MEs in the database. Generation of preprocessed database is a one time process which is shown as dotted arrows in Fig. 1. Preprocessed query is matched against the preprocessed database using LCS algorithm. Structure information is incorporated into the algorithm and a measure based on the modified LCS algorithm has been proposed to rank MEs in the database. Retrieval performance is evaluated using standard precision measure. Proposed approach is discussed in detail in the following subsections.

3.1 Query and Database Preprocessing

In the proposed approach, query as well as MEs in the database in LaTeX form are preprocessed to eliminate extraneous keywords in LaTeX representation. LaTeX format may have extraneous keywords like \displaystyle, \begin{array} etc, which have to be ignored. Sum-like operators (eg: $\sum$, $\prod$ etc.), *fraction*, *squareroot* etc, are not atomic in LaTeX form. For example, *squareroot* in LaTeX

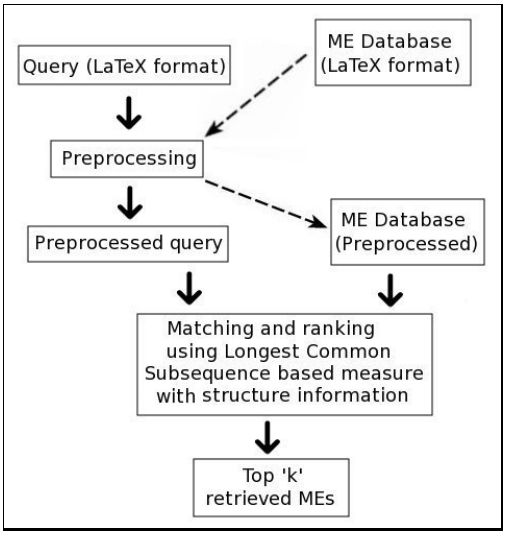

Fig. 1. Proposed approach to ME retrieval

is given by the function \sqrt, which has five symbols. Similarly, sin is written as \sin, which has four symbols. These functions need to be made atomic, without which their individual symbols also involve in matching that is not intuitive. This issue is also considered in the preprocessing step.

Given a ME in LaTeX form, the LaTeX string is scanned from left to right and in the process of scanning, extraneous keywords are eliminated. To make functions atomic, we have created a mapping table that maps LaTeX functions to integer labels. Using that table, each LaTeX function is mapped to an unique integer label. For example, \frac is mapped to 300, \sqrt to 301 etc. Thus LaTeX functions are made atomic using mapping table while scanning the given LaTeX string. Let each atomic entity in the preprocessed string be called as *Term*.

As mentioned in [18], matching can be of any type like exact, instantiation or generalisation. For example, all the variables are considered same with generalisation that is similar to unification process used in AI systems [14]. For example, two MEs $\cos^2 x + \sin^2 x = 1$ and $\cos^2 y + \sin^2 y = 1$ give the same formula but differ only in their variables. If x and y are mapped to same term, then both of them represent the same formula. Similarly, **tan** 30 is an instance of **tan** x (for $x = 30$) and if an user wants to consider all the instances as relevant, numerals can be considered as the instances of variables (instantiation). Retrieval depends on match type and the match type is based on the user requirements. In our approach, any type of match can be easily incorporated using mapping table. We have mapped variables and numerals in MEs to special terms. Let these terms be denoted by V and N respectively. As English alphabet, greek symbols etc, are generally used as variables, they are mapped to V and similarly, numerals are mapped to N.

Structure information like superscript, subscript etc, is retained and treated in a different manner in the preprocessed string. For example, in LaTeX, superscript is written after ^ and subscript after _ (underscore). Four special integer labels called *Structure terms* are introduced to designate start and end of superscript and subscript sub-expressions. Let P_s and P_e (B_s and B_e) denote the structure terms used for start and end of superscript (subscript) respectively. During preprocessing of a LaTeXstring, P_s and P_e (B_s and B_e) are added before and after scanning superscript (subscript) sub-expressions recursively. For the sake of uniformity, we have assumed numerator and denominator for *fraction* as its superscript and subscript respectively. Similarly, degree and contained expression for *squareroot* are assumed as its superscript and subscript respectively. The above assumption allows to use the same structure terms before and after the numerator and denominator (degree and contained expression) for *fraction* (*squareroot*).

For example, consider x^n. Its LaTeX string is x^n. Its preprocessed string is given by: V, P_s, V, P_e. Here, first and second V terms denote x and n respectively and P_s and P_e are written before and after superscript term V (n). Similarly, consider $\sqrt[n]{x_1}$. Its LaTeX string is \sqrt [n] {x_1}. Its preprocessed string is given by: 301 (term for *squareroot*), P_s, V, P_e, B_s, V, B_s, N, B_e, B_e. Here, it can be seen start and end superscript structure terms are added recursively. Superscript and subscript structure terms are added for degree n and the contained expression x_1 respectively. For x_1, again, subscript structure terms are added for 1. Variable x and numeral 1, are mapped to V and N respectively.

Elements in matrices, enumerated functions etc, are also processed in the same manner as discussed above. Row and element delimiters (\\ and & in LaTeX) are mapped to special terms, which are denoted by R and C respectively. That means, R and C are written after each row and column respectively in the preprocessed string.

Consider $\begin{bmatrix} x & y \\ x_1 & y_2 \end{bmatrix}$ for example. Its LaTeX string is given by: \left[\begin {array} {cc} x & y \\ x_1 & y_2 \end{array} \right]. Its preprocessed string is given by: 201, V, C, V, R, V, B_s, N, B_e, C, V, B_s, N, B_e, R, 202. Here, 201 and 202 are labels for the functions \left[and \right] respectively. It can be observed that the extraneous keywords are eliminated, R and C are written after each row and element respectively.

Preprocessing is applied on all MEs in the database to generate a database of preprocessed MEs and this is a one time process as shown in Fig. 1. A given query in LaTeX form is preprocessed and matched against this preprocessed database to retrieve similar MEs.

3.2 Matching and Ranking

Proposed approach uses LCS algorithm for matching query and database MEs. Given two strings, LCS algorithm [3] finds the longest common subsequence between them. In vector space model [18], *bag of words* representation is used, where order of terms is ignored [15]. Unlike *bag of words* approach, LCS

algorithm preserves the order as it matches the terms in a left to right manner. It is also closer to human intuition as humans ought to retrieve MEs by inspecting sub-expressions (structure), which have ordered terms.

Structure information is incorporated into the LCS algorithm and we have quantified structure using levels of terms in the preprocessed string. Level of a term in a ME is its nested depth (that starts from 0) in that ME. For example, in b_i^2, b is at level 0, and i and 2 are at level 1 as they are at same nested depth from b. Levels are computed for each term in the preprocessed string and the ordered string of levels is called as *Level string*.

Level Strings. After preprocessed string is obtained for a given ME, levels for each term (including structure terms) in it, are computed. Structure terms are given the same level values as those of terms that are enclosed by them. For example, preprocessed string of x^n is given by V, P_s, V, P_e. Level of first V is 0 and that of second V is 1. As P_s and P_e enclose second V, they are given the same level value, which is 1. Consider the preprocessed string given in the subsection 3.1, for $\sqrt[y]{x_1}$: 101, P_s, V, P_e, B_s, V, B_s, N, B_e, B_e. Its level string is given by: 0, 1, 1, 1, 1, 1, 2, 2, 2, 1. Here, subscript 1 of x is at level 2 (from *squareroot*) and its enclosing structure terms are also given the same level value.

For the query, level string has to be computed after its preprocessed string is obtained. In fact, it is computed while scanning LaTeXstring to get the preprocessed string. For the database, level strings are readily computed and stored along with the preprocessed strings.

LCS Based Ranking Measure. Let Q and D denote preprocessed strings for query and a database ME. Let p and q be the lengths of Q and D respectively. Let $Q[i]$ and $D[i]$ denote the i^{th} terms of Q and D respectively. LCS [3] is computed using dynamic programming approach to get polynomial time complexity ($O(n^2)$). Let LCS denotes the dynamic programming matrix, whose dimension is given by $(p+1)\times(q+1)$. $LCS[i,j]$ gives the length of LCS by considering first i terms of Q and first j terms of D and $LCS[p,q]$ gives the length of complete LCS between Q and D. If atleast one of the strings is empty, LCS between them is 0. That means, $LCS[0,0] = LCS[i,0] = LCS[0,j] = 0$. $LCS[i,j]$ is given below:

$$LCS[i,j] = \begin{cases} 0 & \text{if } i = 0 \text{ or } j = 0 \\ LCS[i-1,j-1]+1 & \text{if } Q[i] = D[j] \\ \max(LCS[i,j-1], LCS[i-1,j]) & \text{if } Q[i] \neq D[j] \end{cases} \quad (1)$$

It can be seen from equation 1, traditional LCS algorithm gives a match score of 1 for each match of terms and it does not consider structure match. We have modified the algorithm so that structure match is also considered. As we have quantified structure in terms of levels, in our approach, two terms are considered as matched if they are same and their levels are also equal. That means, a match score of 1 is given if two terms as well as their levels are equal. If two terms are same, match score that is a function of absolute difference in

their levels, is assigned. If absolute difference is less, more score value is given and vice-versa. Therefore, match score between two terms $Q[i]$ and $D[j]$ denoted by $score(Q[i], D[j])$ is given by equation 2.

$$score(Q[i], D[j]) = \frac{1}{|l(Q[i]) - l(D[j])| + 1} \tag{2}$$

Here, $l(x)$ denotes the level of the term x and $|x|$ gives the absolute value of x. If $Q[i] = D[j]$, a match score given by equation 2 is assigned. If their levels are equal ($l(Q[i]) = l(D[j])$), a match score of 1 is assigned. Otherwise, a value less than 1 is assigned. As seen from equation 2, more difference results in less score and vice-versa. As modified LCS algorithm takes into account, not only the term match but also their structure match (using equation 2), it is also called as *Structure based LCS* (SLCS) algorithm. SLCS algorithm, where $SLCS$ is used instead of LCS, is given by equation 3.

$$SLCS[i,j] = \begin{cases} 0 & \text{if } i = 0 \text{ or } j = 0 \\ SLCS[i-1, j-1] + score(Q[i], D[j]) & \text{if } Q[i] = D[j] \\ \max(SLCS[i, j-1], SLCS[i-1, j]) & \text{if } Q[i] \neq D[j] \end{cases} \tag{3}$$

$SLCS[p, q]$ which is computed recursively using equation 3, gives term as well as their structure matches between Q and D, and let it be denoted by $Match(Q, D)$. For ranking, this value should be normalized between 0 and 1 by considering mismatched terms. Mismatched terms are the ones in both Q and D, that are not matched by $SLCS$. If the number of mismatched terms is more, rank should be reduced and vice-versa. Let $MisMatch(Q, D)$ be the number of mismatched terms in both Q and D. Therefore, rank of D with respect to Q denoted by $rank(Q; D)$, is given by equation 4.

$$rank(Q; D) = \frac{Match(Q, D)}{Match(Q, D) + MisMatch(Q, D)} \tag{4}$$

In equation 4, if both Q and D are same (perfect match), $Match(Q, D)$ is equal to number of terms in Q or D, $MisMatch(Q, D) = 0$ and so $rank(Q; D) = 1$. If Q and D are entirely different (complete mismatch), $Match(Q, D) = 0$, $MisMatch(Q, D)$ is equal to total number of terms in both Q and D and so $rank(Q; D) = 0$. Otherwise, rank lies between 0 and 1. Database ME that has more number of matches and less number of mismatches to the query, gets a high rank value and vice-versa.

Rank values with respect to a given query are computed for all MEs in the preprocessed database and are sorted in a decreasing order. If two or more MEs in the database get the same rank value, they are sorted in the increasing order of absolute differences between number of terms in the query and that in each of those MEs. If two or more MEs have the same rank as well as the absolute difference values, they are ordered on the *first come first serve* basis (appearance) in the database. Top k MEs are then retrieved and shown to the user.

As LCS based algorithm is used, each match takes $O(n^2)$ time. Let T be the total number of database MEs. Matching of T MEs takes $O(Tn^2)$ and sorting T rank values takes $O(T \ln T)$ time [3]. Therefore, entire retrieval process takes $O(Tn^2) + O(T \ln T)$ and as T is fixed, it takes only $O(n^2)$ time.

An Illustrative Example. We have presented an example in Fig. 2. In this example, two limit ME structures are shown. Preprocessed strings of the query and database MEs, that are denoted by Q and D respectively, are shown in Fig. 2. Their level strings are also shown in this figure. Terms 225, 129 and 135 denote the keywords \lim, \rightarrow and \infty respectively and the remaining terms are explained earlier. Using equations 2, 3 and 4, $Match(Q, D) = 11$, $MisMatch(Q, D) = 2$ and $rank(Q; D) = 0.85$. That means, D matches with Q by 85%. Here, as mentioned earlier, if constant ∞ is considered as an instantiation of a, both the MEs are perfectly matched with a rank of 1.

Top 5 retrieved MEs for the query Q given in Fig. 2 are shown in Table 1 in the decreasing order of their ranks. For each of them, say D, number of terms in D, absolute difference in the number of terms of Q and D, match and mismatch scores, and ranks are shown. Rank values are shown upto two decimal points. It can be seen in the table, first ME with a rank of 1, is the query itself as it is also present in the database. Both third and fourth ones have same ranks as well as the absolute differences and hence ordered based on their appearance in the database. It can be observed that second ME has less number of matches than those of third and fourth MEs. But third and fourth ones have more number of mismatches than that of second one and so their ranks are less than that of the second one. Similarly, last ME has same number of matches but more number of mismatches than that of the second one and hence its rank is less than that of the second one.

3.3 Retrieval Performance

To evaluate retrieval performance, we have used standard precision measure [12]. Precision is the ratio of the number of relevant MEs retrieved to the total number of MEs retrieved. As mentioned earlier, the notion of relevance is purely subjective. Precision for the query example given in Fig. 2 is 100%. In this example, all the retrieved MEs have similar limit (lim) structures and hence all of them are relevant.

4 Experimental Results and Discussion

An enumerated function example is shown in Table 2. Top 5 retrieved MEs along with their match and mismatch scores, and ranks are shown in the table. In this table, first database ME with rank value 1 is the query itself. Match scores and the rank values are shown upto two decimal points in the table. It can be seen from the table, that second and third MEs are enumerated functions that have similar structures to that of the query.

Query: $\lim_{x \to a} f(x) = L$
Database ME: $\lim_{x \to \infty} f(x) = L$

Preprocessed Query (Q):
225, B_s, V, 129, V, B_e, V, (, V,), =, V
Level string of preprocessed query:
0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0

Preprocessed database ME (D):
225, B_s, V, 129, 135, B_e, V, (, V,), =, V
Level string of preprocessed database ME:
0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0

Fig. 2. An example of preprocessed query (Q) and database ME (D) along with their level strings

Table 1. Top 5 retrieved MEs for the query Q in Fig. 2. Number of terms in $Q = 12$.

Database ME (D)	Number of terms in D	Absolute difference in the number of terms of Q and D	$Match(Q, D)$	$MisMatch(Q, D)$	$rank(Q; D)$
$\lim_{x \to a} f(x) = L$	12	0	12	0	1
$\lim_{x \to \infty} f(x) = L$	12	0	11	2	0.85
$\lim_{x \to a^+} f(x) = L$	15	3	12	3	0.8
$\lim_{x \to a^-} f(x) = L$	15	3	12	3	0.8
$\lim_{x \to -\infty} f(x) = L$	13	1	11	3	0.79

Fourth ME is not an enumerated function, but it has several matching sub-expressions (like $ax^2 + bx + c$ under squareroot, on which integral is applied) to the query and matches the query by only 33%. Similarly, fifth ME has some matching sub-expressions like integrals, squareroots, *ln* function applied on the absolute value of a sub-expression etc. But it matches the query by only 33%.

As the rank value gives the percentage of match, a threshold on it can be used to filter out MEs. For example, MEs that have rank value of atleast 0.5 (50% match) can be retrieved. Using this threshold in the above example, only first three MEs are retrieved and the last two MEs are filtered. If an user seeks only enumerated functions, without thresholding, precision is 60% ($\frac{3}{5} \times 100$). With thresholding, only three are retrieved and all of them are relevant to the user and so precision is 100%.

We have presented results on ten queries taken from our database, in Table 3. As mentioned earlier, we have created a database of 829 MEs in LaTeX form that covers different branches of mathematics. Queries shown in Table 3 span branches like Algebra, Trigonometry, Geometry and Calculus. For each query, precision percentages without and with thresholding rank values, on the top k retrieved MEs (for $k = 5$ and 10) are shown. The threshold value is taken as 0.5 (atleast 50% match) and the results shown in Table 3 are discussed below.

Table 2. Top 5 retrieved MEs for the enumerated function query that is same as the first retrieved ME

Database ME (D)	$Match(Q,D)$	$MisMatch(Q,D)$	$rank(Q;D)$
$\int \frac{dx}{ax^2+bx+c} = \begin{cases} \frac{1}{\sqrt{b^2-4ac}} \ln \mid \frac{2ax+b-\sqrt{b^2-4ac}}{2ax+b+\sqrt{b^2-4ac}} \mid, & \text{if } b^2 > 4ac, \\ \frac{2}{\sqrt{4ac-b^2}} \arctan \frac{2ax+b}{\sqrt{4ac-b^2}}, & \text{if } b^2 < 4ac, \end{cases}$ (Query)	147	0	1
$\int \frac{dx}{\sqrt{ax^2+bx+c}} = \begin{cases} \frac{1}{\sqrt{a}} \ln \mid 2ax+b+2\sqrt{a}\sqrt{ax^2+bx+c} \mid, & \text{if } a > 0, \\ \frac{1}{\sqrt{-a}} \arcsin \frac{-2ax-b}{\sqrt{b^2-4ac}}, & \text{if } a < 0, \end{cases}$	83	66	0.56
$\int \frac{dx}{x\sqrt{ax^2+bx+c}} = \begin{cases} \frac{-1}{\sqrt{c}} \ln \mid \frac{2\sqrt{c}\sqrt{ax^2+bx+c}+bx+2c}{x} \mid, & \text{if } c > 0, \\ \frac{1}{\sqrt{-c}} \arcsin \frac{bx+2c}{\mid x \mid \sqrt{b^2-4ac}}, & \text{if } c < 0, \end{cases}$	87.83	72	0.55
$\int \sqrt{ax^2+bx+c}dx = \frac{2ax+b}{4a}\sqrt{ax^2+bx+c} + \frac{4ax-b^2}{8a} \int \frac{dx}{\sqrt{ax^2+bx+c}},$	50.67	101	0.33
$\int \frac{x}{\sqrt{a+bx}} dx = \frac{1}{\sqrt{2}} \ln \mid \frac{\sqrt{a+bx}-\sqrt{a}}{\sqrt{a+bx}+\sqrt{a}} \mid, a > 0,$	51	103	0.33

All the retrieved MEs are relevant for the first three queries and they have rank values greater than 0.5. Hence precision is 100% for these queries, whether or not thresholding is performed. For them, there are more than ten MEs which have similar structures (differentials, integrals and trigonometric functions), in the database and hence precision is 100% even after 10 MEs are retrieved.

For the fourth query (matrix determinant), 3 out of 5 are relevant (60% precision). The remaining two MEs are not relevant but they have rank values less than 0.5 and so a precision of 100% is obtained after thresholding. Precision is reduced further to 30% if top 10 MEs are retrieved as only 3 of them are relevant. As the remaining seven MEs have rank values less than 0.5, after thresholding, a precision of 100% is obtained. Similarly, for the fifth query, four out of the retrieved MEs (5 or 10) are relevant and a precision of 100% is obtained after thresholding.

For the sixth query, only four of the retrieved MEs (5 or 10) are relevant. Eventhough the remaining MEs are not relevant, they have rank values around 0.52 (greater than but in the neighbourhood of the threshold) and so they are not filtered by thresholding. Hence precision values of 80% and 40% are obtained on the top 5 and 10 retrieved MEs respectively, irrespective of thresholding.

For the seventh query, three of the retrieved MEs are relevant and so a precision of 60% is obtained. Remaining MEs that are not relevant, are filtered by thresholding to get 100% precision. All the five retrieved MEs are relevant for the eighth query and so it gets 100% precision. After ten MEs are retrieved, 3 more relevant MEs are obtained. Hence, for this query, precision is reduced to 80% as only 8 out of 10 MEs are relevant and the two non-relevant MEs are filtered to get 100% precision.

All of the top 5 retrieved MEs are relevant for the ninth and tenth queries. But only 9 and 6 MEs respectively are relevant for them in the top 10 retrieved MEs so that precision values of 90% and 60% are obtained respectively. Non-relevant MEs are filtered to get 100% precision for both of them.

Table 3. Ten queries and their precision percentages without and with thresholding rank values (at 0.5) on top k retrieved MEs for $k = 5$ and 10

S.No	Query	Precision percentage on top k retrieved MEs			
		$k = 5$		$k = 10$	
		Without thresholding	With thresholding	Without thresholding	With thresholding
1	$\frac{d}{dx}(cx^n) = ncx^{n-1}$	100	100	100	100
2	$\int udv = uv - \int vdu$	100	100	100	100
3	$\sin^2 x = 1 - \cos^2 x$	100	100	100	100
4	$\begin{vmatrix} x & y & 1 \\ x_0 & y_0 & 1 \\ x_1 & y_1 & 1 \end{vmatrix} = 0$	60	100	30	100
5	$\frac{\pi}{4} = 1 + \frac{1^2}{2+\frac{3^2}{2+\frac{5^2}{2+\frac{7^2}{2+\cdots}}}}$	80	100	40	100
6	$\frac{(x-h)^2}{a^2} + \frac{(y-k)^2}{b^2} = 1$	80	80	40	40
7	$\cos\theta = \frac{(x_1,y_1)\cdot(x_2,y_2)}{l_1 l_2}$	60	100	30	100
8	$S(x) = \Sigma_{d\|x} d = \Pi_{i=1}^{n} \frac{p_i^{e_i+1}-1}{p_i-1}$	100	100	80	100
9	$\int \sec^n x dx = \frac{\tan x \sec^{n-1} x}{n-1} + \frac{n-2}{n-1}\int \sec^{n-2} x dx, n \neq 1$	100	100	90	100
10	$\lim_{h\to 0} \frac{1}{h}(\frac{-h}{x(x+h)}) = \lim_{h\to 0} \frac{-1}{x(x+h)} = -\frac{1}{x^2}$	100	100	60	100

5 Conclusions and Future Directions

In this paper, we have proposed a novel structure based approach to ME retrieval problem. In our approach, query given in LaTeX format is preprocessed to eliminate extraneous keywords while retaining the structure information. Database MEs are also preprocessed and stored in the same manner. We have created a database of 829 MEs in LaTeX form, that covers various branches of mathematics like Algebra, Trigonometry, Calculus etc. Preprocessed query is matched against the database of preprocessed MEs using LCS algorithm. As it preserves the order of terms in the preprocessed MEs unlike *bag of words* approach in the traditional text retrieval techniques, LCS algorithm is chosen.

Structure information is incorporated into LCS algorithm and based on the modified algorithm, a measure has been proposed for ranking database MEs. As proposed approach is based on structure information, it is closer to human intuition. As LCS based algorithm is used, time complexity of the proposed approach is $O(n^2)$. Retrieval performance has been evaluated using standard precision measure. As a future work, proposed approach can be integrated to ME recognition system which converts a query given in the image format to LaTeX form and then MEs are retrieved from the database.

References

1. Adeel, M., Cheung, H.S., Khiyal, A.H.: Math go! prototype of a content based mathematical formula search engine. Journal of Theoretical and Applied Information Technology 4(10), 1002–1012 (2008)

2. Adeel, M., Sher, M., Khiyal, M.S.H.: Efficient cluster-based information retrieval from mathematical markup documents. World Applied Sciences Journal 17, 611–616 (2012)
3. Cormen, T.H., Leiserson, C.E., Rivest, R.L.: Introduction to Algorithms. The MIT Press and McGraw-Hill Book Company (1989)
4. Graf, P.: Substitution Tree Indexing. In: Hsiang, J. (ed.) RTA 1995. LNCS, vol. 914, pp. 117–131. Springer, Heidelberg (1995)
5. Kamali, S., Tompa, F.W.: Improving mathematics retrieval. In: Proceedings of Digital Mathematics Libraries, Grand Bend, pp. 37–48 (2009)
6. Kohlhase, M., Sucan, I.: A Search Engine for Mathematical Formulae. In: Calmet, J., Ida, T., Wang, D. (eds.) AISC 2006. LNCS (LNAI), vol. 4120, pp. 241–253. Springer, Heidelberg (2006)
7. Lamport, L.: LaTeX: A Document Preparation System. Addison-Wesley (1986)
8. Lucene: Indexing and retrieval library, `http://lucene.apache.org`
9. MathML (2010), `http://www.w3.org/Math/`
10. Miner, R., Munavalli, R.: Mathfind: A math-aware search engine. In: Proceedings of the International Conference on Information Retrieval, New York, USA, pp. 735–735 (2006)
11. Miner, R., Munavalli, R.: An Approach to Mathematical Search Through Query Formulation and Data Normalization. In: Kauers, M., Kerber, M., Miner, R., Windsteiger, W. (eds.) MKM/CALCULEMUS 2007. LNCS (LNAI), vol. 4573, pp. 342–355. Springer, Heidelberg (2007)
12. Müller, H., Müller, W., Squire, D.M., Marchand-Maillet, S., Pun, T.: Performance evaluation in content-based image retrieval: overview and proposals. Pattern Recognition Letters 22(5), 593–601 (2001)
13. Pavan Kumar, P., Agarwal, A., Bhagvati, C.: A Rule-Based Approach to Form Mathematical Symbols in Printed Mathematical Expressions. In: Sombattheera, C., Agarwal, A., Udgata, S.K., Lavangnananda, K. (eds.) MIWAI 2011. LNCS, vol. 7080, pp. 181–192. Springer, Heidelberg (2011)
14. Rich, E., Knight, K.: Artificial Intelligence, 2nd edn. McGraw-Hill Book Company (1991)
15. Salton, G., McGill, M.J.: Introduction to Modern Information Retrieval. McGraw-Hill, Inc., New York (1986)
16. Springer: LaTeX search, `http://www.latexsearch.com/`
17. Yokoi, K., Aizawa, A.: An approach to similarity search for mathematical expressions using mathml. Towards a Digital Mathematics Library, Grand Bend, pp. 27–35 (2009)
18. Zanibbi, R., Blostein, D.: Recognition and retrieval of mathematical expressions. In: IJDAR. Springer, Heidelberg (2011)
19. Zanibbi, R., Yu, L.: Math spotting: Retrieving math in technical documents using handwritten query images. In: 2011 International Conference on Document Analysis and Recognition, pp. 446–451 (2011)
20. Zanibbi, R., Yuan, B.: Keyword and image-based retrieval of mathematical expressions. In: Document Recognition and Retrieval XVIII, vol. 7874, pp. 1–10. SPIE (2011)
21. Zhao, J., Kan, M.Y., Theng, Y.L.: Math information retrieval: user requirements and prototype implementation. In: Proceedings of the 8th ACM/IEEE-CS Joint Conference on Digital Libraries, JCDL 2008, pp. 187–196. ACM, New York (2008)