MATH GO! PROTOTYPE OF A CONTENT BASED MATHEMATICAL FORMULA SEARCH ENGINE

Muhammad Adeel[1][2], Hui Siu Cheung[2], Sikandar Hayat Khiyal[3]

1Dept of Computer Science, Faculty of Basic & Applied Sciences, International Islamic University, Islamabad, Pakistan
School of Computer Engineering, Nanyang Technological University, Singapore
2Department of Computer Science/Software Engineering, Fatimah Jinnah Women University, Islamabad, Pakistan

m.sikandarhayat@yahoo.com
asschui@ntu.edu.sg
madeel@ntu.edu.sg, adeel@iiu.edu.pk

ABSTRACT

In recent years, mathematical content has started appearing on the web. As with other types of information, search capabilities must be provided to users to retrieve their required information. Conventional text-based search engines fall short of providing math-aware fine grain search. Several efforts have been made in the recent past to develop web-accessible mathematical and scientific search systems. We identify major issues in developing math search systems and present techniques for addressing them. The issues involved in developing such a system include query format, representation of mathematical content, as well as their comparison for matching purposes. The results generated by the formula search system must strive to maximize the standard measures of precision, recall. It should also sort the query hits according to some domain specific measure and provide the hits to the user as a ranked list. This paper presents our efforts in developing a mathematical search engine for mathematical content retrieval. We present the Math GO! System to search and present the mathematical information encoded in mathematical expressions. Our approach uses the concept of template based math block identification, vector representation, searching from mathematical topic based clusters and relevance ranking. The search system interacts with the user through a simple query mechanism and provides ranked listing of results. The system is comprised of a modular architecture to organize, query, compare and presentation of math results to the user. We compare our approach with existing approaches and present the results. We have achieved encouraging results with the system.

Keywords: *Formula search, mathematical prototype, Math Go*

1. INTRODUCTION

The mathematical literature being published on the web is increasing day by day [6][8][9][10][11][12][13]. More and more users are turning over to the internet to publish math content, share information and find solutions to their mathematical problems. Clearly user needs specialized search systems to find the math content relevant to their requirements. Efforts are underway for the development of such systems. Usually the internet search engines [1][2] are used to formulate a mathematical query consisting of keywords. Unfortunately, the conventional text based search engines fail to recognize the special mathematical symbols and constructs. Some math search engines [3][4] available on the internet also follow the keyword based approach. Although some time the results are relevant but this approach fails to work most of the time. This is because the major constituent of a mathematical document are its equations. It is estimated that over 80% of the Digital Library of Mathematical Functions (DLMF) handbook/website [6] contents are equations [14]. Therefore, for a math search engine, the capability to search and retrieve mathematical equations is fundamental. As such, there are unique problems posed by the challenge of storing and retrieval of such specific documents.

Furthermore, synonym is also a problem in mathematical expressions where some terms share high semantic information. Such synonym matching is not provided/utlizlied by the contemporary search systems. Math content has many ambiguities (same symbol used for different purpose etc) which are not understood by the current search systems. Math content lends itself toward the formation of taxonomy and hierarchy which is not handled by the present search systems.

The development of math-aware search system is a new area of research with many challenges. These challenges include but are not limited to recognition of mathematical symbols and structures, mathematical expression matching, query hit ranking etc. Some progress has been achieved in this regard by [8][9][10][11] but there are still many issues to resolve.

The primary goal of this research is to create a search system

1. enable user to search for mathematical formula content
2. allow users to express math queries naturally and easily
3. provide ranking of results to the user
4. provide hybrid search modes (hierarchy of topics as well as free style search)
5. improve retrieval performance by clustering the questions and hence investigate the effect of clustering on formula retrieval
6. an extensible framework to add full math document search later on

This paper will identify the issues involved in meeting the goals mentioned above, and explicate the shortcomings of the present search systems to handle content based math search and retrieval. Afterwards, the paper will introduce specific approaches and specific techniques for addressing some of the major issues. Lastly, preliminary results indicating the potential of our search system will be presented. We have achieved encouraging results with the system.

The rest of this paper is organized as follows. Section 2 discusses the architecture of the mathematical search engine. Section 3 discusses the improvements to the recognition of mathematical expressions and the use of MathML as the markup language. Section 4 presents retrieval of mathematical content in detail. Section 5 gives some initial experimental results. Finally we conclude in Section 6 with the conclusions and the directions for future research.

2. RELATED WORK

One of the most recent efforts in the field of math content retrieval is the Digital Library of Mathematical functions (DLMF) project [6]. This is a work in progress and full math search is still not available. This will serve as a major mathematical reference source on the web for special functions and their applications [14]. Other research prototypes and systems [3][4] assist in finding mathematical problems. However, when finding appropriate mathematical expressions, most of these systems only support mechanisms to search expression in a strict exact manner, or search some similar problems based on wildcard, not on the similarity of expression structures and semantic meanings. Such mechanisms restrict users significantly from achieving meaningful and accurate search results of mathematical expressions. MathDeX [1011], a web based math-aware search engine, is developed by Design Science as part of an NSF grant to facilitate math search. It indexes LaTeX as well as Presentation MathML.

A new query language MathQL for searching math content was proposed in [12]. Our focus is more inclined towards the better retrieval. Our template based query language is natural and intuitive to use. There are other mathematical knowledge management efforts which are not related directly to math search. They can have an indirect effect on the math search. E.g., research in the development of better user interfaces can lead to improved mathematics retrieval.

There are early research prototypes and systems for math search [3][4]. Other mathematical knowledge management efforts like [1012] are indirectly related to math search. E.g., research in the development of better user interfaces can lead to improved mathematical content retrieval.

3. MATH FORMULA SEARCH ISSUES

Our survey of existing search systems and experiences with building a search system has revealed several issues. These issues must be addresses in the development of a math search system engine. They are listed below. Approaches to address these issues will be presented later.

1. *Search Mode*
   Search Modes have two main types. One is the static and limited approach. Table of Contents, hierarchy of topics and topic based

ontology are example of static mode.[1]. The other is the dynamic and free form approach. The typical text based web search engines on the web [1][2] use this second mode. Hybrid of the two approaches is menu driven search which can be based on ontology or constrained/standard vocabulary. Selection of a suitable search mode is a critical design issue in math search.

2. *Query Format*
   It is obvious that when the search space is comprised of domain specific topics, a hybrid form of technique is required. Development of a query language that is intuitive, natural and consistent is vital for a mathematical formula search engine. The query format should allow user to express his query in a natural and intuitive manner. The search system should be able to understand different mathematical symbols and structures. (different forms of differentials, complex integrals etc).
3. *Math Formula Searching*
   Mathematical expressions have the semantic meaning hidden in their representation. Proper indexing of mathematical expressions requires the extraction of metadata from the expressions. The standard text IR techniques of preprocessing, normalization etc need to be adapted in mathematical context. After Textualization, scoping and normalization of mathematical expressions, they can be stored in database to be used for retrieval purpose. Same operation needs to be applied on the query before the start of search operation.
4. *Matching*
   The most important design issue in a search engine is the expression matching. Without proper matching, the search engine would fail to provide relevant results to the user. This would result in high user dissatisfaction with the search engine.
5. *Relevance* Ranking
   Ranking of results is very convenient for the user. Imagine a web site retrieving all the required results but they are just presented to the user without being ranked. This can result in high user dissatisfaction from the system.
6. *Performance*
   It is important for a search engine to present results to the user quickly. The latency involved must not exceed a few seconds. A good search system finds the most relevant results quickly. (preferably most queries should take milliseconds to complete.)

These are the major issues which we have identified for the development of a math search engine. Next, we present the approaches adopted to address the issues in the MathGo! Search system.

Math Representation Issues

A mathematical problem consists of textual data, mathematical expressions and optionally diagrams. For mathematical expression, a special encoding mechanism is required for the representation of mathematical expressions. The two best known open markup formats for representing mathematical fornulae for the web are MathML [7] and OpenMath [17]. MathML provides a standard way of representing math expressions. It is an XML application for describing mathematical notation and is used to encode mathematical content within text format.

In MathML, there are two styles of encoding, content encoding and presentation encoding [14], dealing with the meaning and display of formulae respectively. For example, let us consider the $(a + b)^2$ expression. This expression naturally breaks into a "base," the (a + b), and a "script," which is the single character '2' in this case. The base decomposes further into a sequence of two characters and three symbols. Of course, the decomposition process terminates with indivisible expressions such as digits, letters, or other symbol characters. The MathML presentation encoding of this expression is shown in Figure 2.7. 18

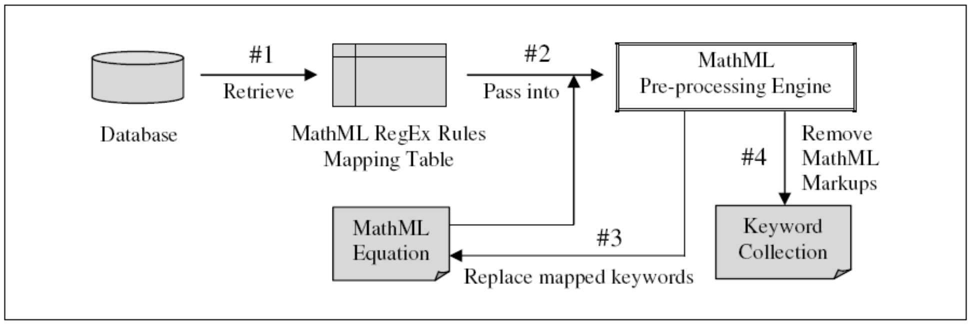
```
<apply>                                  <msup>
        <power/>                                 <mfenced>
        <apply>                                          <mi>a</mi>
                <plus/>                                  <mo>+</mo>
                <ci>a</ci>                               <mi>b</mi>
                <ci>b</ci>                       </mfenced>
        </apply>                                 <mn>2</mn>
        <cn>2</cn>                       </msup>
</apply>
```

Fig 1: MathML Content / Presentation encoding

[1] http://www.seas.gwu.edu/~ayoussef/search/Search-IMA-4-04-web/presentation.html

Formula Retrieval

Before the retrieval step can be performed, the MathML obtained from the equation editor is preprocessed. Conventional text preprocessing mechanism cannot be applied to mathematical expressions due to the presence of different markup tags with special meaning. We use regular expressions to match some patterns or key symbols in MathML to the appropriate keywords. A Regular expression is a special text string for describing a search pattern.



Step 1: The Regular Expression (RegEx) rules and corresponding keywords are retrieved together from the database.

Step 2: The MathML Pre-processing Engine will compare the fetched RegEx rules with the MathML Equation for patterns and symbols to be discovered.

Step 3: Once a pattern or symbol has been discovered, that pattern or symbol will be replaced with the corresponding mapped keyword(s) present in the MathML Equation.

Step 4: The remaining MathML markup tags will be removed, leaving only the keywords behind.

The regular expression approach allows us to index math expressions with a standard text IR system. Both the equations entered by the user and the equations saved in the database have to pass through the preprocessing stage. This results in the same intermediate representation of both. The rationale behind the MathML precprocessing it to discover important patterns from the equation and assigning meaning to them. Thereafter, by counting their respective number of occurrences, the final document vector can be constructed.

Equation	MathML Representation
$A = \sqrt{24} \times \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$	`<math><mrow>` `<mi>A</mi>` `<mo>=</mo>` `<msqrt><mrow><mn>24</mn></mrow></msqrt>` `<mo>&#x00d7;</mo>` `<mrow>` `<mo>[</mo>` `<mtable>` `<mtr>` `<mtd><mn>1</mn></mtd>` `<mtd><mn>2</mn></mtd>` `</mtr>` `<mtr>` `<mtd><mn>3</mn></mtd>` `<mtd><mn>4</mn></mtd>` `</mtr>` `</mtable>` `<mo>]</mo>` `</mrow>` `</mrow></math>`

Template Rules	Mapped Keyword
<mo>[\(\[]</mo>\s*(<mrow>)?\s*(<mtable>\s*(<mtr>(\s*<mtd>\s*\p{Graph}+\s*</mtd>){2,}\s*</mtr>){2,}\s*</mtable>)\s*(</mrow>)?\s*<mo>[\)\]]+</mo>	Matrix
<m(?:sqrt|root)>\s*(?:(<mrow>\s*)?<mn[^>]*>\d+</mn>\s*(</mrow>\s*)?)+</m(?:sqrt|root)>	Root

Once a pattern or symbol is represented in MathML representation, it is replaced by it's corresponding mapped keyword. The equation shown above will be identified as Root, Matrix i.e., the question vector formed will have value 1 for the keys root & Matrix and 0 otherwise.
A list of a few templates is given below

Template Rules	Mapped Keyword
<mrow><mo>\{</mo><mrow><mn>\d+</mn>(<mo>,</mo><mn>\d+</mn>)*</mrow><mo>\}</mo></mrow>	Set
<mrow><mo>\(</mo>(<mo>&#x2212;</mo>)?<mn>\d+</mn><mo>,</mo>(<mo>&#x2212;</mo>)?<mn>\d+</mn><mo>\)</mo></mrow>	Point
<msup>\s*<mrow>\s*(?:<mi[^>]*>\w+</mi>\s*)</mrow>\s*<mrow>\s*<mo[^>]*>(?:&minus|&#x2212;|&#8722;)</mo>\s*<mn[^>]*>1</mn>\s*</mrow>\s*</msup>	Inverse
<mo[^>]*>(&#x003e;|&#x003c;|&#x2265;|&#x2264;|&#x2260;)</mo>\s*(<mrow>\s*)?<mi[^>]*>\w+</mi>\s*(</mrow>\s*)?<mo[^>]*>(&#x003e;|&#x003c;|&#x2265;|&#x2264;|&#x2260;)</mo>	Interval
<msup>\s*(<mrow>\s*)?<mi[^>]*>\p{Alpha}+</mi>\s*(</mrow>\s*)?(<mrow>\s*)?(<mfrac>\s*(<mn[^>]*>\d+</mn>\s*){2}</mfrac>|<mn[^>]*>\d+</mn>)\s*(</mrow>\s*)?</msup>	Exponential

In the present implementation, we use regular expressions for template matching. The template based approach allows us to index math expressions with a standard text IR system. It is possible to plug in another approach (tree based matching, graphs) to the same function. Both the equations entered by the user and the equations saved in the database have to pass through the preprocessing stage. This results in the same intermediate representation for both.

Algorithm 2.1. Query Processor (QML)

Input:

M – {Equation(s) MathML}
D– {set of all math equation vectors}

Output:

Math Equation Vector *VEC* with keywords and normalized weights

Process:

```
1. Initialize the Question Vector Q
2. for all $d_i$ is in D do
3.     for all $KeyWord_i$ is in di do
4.         if isPresent ($KeyWord_i$ in Q) then goto step 3
5.             Add $KeyWord_i$ to VEC
6.         end if
7.     end for
8. end for
9. for all $KeyWord_i$ is in $VEC$ do
```

```
10.     if isPresent ($KeyWord_i$ in Query)
11.         VEC ($Keyword_i$, $Value$) ← Occurrence of $Keyword_i$ in Query
12.     end if
13. end for
14. Calculate normval as SQRT sumofSqr of all non null values in VEC
15. for $i$ = 1 to n, which is the max length of VEC do
16.     if not null VEC ($KeyWord_i$, $Value$) then
17.         Normalize by dividing value by normval
18.     end if
19. end for
20. return VEC
```

As discussed in section 2, we use MathML to represent the expression. When ranking retrieved problems based on their similarity to query, we need to evaluate two set of mathematical terms. The conventional text based ranking schemes e.g., the tf-idf measure are highly inadequate for math search. A large-size formula does not essentially means more relevant for the user. In the mathematical context, significance of a term is more important than its occurrence. Other important factors to consider are the co-occurrence of the term with other math terms and the place in the mathematical structure where it lies. The current generation of mostly experimental mathematical systems uses the same traditional tf-idf metric [15]. Alternative specialized metrics need to be utilized for math search.
A list of mapped regular expression → keywords is maintained. This has the effect of selecting only the valid input from the user.
Integral (sin x + cosx)
The above expression will match three regular expressions. The regular expressions will be assigned appropriate names.

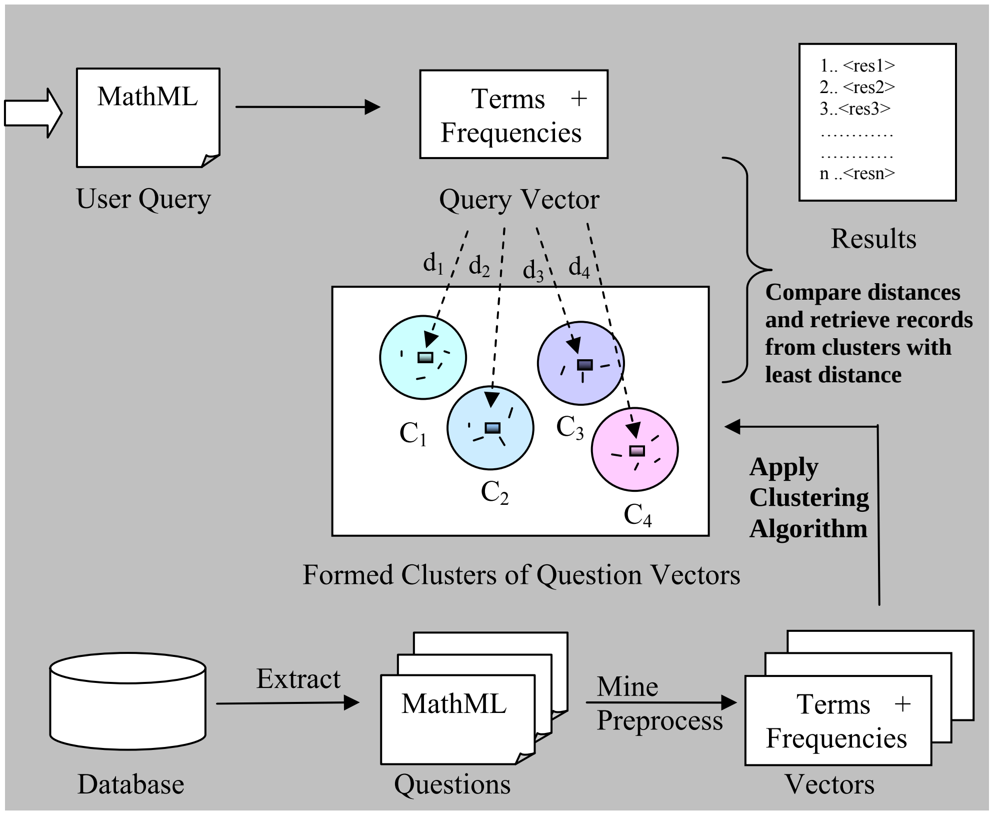
```
Algorithm 2.1. Content Retrieval (VEC, DVEC, THRESHOLD)
Input:
        VEC – Question Equation Vector
        DVEC ={$d_i$ | $1 \le i \le n$ } – set of all math equation vectors stored in the database
        THRESHOLD – limiting value
Output:
        CVEC – the set of closest (relevant) equation vectors
        RES – the final set of results.
Process:
1. for all $d_i$ in DVEC do
2.     Insert Sim ($d_i$, Vec) into CVEC
3. end for
4. for $i$ = 1 to n
5.     sort (CVEC)
6. end for
7. RES  ←  {Matched and sorted equation vectors (CVEC) above the predefined threshold}
8. return RES
```

Ranking: Relevance ranking in text search has been an active research field for many years. received much research attention over three decades. Although several relevance metrics have been developed and studied, most are elaborations and variations of one central metric, often referred to as the tf-idf metric (term frequency inverse document frequency) [15]. The tf part incorporates the relevant importance of a query keyword hit document. The idf part handles its global importance. According to this metric, query keyword occurring more frequently in a hit document makes it more relevant. Two documents have lengths *l1 and l2 with differing lengths d1 < d2. If num (terms) in d1 = num (terms) in d2 then tf-idf measure will rank d1* as more important than d2.

We employ the modified tf-idf metric for math content, the documents are ranked and displayed in the same way as their text counter parts. This is done by calculating the cosine similarity between the formed vectors of the mathematical equations.

With the modified tf-idf metric for math content, the documents are ranked and displayed in the same way as their text counter parts. We have experimented with changing values for the parameters for more importance e.g. assigning more weight to a function (sin) than to an operator (+). By modifying different weights different results can be obtained. This feature allows for retrieval based on categories. By modifying the weight of derivative, all differential equations relevant to the query vector will be ranked first.

Formula Clustering: In this section, three clustering techniques are presented which were used to improve the retrieval experience. We cluster the results and compute the similarity of the query vector with the centroids of all generated clusters. As a result, the most similar cluster is selected. Then the similarity of all the question vectors in the cluster are computed with the query vector. The most similar set of entries is selected and displayed as a result. We use three clustering techniques to cluster the questions. The cluster centroids are matched with the query vector and the closest clusters are displayed in the list.



System Architecture: In this section we present methods to address the issues presented above in our prototype search system Math GO!. Broadly, there are two ways for building math search systems. The first is a text based IR approach, where math aware functionality is added on top of a conventional search engine [14]. This approach is easier and much faster to carry out. The other approach is to use a radically different approach based on the emerging XML based technologies and markup languages (i.e., XML-Schema, RDF, OWL). These technologies are not fully developed and as such need sometime to get mature. We follow the first approach in our development of the MathGO! Search system. This is also the direction adopted by a related project of Digital Library of Mathematical Functions (DLMF) [14].

The search engine is based on the text IR system principles with techniques for making it math aware. The search engine uses a hybrid model where user can move to and fro between a browsing and querying model. This can be called as browsing plus search model. User can navigate to his required topic and can use the search facility provided. We give eqn ID to every equation. We store the equation ID in a separte table. Equations are identified with unique id's system wide. When the user inputs a new formula, an ID of 0 is assigned to it. Regular expression matching is used to extract the keywords from the newly entered formula. Then the Query Processor component forms the query vector for the entered formula. Then the query vector is transformed to uniforms size. Convert the word frequency values to qf-idf and then normalize the qf-idf values. QF-IDF is employed for better weighting. We have normalized vector, we retrieve all the question document vectors for all the equations. The equation id' s concept and other id's etc. Equations are represented with unique id's systemwide. Equations are stored in a table.

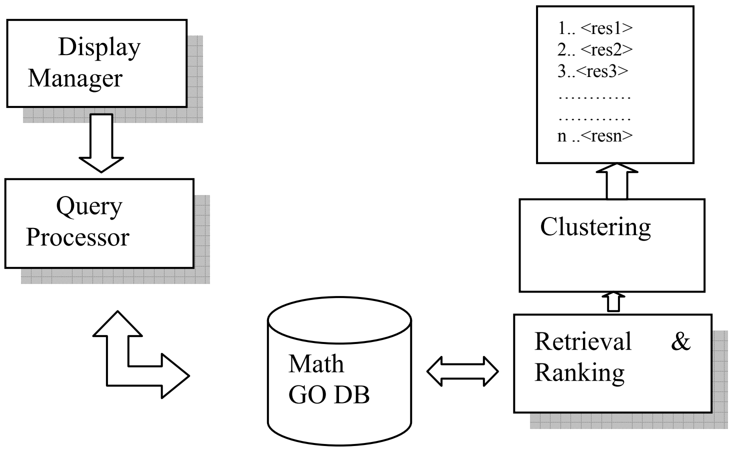


Fig 1: Math GO! High Level Architecture

There are several approaches for mathematical query input. These approaches range from LaTeX style input to free form input of math expresson. We use a hybrid model of search. We provide a hierarchy of mathematical topics. This is coupled with the search interface for the user to search for any formula content. We also provide the functionality to search within a sub topic. We use design science WebEQ editor[2] for query input which is specifically designed for working with equations encoded in MathML, the Web standard for representing mathematical expressions.

The content is grouped into meaningful categories. Thus we provide a hybrid search model. User can use the taxonomy of formulas to navigate to the required result or he can search for the formula. The taxonomy is based on the A' Level formulas categorization. We store the main data in form of relational database.

4. PERFORMANCE ANALYSIS

In this section, we present experimental results to evaluate the effectiveness of our approach. We have tested and evaluated the performance of the system using the standard metrics of precision and recall.

Performance analysis of a math search system is not an easy task due to the lack of standard math query benchmarks. This area of research has still not matured enough to have a TREC like infrastructure for evaluation of math retrieval methodologies. No standard mathematical collection exists which can be used for testing and evaluation purposes. Same is the case with the selection of subtopic specific math queries. The authors had to manually decide the relevancy of the questions. The notion of

[2] http://www.dessci.com/en/products/webeq/authoring/editor.htm

similarity between mathematical documents is extremely subjective. There is sure to be some disagreement among different researchers over what is relevant and what is not. (e.g., searching for sinx, the results containing cosx or tanx should be considered as relevant or not). For our part, we consider the results sharing semantic meaning as relevant. E.g., while searching for tanx, we classify the results containing secx cotx as relevant too. We evaluated the system with hundreds of queries. Table 1 below shows the results of a few queries and their corresponding precision and recall score's.

We experimented on a collection of about 500 mathematical documents containing about 1400 mathematical equations. The topics ranged from Integration, Differentiation, Complex no's, Matrices, Vectors, Trigonometry, Conic Sections etc. The search process was conducted by using four different methods. Simple similarity matching was utlilized to measure precision and recall of the system. Further experiements were conducted with the K-Means, AHC and K-Som algorithms.

Table 1.
Experimental results based on different mathematical domains.

	Similarity Match		**K-Means**		**AHC**		**K-SOM**	
	Prec (%)	**Rec (%)**	**Prec (%)**	**Rec (%)**	**Prec (%)**	**Rec (%)**	**Prec (%)**	**Rec (%)**
Trigonometric Functions	80	90	80	100	75	95	90	100
Integration	100	100	75	100	70	100	90	100
Complex Numbers	80	50	85	100	85	100	80	100
Vectors	85	100	100	100	100	100	100	100
Differential Calculus	75	100	70	100	85	100	90	100
Special Functions	85	100	80	100	100	100	80	100

The average calculated retrieval time was in the order of 2 to 3 seconds. The code is Java based and debugging/profiling information attached to it takes its toll. The query preprocessing part takes in the order of milliseconds while the retrieval and matching part takes most of the time (hundreds of milliseconds). The results are recorded on a P4 3.4 GhZ with 1GB RAM. The system uses a lot of memory due to it's java based setting. With minor changes, the system performance can be improved significantly. This is due to the fact that the data is presently organized in the form of relational database and knowledge base. The implementation language is java which is known to be relatively slow. It is argued that after transfer of data to a web based environment, the performance will improve significantly.

The initial usability study of our system revealed several advantages compared to others. [3][4] only allow input in form of keywords. [5] provides separate fields for text and formula. [6] does not has an option of formula based search.

5. CONCLUSIONS

This paper presents the MathGO! formula search engine. The paper has identified the issues involved in building a math search system and presented our approach to solve them. The experimental results have shown the capability of our approach. The performance has been evaluated using the precision and recall as means of quality measure. The system makes several contributions.

1. Use of template based approach to recognize mathematical expressions.
2. Clustering of mathematical formula content by using K-Som, K-Means and AHC for better retrieval.
3. Uses a hybrid search mode for better information access where user can either search the query or reach to it through browsing.

In future, it would be interesting to investigate the possibility of combining mathematical documents containing question text as well as equations. Our project only deals with common mathematical functions. We have designed the system in a modular way. To match expressions, some researchers have used tree matching, graph matching techniques. Our system allows for easy plugging in of the other methods of expression matching. The drawback in our approach is that as the no of templates will grow, the complexity will also increase unless some optimization measure (e.g., using small template list to match the question and resort to the bigger list only if required) is adopted. The present use of database makes the performance slower. The webeq editor cannot be customized for user requirements.

This is an ongoing research. We have the following goals in mind.

1. Our system deals with formulas. We are investigating techniques to extending the system to incorporate full query support (text content and formula content).
2. To investigate other formula matching techniques like graph matching, tree based matching, edit distance matching etc for better results.
3. Clustering the output to form semantically coherent clusters. We are currently working on the issue of clustering of mathematical documents. We intend to look into clustering of retrieved mathematical results at runtime into meaningful categories.

We are in the process of porting the system from a desktop application to a web based one.

6. ACKNOWLEDGEMENTS

We would like to acknowledge the sponsorship and support provided by Higher Education Commission, Pakistan, International Islamic University, Islamabad, Pakistan and School of Computer Engineering, Nanyang Technological University, Singapore for this research. We also thank the students who worked on the implementation issues. (TingFang, Adrian Loke and Sidath Harshanath Samarashinghe)

REFERENCES

[1]. The Google Search Engine, http://www.google.com

[2]. Microsoft Live Search, http://www.live.com/.

[3]. Dr. Math. Math Forum @ Drexel, http://mathforum.org/library/drmath/mathgrepform.html

[4]. MathSearch, http://www.maths.usyd.edu.au/MathSearch.html

[5]. MathDex, http:// www.mathdex.com/

[6]. The Digital Library of Mathematical Functions (DLMF), the National Institute of Standards and Technology, http://dlmf.nist.gov/

[7]. MathML 2.0, a W3C Recommendation, July, 2008, http://www.w3.org/Math/.

[8]. Robert Miner and Rajesh Munavalli, ―An Approach to Mathematical Search Through Query Formulation and Data Normalization, in Towards Mechanized Mathematical Assistants, SpringerLink, 2007, pp. 342-355.

[9]. *Michael Kohlhase and Ioan Sucan, ―A search engine for Mathematical Formulae, in Artificial Intelligence and Symbolic Computation, pp. 241-253, SpringerLink, 2006.*

[10]. Andrea Asperti, Ferruccio Guidi, Claudio Sacerdoti Coen, Enrico Tassi, and Stefano Zacchiroli, ―A Content Based Mathematical Search Engine: Whelp, in Types for Proofs and Programs, pp. 17-32, SpringerLink, 2006.

[11]. F. Guidi, Searching and Retrieving in Content Based Repositories of Formal Mathematical Knowledge, Ph.D. Thesis in Computer Science, University of Bologna, March 2003, Tech Report UBLCS 2003-06.

[12]. F. Guidi and I. Schena, “A Query Language for a Metadata Framework about Mathematical Resources.”the 2nd International Conf. Mathematial Knowledge Management, Bertinoro, Italy, Feb 2003.

[13]. T. Hardin, “Mathematical Knowledge Management in FOC[France]”, Schloss Scholls Hagenberg, Austria, Sep 24-26, 2001.

[14]. Abdou Youssef, "Information Search And Retrieval of Mathematical Contents: Issues And Methods," the ISCA 14th Int'l Conf. on Intelligent and Adaptive Systems and Software Engineering (IASSE-2005), July 20-22, 2005, Toronto, Canada.

[15]. Abdou Youssef, "Methods of Relevance Ranking and Hit-content Generation in Math Search", The 6th Mathematical Knowledge Management Conference, Hagenberg, Austria, June 27-30, 2007.