ISSN 2278-3091
Volume 9, No.1, January – February 2020
International Journal of Advanced Trends in Computer Science and Engineering
Available Online at http://www.warse.org/IJATCSE/static/pdf/file/ijatcse16912020.pdf
https://doi.org/10.30534/ijatcse/2020/16912020

Recognition and Retrieval of Mathematical Expressions from Arabic Documents

Emad Al-Shawakfa[1] , Mohammed Tawalbeh[2]
[1]Information Systems Department, Faculty of IT, Yarmouk University, Jordan, shawakfa@yu.edu.jo
[2]Information Systems Department, Faculty of IT, Yarmouk University, Jordan, mt@just.edu.jo

ABSTRACT

Recently, researchers have strived to enhance existing search tools to enable the retrieval of a diversity of data types to help users in finding what they are looking for from the Web, however, very limited research is applicable to Arabic content.
In this research, we introduce a rule-based approach to search for mathematical expressions written in Arabic and/or English from Arabic stored documents. A set of normalization as well as math expressions' equivalence rules were built to enhance the capabilities of a math search engine. Furthermore, rules for structural search to enable search for sub-expressions were also built. An indexing mechanism and a mapping between Arabic and English expressions during the search process was also produced. The approach was applied using a set of forty queries; written in Arabic and/or English, was applied on a manually collected dataset of 100 documents which has produced an overall accuracy of 75%.

Key words: Math Information Retrieval, Arabic Math Expression, Text-Based Search, Normalized Math Tree.

1. INTRODUCTION

Search engines were developed to help in locating data of various types, including text, images, audio, and video over the Internet. However, domain specific documents may contain special forms and characters such as: mathematical expressions (ME), drawings, charts, tables, and diagrams that are more structured and difficult to retrieve using traditional search engines. Furthermore, such types of data cannot be effectively indexed and/or queried via conventional search engines. As mentioned by [1], navigation by search engines are either static or dynamic.
According to [2], web content can be categorized into two major types: structured and unstructured. As defined in [3], structured information is information that is ordered in a particular way and combined with extra features to describe it and to construct the relationship between its contents; a information. On the other hand, unstructured information is defined as information that exists in random pieces which does not contain any hidden information other than text; a normal text-based information is a good example of unstructured information.
Many existing web sites and digital libraries include huge number of scientific papers and books in digital format that contain mathematical expressions and formulae; an essentiality to researchers of different scientific areas like: Mathematics, Physics, Biology, Chemistry, etc. [4]. As Arabic math content has grown extensively on the Internet lately, and with the need by Arab researchers and/or students to search for such content on the Internet, we feel the need to provide a mechanism to search for such content easily. Unfortunately, existing search engines do not provide suitable ways to support the search for such documents containing ME. Figure 1 gives a screenshot of an existing Arabic content from a physics topic on the web [5].

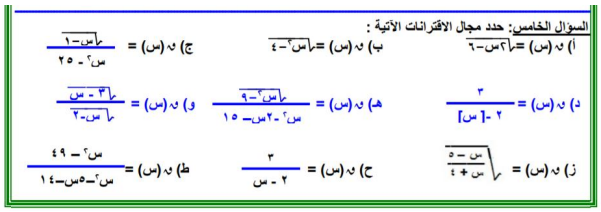


Figure 1: Arab math content on the web [5]

As noted by [6], a math expression; whether written in Arabic or English scriptures, could be categorized under structured information and are very formal since they contain special characters and symbols and contain both the structure and the semantic of the information that tell much about the expression itself; for which special attention and processing tools are needed.
With traditional text-based search engines, when searching for functions of symbols like **sin**, **cos**, **log**, and many others, the search will be for the best text matching of symbols only, and not their meaning. With the existence of many expressions with the same meaning presented in different ways, a math search engine must be able to retrieve all related expressions of the same meaning and with different structures than a given query, regardless of the language used; English

or Arabic. For instance, an expression like “a+b” is similar in meaning to that of “x+y” while an expression like (x-y) is mathematically different than (y-x). With traditional search engines a search is performed, and an exact content match is performed. So, the search for the “x-y” expression for instance, will be searching only for x, y, as well as the minus sign (-), regardless of their order hence, documents containing expressions ‘x-y’ and ‘y-x’ will be retrieved even though their meaning is different. A math search engine, on the other hand, must retrieve all documents with all related formulae of the same meaning and structure and simply not an exact content match. A math search engine should allow the search for sub-expressions.

Furthermore, traditional search engines are not equipped with interfaces to enable the search for structured information and thus, poor results are usually produced. Usually, it is not that easy to get free tools, or even freely available Math editors to support the input of math special characters and symbols and arrange them in an appropriate and suitable manner to be used by a search engine; a crucial need to enable users to write and search for different types of mathematical information and obtain good results.

When building a math search engine, two main things must be addressed in addition to the meaning of an expression: the representation of the mathematical expressions and the mechanisms used to describe such expressions. In order to properly search for mathematical expressions, they must be represented in an efficient way; both in the database as well as the query. There are many special purpose languages and tools in existence today that could be used to represent math expressions and their structures, like MathML [4], LATEX format [6], OpenMath format [7], and Math Objects as in Microsoft Word editor. Such formats describe ME and make them available and readable to users. In addition, some math styles and frames that are more consistent and less ambiguous are used to represent and describe mathematical expressions’ content and structure, like that in [9]. These descriptions are quite helpful in handling complex math notations and the support for non-keyboard symbols.

As noted by [10], to process mathematical expressions properly, we need to apply a math normalization process; a process that should convert math expressions into unique representation to enable the search for expressions represented in different structures. For instance, expressions like "a*x+a*x*y+a*x*z" and “a*x*(1+y+z)” have the same meaning, even though they have different structures. Applying a normalization process makes the matching process between expressions and equivalence detection process possible, and hence, the similarity score becomes measurable.

Another issue worth noting is that, the same math expression can be represented in different equivalent forms; which is similar to the usage of synonyms with text-based searches. However, with text-based searches, thesauri are used to resolve this problem. In math-based search, a thesaurus cannot be used to resolve equivalent math expressions because they are sometimes too many. For instance, the expressions (½) , (0.5) , (2^-1) , (2^{-1}) , (2/4), are all equivalents, also "(ظاس)" is equivalent to "(جا(س)÷جتا(س)" , and “tan(x)” is equivalent to ”sin(x)/cos(x)”, etc.

To better search for ME, an indexing process is needed. According to [6], the most general indexing processes use text-based mechanisms. Other techniques, however, do exist such as tree-based [11] or image-based [12]. Nowadays, the most widespread mechanism is the tree-based technique.

Due to the increase volume of Arabic content on the web, and the increased percentage of Arab users of the Internet, more and more requests to search for scientific content has risen. The main objective of this research is to help in searching for Arabic scientific content; documents with formulae written either in Arabic and/or English. To achieve this objective, a Rule-Based approach for the retrieval of math expressions from Arabic Documents was built. We used existing normalization and equivalence rules and developed new rules to help in normalizing different structures of math expressions. We represented structures as math-trees and then converted them into a unique indexable and searchable form.

2. RELATED WORK

Many math retrieval systems were built to deal with Math content which were mainly based on the usage of MathML or Latex formats in their representation. Such systems were based on a Structural approach, a Textual approach, or a Hybrid approach. Structural approach systems are concerned with how to express the meaning of a content based on the structure, like that of ([13]; [14]; or [15]), Textual approaches are concerned with addressing math content as textual information like the work of ([16]; [6]; [17]; and [18]). Hybrid approaches, on the other hand, combine both structural and textual approaches to search for math content like that of ([19]; [11]; and [4]. Other systems existed like the works of [8]; [20]; [21]; [6]; [22]; [23]; and [24].

Many enhanced systems were introduced into the literature like that of (10]; [25]; [26]; [3]; [27]); [28]; [29]; [30]; and [31]), as well as others. Techniques to search for mathematical expressions from image documents were also presented like that by ([32]; and [12]). A good survey on mathematical information retrieval using document-recognition technologies was provided in [33].

A standardization process to enhance mathematical information retrieval and a set of normalization rules were produced by [34]. In their research, the math-tree structure is built and then transformed into a unified normal form using math normalization rules. A research by [35] used a mapping process to transform ME from an original form into an equivalent one with the same representation. [36] presented a

new framework for mathematical information retrieval that helps in extracting the ME and then represent them as a Math tree and convert them into a normalization form using equivalence rules that were later expanded.

[37] used Data analysis techniques for math search, in which the Lattice-based approach was used for math search. The approach extracts features from MathML format to construct a mathematical concept lattice using some definitions, or rules where relations were represented as a term-document matrix.

As for Arabic Math Search Systems, the only work we came across was that of [38]. The author proposed a mathematical information retrieval system that enables search for and access mathematical information in Arabic. The indexing and retrieval of ME was based on the similarity of formula structures. In addition, the used indexed math expressions were based on the hierarchy structure (tree-based), where MathML format was used to represent the math expressions. Many attempts were made to introduce query languages suitable for the retrieval and understanding of math content. Such query languages were introduced in the works of [9], [39], [40], and [41].

3. RESEARCH METHODOLOGY

In our approach, we introduced a math style to represent math expressions in the repository and in an index file based on the plain text formats. Subsequently, we used the plain text format to build math expressions in a tree format representation; known as Math-Tree, and later on used it in the matching process. Moreover, we adopted and built equivalence detection rules to produce normalized-tree forms of math expressions. The approach we followed to complete this research is presented in Figure 2.

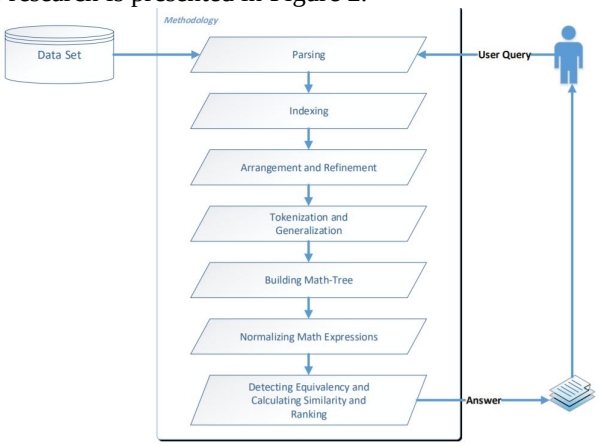


Figure 2: An Overview of the Research Methodology

3.1 Dataset Collection

For the purposes of this research, we could not find any standard dataset of documents that contain mathematical functions and/or expressions written in Arabic similar to that of (DLMF) or (EuDML) datasets mentioned in [29]. For this, we had to manually build our own dataset through collecting at least 100 documents from different web sites dedicated to online teaching. Our dataset contained different math functions written in Arabic and/or English scriptures. Furthermore, more documents were created via manually adding some math expressions with different symbols and structures. In our collected dataset, we assumed that the math expressions should exist in the plain text format and are represented in our proposed math style that was inspired from the Microsoft Word Math Editor for which, we manually reviewed and reformulated the dataset accordingly.

3.2 Research Phases

The research methodology is made up of several major phases: Parsing, Indexing, Arrangement and Refinement of Math Expressions, Tokenization and Generalization, building a Tree, Normalizing Math Expression, Equivalency Detection, Calculating Similarity and Ranking, and finally retrieving documents and/or paragraphs of relevant math expressions.

A. Parsing

In our approach, we dealt with three resources of mathematical information that need to be parsed to extract their semantic and structural meaning, two of them came from user queries, while the third one has come from our dataset. For user queries, in addition to the Math Expression Editor Light (MEEL) shown in Figure 3, we built our own math editor to enable the writing of Arabic math expressions. Furthermore, our adopted math style; represented as a set of linear strings, was used for both the Arabic-math editor and the dataset and helped in resolving the ambiguity of script characters. Non-keyboard symbols like the integral, trigonometric functions and other symbols were also supported and covered in our approach.

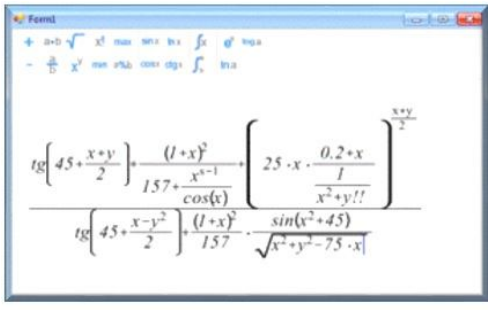


Figure 3: Math Expression Editor Light (MEEL)

Although MEEL can be used to represent and express user's queries in an intuitive manner, it could produce ambiguous keywords and notations with several restricted features, rules, or drawbacks. Furthermore, MEEL suffers from some

limitations like: the no support for Arabic right-to-left writing style, Functions and Operators are omitted and must be replaced by symbols or keywords, no support for Negation, no support for compact numbers (numbers with 2 or more digits), in addition to that complex operators and functions are represented as single isolated letters and not as a whole connected word.

The drawbacks in MEEL have motivated us to build our own Arabic Math-Query Editor (ABME) based on the adopted math style of the approach given in Table (1). ABME allows Arab naïve users to type and express their own queries in an accurate and simplified manner using Arabic terminology, ABME is given in Figure 4.

The parser preprocesses and extracts mathematical expressions written in Arabic and/or English from both the dataset and the user's query. This is done through recognizing different math symbols and keywords from different documents. If any math symbol is found, the mathematical expression is then traced from its start to end. In addition, the parser extracts all related data to each mathematical expression, such as identifying file names that contains the mathematical expressions, the page number, and paragraph number, which shortens time to reach the needed ME. The parsing process would be executed once for each document; unless modified or changed; based on their date and size, to enhance the searching process and reduce the time spent in the searching process.

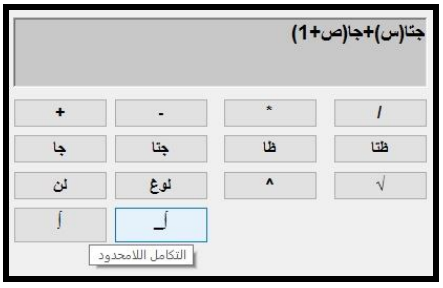


Figure 4: Arabic-Based Math-Query Editor (ABME)

Eventually, all mathematical expressions that have been read; whether from user queries or the dataset, are converted into a unified form to become easier to be addressed and implemented. In our approach, we built our standard structural plain text formats that are used to implement and compute mathematical expressions.

B. Indexing

Because of the huge amount of information and documents on the Internet, which makes it more difficult to search for ME, an indexing process is needed to enhance the searching process and reduce the time needed for parsing. To enhance the retrieval process, without an index, a comprehensive search through all documents must occur to extract all mathematical expressions, which means more time is spent and more machine resources are consumed. The usage of an index table enhances the search process and offers shorter time to retrieve relevant expressions.

Accordingly, we developed an indexing technique, for which, we built two index files: a file called IndexFileNames; that stores the names of documents containing the mathematical expressions that were parsed previously, and an IndexTable; that stores the extracted mathematical expressions. The main goal of the created IndexFileNames is to reduce the time spent in parsing documents. When the parsing process is initiated, all file names from the directory containing the dataset are read and stored. Furthermore, the IndexFileNames is invoked and the names will be retrieved from IndexFileNames for those files that are parsed and indexed previously. The output of this process is a list of file names which were not parsed previously, files with modified or changed date, or newly added files that need to be parsed. Taking into consideration the deletion or modification of files in our dataset, then the IndexFileNames must be frequently updated.

Table 1: Developed Math Expressions representation style

Operators & Functions	**English Form**	**Arabic Form**
Arithmetic operators	+ , - , * , /	+ , - , * , ÷
Exponentiation	^	^
Square Root	√(Exp)	√(التعبير الرياضي)
Infinite Integration	∫_(Exp)	∫_(التعبير الرياضي)
Integration	∫(Lower-Limit)→(Upper-Limit)_(Exp)	∫(الحد الاعلى) →(الحد الادنى)_(التعبير الرياضي)
Logarithm Function	Log(Base, Exp)	لوغ(الأساس, التعبير الرياضي)
Natural Logarithm Function	Ln(Exp)	لن(التعبير الرياضي)
Trigonometric Functions	sin(Exp) cos(Exp) tan(Exp) cot(Exp).	جا(التعبير الرياضي) جتا(التعبير الرياضي) ظا(التعبير الرياضي) ظتا(التعبير الرياضي)

The IndexTable file contains information about extracted math expressions, their location; file name, page number, and paragraph number, as shown in Table 2. The IndexTable is associated with the IndexFileNames file. When changes occur to IndexFileNames; deleting documents and/or modifying their internal content, the IndexTable requires an update as well. After building the index table, the search process will use the IndexTable file as a reference for any math expressions search; thus, reduces the time consumed in finding needed expressions.

To avoid overloading of index terms, we extract; at run time, sub-expressions from each expression in the IndexTable

through a tokenization process, so that both expressions and their sub-expressions are used as a reference for a user query.

Table 2:: Structure of the IndexTable

Expression	File name	Page number	Paragraph number
ا+ب+ص	word2	1	2
x^-2+√(a+b+c)+y*y*y+z	word5	3	2
ي-^ا	word14	1	3
:	:	:	:
((ص)^أ+س+جا	Word50	20	7

C. Arrangement and Refinement

This process produces a Layout representation of mathematical expressions written in Arabic and English. The layout representation is produced through multifaceted sub-processes like removing a space, removing duplicate parentheses, converting the math expression into a unified format in which parentheses are added based on mathematical operations priority rules that must be followed by the user, otherwise the semantic meaning of the math expressions would definitely change. The intelligence of this process is to extract and determine dummy variables in Integral functions as well. For example, the dummy variable in "∫(a)→(b)_(x^2+4x+1)" is dx and the expression will be converted into a simple unified format, hence, the representation will be as following: "∫((x^2+4x+1),x,b,a)". The layout representation describes both the content and the structure of the mathematical expression. Figure 5 gives the roduced layout representation for the expression "ا*ص+ب*ت+س^2".

Figure 5: Layout representation for "ا*ص+ب*ت+س^2"

D. Tokenization and Generalization

After producing the layout representations for both dataset and a user's query, we extract different sub-expressions using tokenization, where each sub-expression is called a token. For example, the sub-expression "(ا*ب)" is a token of "(ت+(ا*ب))". Tokenization is a straightforward process for obtaining sub-expressions from an expression and is a technique that improves the results, where the expressions are tokenized and subtrees are built to allow the searching for sub-expressions.

In our approach, tokens (or sub-expressions) are extracted through what is known as Math Expression Tokenizer [4], [18]. All tokens are stored in the location related to the original expression to avoid overloading of terms in the tree-based indexing method, and to help in searching for sub-expressions from the original ones, when needed.

Furthermore, we satisfy the hierarchal tokenization and order based on the different levels of the parse-tree. The top level takes more significance than the lower level during a searching process. Whenever necessary, we go deeper into lower levels to find appropriate relevant sub-expression. Figure 6 describes the searching strategy for the expression ((2^س)+((ب*ت)+(ا*ص))) and its sub-expressions given in Table (3) .

The weakness of the tokenization process is that extracting sub-expressions is mainly based on the priority and location of parentheses. For example, the sub-expression "((2^س)+(ب*ت))" of the original expression "((2^س)+((ب*ت)+(ا*ص)))" will be ignored due to the priority and usage of parenthesis.

In this research, we repurposed the working principle of the generalization process to increase the accuracy in finding needed math information. In our approach, we extract the structure of a user's query and retrieve all mathematical expressions that are generalized from such query based on the structure. Figure 7 describes a generalization process.

As shown in Figure 7, the lowest level of the parse-tree will be ignored, and the search process is concerned around the structure of the math expressions. For example, if a user types an expression like "((ع+غ)^(د-2))" the structural expression is represented as "((؟+؟)^(؟-؟))" and the math search must retrieve all math expressions that have exact, or similar structures, such as "((ع+غ)^(د-2))", and/or "((ص+س)^(ش-2))", and/or "((ب+ا)^(ث-2))".

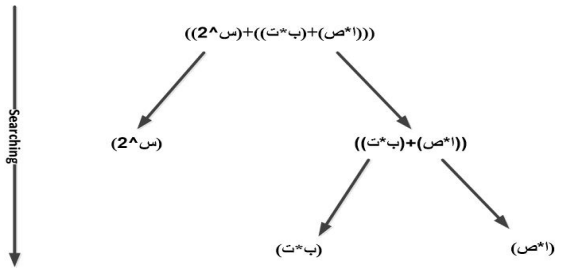


Figure 6: Searching strategy for ((2^س)+((ب*ت)+(ا*ص)))

Table 3: Sub-expressions for ((2^س)+((ب*ت)+(ا*ص)) using priority rules

Level	Tokens
(1)	((2^س)+((ب*ت)+(ا*ص)))
(2)	(2^س) ((ب*ت)+(ا*ص))
(3)	(ب*ت) (ا*ص)
Num# of sub-expressions=5	((2^س)+((ب*ت)+(ا*ص))) ، ((ب*ت)+(ا*ص)) ، (2^س) ، (ا*ص) ، (ب*ت)

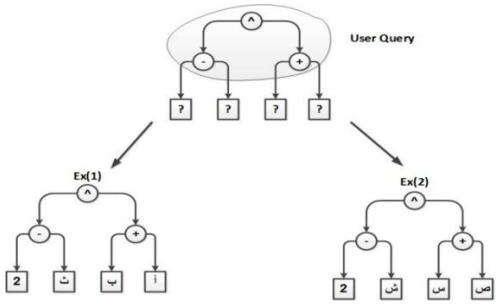


Figure 7: Generalization Process

Furthermore, the generalization process of our approach maps between equivalent trigonometric functions. For example, if a user types "(؟)ظا" then the math search must retrieve "(؟)ظا", and/or "(؟)جتا\(؟)جا", and/or "tan(?)" and/or "sin(?)/cos(?)". In addition, our approach performs mappings between Arabic and English complex functions (sin↔جا),(cos↔جتا),(tan↔ظا), (cot↔ظتا),(ln↔لن),(log ↔لوغ).

To increase the recall, we integrated the tokenization and generalization processes, and applied them to the original expressions as well as their sub-expressions. For instance, the tokens and the generalization terms of the expression "$\frac{\frac{س}{ص}(أ+ب)}{ت}$" are given in Table (4).

To resolve the ultimate number of equivalent constants, we built the numbers calculation rule. In this rule, constant children of each function, or operator, will be calculated and the function, or operator, will be replaced by the result. For instance, in the expression, "(ب^(4÷6))" the division operation (÷) is replaced by (1.5) and thus, the expression will be transformed into "(ب^1.5)".

Table 4: Tokens & generalization terms for $\frac{\frac{س}{ص}(أ+ب)}{ت}$

Level of Parse-Tree	Tokens	Generalization
1	$\frac{\frac{س}{ص}(أ+ب)}{ت}$	$\frac{\frac{*}{*}(*+*)}{*}$
2	$(أ+ب)$	$(*+*)$
2	$\frac{س}{ص}$	$\frac{*}{*}$

To simplify the representation of math expression items and eliminate parentheses, we have adopted the prefix notation like that in [19]; which has helped us in building a math-tree structure. For example, given the expression ((2^س)+((ب*ت)+(ا*ص))) the corresponding prefix notation is (ص,س,^,ت,ب,*,ص,ا,*,+,+). Figure 8 gives the prefix notation of the expression "2^(س+ص)".

Furthermore, this process applies two of the normalization rules; the Associative Property rule and the Grouping Property rule. The parentheses are only used to identify the priority and to extract sub-expressions from the whole expression.

^	power operator
+	plus operator
ص	variable ص
س	variable س
2	constant 2

Figure 8: Prefix notation of the expression "2^(س+ص)"

E. Building the Math-Tree

To convert the math expression from the layout representation to the semantic representation (tree-based), and after obtaining the prefix notation for each math expression, each mathematical expression can then be transformed into its basic components. The most important advantage of this process; tree representation, is to express the structure of a mathematical expression and provide extra features that could determine relevant expressions.

In building a Math-Tree in this research, in addition to simple math expressions, we covered the list of functions and keywords shown in Table (5); considered as parents, which have different number of arguments. On the other hand, constants, variables, and numbers, are considered leaf nodes, which are the lowest levels of a parse-tree. Initially, we have built a mixed algorithm (Top-Down and Bottom-Up) to build the Binary Tree, which later became more flexible, modifiable, and transferable (Figure 9). In building a tree, we took into consideration that the priority decreases from top to down and decreases from left to right at the same level.

In the binary tree, each parent have either one or two other children however, an Integration function has more than two arguments so, we added two children to the integration function, FromTo child; to connect the lower limit and the upper limit, and the Derivation child; to connect the Expression with its dummy variables, taking into account that the lower and upper limits may not only be constants or variables, but may be math expressions. In other words, we assign, at most two children for all parents to achieve the principle of the binary tree.

Table 5: Functions and Number of Arguments

Keywords	# Arguments
Square Root Natural Logarithm Function Trigonometric Functions Infinite Integration	One Argument
Arithmetic operators Exponentiation Logarithm Function	Two Arguments
Integration	Four Arguments

For example the structural math-tree for the expression "∫(t+1)→(t+2)_(x+1)" is represented in Figure 10(a) and the Tree-View representation for the expression ((∫(a)→(b)_(x+1)/x+y)) is given in Figure 10(b).

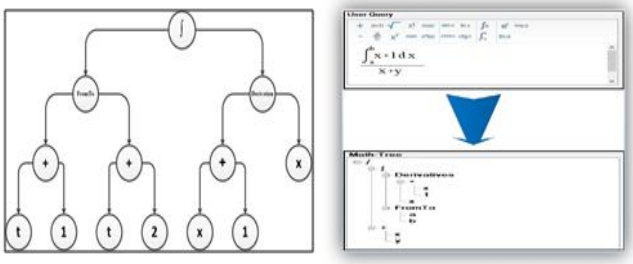
F. Normalizing Math Expressions

In this research, we used the concept "normalizing math expression" adapted by [10] to refer to the sequence of transformations needed to transform an original expression from its (algebraic/structural) format into a unified text format. So, different math expressions, with the same meaning, are converted into one uniform format, or transforming the expression-tree into a normalized-tree (semantic tree), to ensure a high recall of the relevant math expressions.

```
Input: prefix notations for expression
Output: Binary Tree for expression

Root-Next Equal First Item in list
Current Equal First Item in list
While   Current Not-Equal last item in list    Loop
{
     If Current is Parent Then
     {
        Current-LeftChild Equal Current-Next
        If Current is Parent And Current-Back isn't Parent Then
           {
             Temp Equal Current
             Current Equal Current-Back Until find LastParent

             If LastParent support two childs  And LastParent-
             RightChild Equal Null Then
                   {
                     LastParent-RightChild Equal Temp
                     Current Equal Temp
                   }
           }
     }
     Else If  Current isn't Parent And Current-Back isn't
      Parent Then
     {
          Temp Equal Current
          Current Equal Current-Back Until find LastParent
          If LastParent support two childs  And LastParent-
          RightChild Equal Null Then
               {
                 LastParent-RightChild Equal Temp
                 Current Equal Temp
               }
     }

  Current Equal Current-Next
}

Output: binary tree
```

Figure 9: Adopted Algorithm to build the binary tree

Most of the previous research produced and embraced normalization rules in their approach to make math search identical to text search in a concept in which mathematical expressions are converted into a normal form, to progress the matching process. In this research, in addition to covering and matching simple math expressions, we extracted and adopted the normalization rules used by [10]; and [8] and built new ones, to reduce the complexity of the math-tree in order to simplify the matching process (see Table (6)). Rules were built to search for the following types of expressions: Trigonometric Functions, Calculating numbers, Denominator negative power, Multiplication to power, Extraction of co-factors (variable), Power of one, and Multiplication of one.



(a) (b)

Figure 10: Math-tree (a) and Tree-View representation (b)

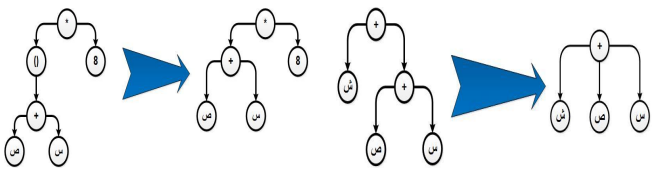
In addition, other rules were also developed like: Grouping Property rule; which deals with expressions according to priority (Figure 11 a), Tree Height Reduction rule; which combines similar parent nodes that descend form the same node with the lowest level parent node (Figure 11 b).

Table 6: Developed Normalization Rules

Rule	Original Expression	Equivalent Expression after applying rule
Associative Property	س + (ص + ز)	(س + ص) + ز
Commutative Property	ص * س	س * ص
Distributive Property	س * (ص + ع)	(س*ص) + (س * ع)
Calculating Numbers	س + (2*5) + ص	س + 10 + ص
Denominator Negative power	$س^{-2}$	2^س
Numerator Negative power	2-^س	1÷(2^س)
Power of one	س+(ص^1)	س+ص
Multiplication of one	س+(ص*1)	س+ص
Multiplication to power	ص+(س*س*س)	ص+(س^3)
Extraction of Co-Factors	س*ص+س*ص*ش	(س*ص)*(1+ش)

Furthermore, a Reorder Rule was built. This rule reorders the leaves of each parent node according to the priority rules according to the following: (-,+) < (*,/) < Power ^ < Numbers < Complex Operators and Functions < Alphabetic (Strings,Characters) (Figure 12).

Rules to search for Trigonometric functions, or identities, were also built. Such functions are classified into ten types [42]: Reciprocal identities, Pythagorean Identities, Quotient Identities, Co-Function Identities, Parity Identities, Sum & Difference Formulae, Double Angle Formulae, Power-Reducing/Half Angle Formulae, Sum-to-Product Formulae, and Product-to-Sum Formulae. Additionally, the trigonometric functions have special relationship between each other and have a lot of synonyms which make the similarity measure difficult. In this research, we have limited our scope to the following functions only, (جا , جتا , ظا ,ظتا ,sin , cos , tan , cot). Table (7) gives the Synonyms expressions for (tan / ظا) and (cot/ ظتا) based on the Reciprocal and Quotient identities.



(a) (b)

Figure 11: (a) Grouping Property, (b)Tree Height Reduction property

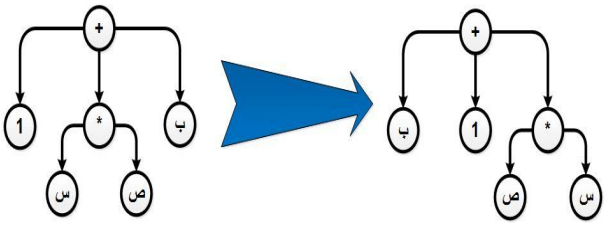


Figure 12: Reorder Rule property for the expression "1+(ص*س)+ب"

G. Detecting Equivalency and Calculating Similarity and Ranking

This process is responsible for detecting the equivalency of math expressions during the searching process and ranking the results based on calculating the similarity distances between the expressions and the user's query. Rather than the ON-OFF detection of equivalency followed in previous research, we built a detection mechanism; that we think effective, that looks for more details in detecting the equivalency, increase recall of relevant expressions, and enhance the accuracy of the retrieved expressions. This developed mechanism was based on developing and executing three equivalency detection algorithms in the following order: Exact Matching, Structure Matching, and Arabic-English Matching. If a match is found using one of the algorithms, the remaining algorithms are then skipped.

The Exact matching algorithm searches for two math trees; representing expressions and queries, that are of the same meaning and structure. The Structural matching, on the other hand, is a complementary process and an application of the generalization concept in which the matching is performed on math trees of similar structure. The Arabic-English Matching algorithm is performed to look for similarities between expressions that are of the same structure, but, written in two different languages: Arabic and English. To check for complex functions and operators, an Arabic/English (Ar↔En) mapping table was created for the transformation process like the matching of (sin↔جا).

In case of no match between a query and an expression is found, a search for sub-expressions is then performed. Moreover, we classified the searching process into three strategies: Query vs Expression, Query vs SubExpression, and SubQuery vs SubExpression. Figure 13 gives this strategy

Table 7: Synonyms expressions for (tan, cot, ظا , ظتا)

Normal Form	Synonyms expressions	
	Reciprocal identities	Quotient Identities
Tan(x)	$\frac{\sin(x)}{\cos(x)}$	$\frac{1}{cot(x)}$
Cot(x)	$\frac{\cos(x)}{\sin(x)}$	$\frac{1}{tan(x)}$
ظا(س)	$\frac{جا(س)}{جتا(س)}$	$\frac{1}{ظتا(س)}$
ظتا(س)	$\frac{جتا(س)}{جا(س)}$	$\frac{1}{ظا(س)}$

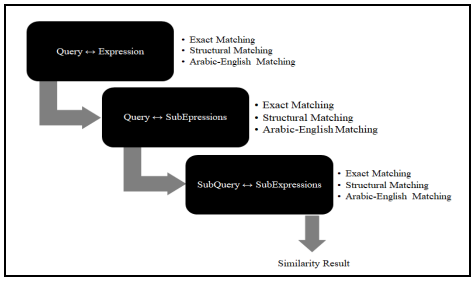


Figure 13: Searching Strategy

4. EXPERIMENTS AND EVALUATION

Due to the lack of related Arabic datasets, our experiments were carried out on a dataset of 100 documents (50 in Arabic, 50 in English). The dataset includes 1,479 mathematical expressions of different formats of functions and operators. Our approach was executed using 40 queries, we classified the queries into 10 queries for complex functions and operators, 10 queries for simple functions, while the rest of queries represents different forms. Tables (9a and 9b) give sample of Complex Queries (English and Arabic) and the covering rules.

To rank the retrieved expressions, we must measure the similarity distance between both a user query and the retrieved expressions. We could not find any suitable measures to be followed in determining the value of similarity between the Math-Trees. For this purpose, we built our own similarity distances between a user query and the retrieved expression using a weight value that was set based on the percentage of match type between similar expressions during a searching process. The values are assigned during the searching time, and accordingly, the location of similar expressions can be used to determine the similarity distance value between the location of the original non-tokenized expression and the location of the similar expression. The assigned similarity weight values in our approach are given in Table (8).

Table 8: Weights of Similarity Values

Searching Strategies	Type of Matching	Weight value
Query↔Expression	Exact matching	**100%**
Query↔SubExpression	Exact matching	**90%**
Query↔Expression	Structural matching	**80%**
Query↔SubExpression	Structural matching	**70%**
Query↔Expression	Arabic-English matching	**60%**
Query↔SubExpression	Arabic-English matching	**50%**
SubQuery↔SubExpression	Exact matching	**40%**
SubQuery↔SubExpression	Structural matching	**30%**
SubQuery↔SubExpression	Arabic-English matching	**20%**

To evaluate and interpret our results properly, we have adopted a modified set of equations for recall and accuracy [43] depending on the number of expressions. The main reason behind this refers to the fact that the matching is performed based on different matching types and ignoring irrelevant expressions, thus, the number of relevant expressions may be of no influence. Equations are as follows:

$$Recall = \frac{Number\ of\ Relevant\ Retrieved\ Expressions}{Total\ Number\ of\ Relevant\ Expressions}$$

$$Accuracy = \frac{Total\ weights\ for\ Retrieved\ Expressions}{Total\ Number\ of\ Retrieved\ Expressions}\ \%$$

Table (10) gives some Simple queries and their results, while Table (11) gives a sample of the complex queries with their results.

From Table (10), we can notice that some simple expressions' queries have the same accuracy values for both approaches. This can be referred to that either the Normalization rules are not applicable to such queries or that the normalization rules did not change their structure.

The Normalized Math-Tree search substituted the variance of weight by the quality of the retrieved expressions and not by the quantity, so the search for different expressions with the same meaning would be gives higher weights than search using traditional approaches which usually will not retrieve such expressions. For example, if a user types x^-1, the math tree search does not retrieve expressions like 1/x while the normalized math tree search will retrieve 1/x and gives it a weight of 100%. The number of retrieved expressions written in Arabic for an English query or vice versa is influenced by the number of applicable normalized rules and by the total number of retrieved expressions.

Table 9a: Complex English Queries and Covered Rules

Query	Covered rules
$a*b \mid a*c*d$	Extraction of co-factors.
$\int x*x*x+\frac{1}{y^{-2}}$	Denominator negative power Multiplication to power.
$\frac{1}{\cot(x+1)}$	Trigonometric Functions.
$\frac{\sin(x)}{\cos(x)}+\frac{1}{(x+1)^{-1}}$	Denominator negative power. Power of one. Multiplication of one. Trigonometric Functions.
$\int_{1+2}^{b} \ln(2*x+x)$	Calculating numbers.

Table 9b: Complex Arabic Queries and Covered Rules

Query	Covered rules
$\frac{1}{(ص+ض)^{-1}}$	Denominator negative power. Multiplication of one.
$\frac{(ب+1)جتا}{(ب+1)جا}$	Trigonometric Functions.
$((2+س)(ا)لوغ)^{-ع}$	Denominator negative power.
$\int \frac{1}{(ب*2+ب)^{-ع}}^{(2)}$	Denominator negative power. Multiplication of one.
$\sqrt{1+(2^{\wedge}س)جا}$	Crossed complex function

Table 10: Simple Queries and their results

Queries	Accuracy	
	Math-Tree	**Normalized Math Tree**
$2*ب \mid ب$	**38%**	**42%**
$س^2+ص^2$	**32%**	**34%**
$\frac{س+س^2}{أ}$	**32%**	**32%**
$\frac{أ^3*ب^3}{1-س}$	**30%**	**30%**
$\frac{50 \mid س}{7}$	**32%**	**33%**
s^2*h	**30%**	**50%**
$\frac{1}{2}*a*b*sin(c)$	**29%**	**30%**
$2*x+x$	**39%**	**42%**
$\frac{2}{3}*x-3$	**30%**	**31%**
x^{m+n}	**33%**	**34%**

As shown in Table (11), we can observe an enhancement in the recall and accuracy values after applying the normalization rules, especially, when applied to complex queries. We can also observe some queries with the same value of recall in both cases (0.7, 0.8, and 0.9) but have different accuracy values. This is due to the importance of applying the normalization rules and our adopted searching mechanism, some expressions are similar, but the Math-tree search retrieved them as subexpressions and not as a whole expression, which influenced the result and reduced the overall weight. The approach obtained an overall accuracy of 75%.

5. CONCLUSION AND FUTURE RESEARCH

In this research, we have developed a rules-based approach to search for mathematical expressions; especially Arabic math expressions from Arabic documents, with enhanced capabilities of math search of ME using normalization forms. In this research, the semantic meaning of math expressions was produced by transforming them into math-tree expressions. Techniques and algorithms were developed for detecting equivalency between different math-tree represented expressions. In addition, we explored some existing normalization rules and developed new ones to transfer both the dataset content and user queries into normal forms. The normal form is then used to compare a user query against the searchable index table. Using the normal form, the search process was found to retrieve the most similar expressions of a common meaning, but with different structures. This process showed very good and promising enhancements in retrieving complex expressions and slight enhancement for simple expressions.

The normalization process has enhanced the precision and recall in a way that could enhance a mathematical information retrieval system; especially, for complex math expressions. The normalization process works in a similar principle to the stemming process in text-based searches, where the normal form is used to improve the comparison and the matching during the searching process.

To enhance the search process, sub-expression search was performed through a tokenization process, if an expression is found to be irrelevant, we segment it into smaller pieces that are included in the search. Structural search is performed through a generalization process. Structures of math expressions were used to retrieve expressions with similar structures. Both related Arabic and English expressions of the same structure are retrieved through a mapping process between the two languages.

An effective graphical user interface that satisfies the visualization of results for the users was built which also helped in determining users' targets through enabling them to write their queries in an easier manner needed to reach intended results. Many challenges were faced in building our approach; where the major challenge for us was the lack of Arabic resources to support the approach. Some of these challenges and problems were resolved in our approach while others; like the writing of some Arabic Math Symbols have remained and determined unresolved.

A dataset of 100 documents (doc and rtf formats) that are rich in Mathematical Expressions; Simple and Complex, was built in this research. To test the approach, two sets of queries were used; one to test the approach in search for simple math queries, while the other was used to test for search of complex expressions. Forty (40) different queries were used to test the approach; 10 in Arabic, 10 in English, and 20 generic ones with mixed content. The overall obtained accuracy was 75%.

The limitation of our approach lies in the following; which constitute some possible directions for future work:

- Employing the approach to search for Arabic math content on the web.
- Expanding Normalization rules to cover more math content.
- Covering more complex expressions and functions for both Arabic and English contents.
- Dealing with different types of document formats; other than doc and rtf.
- Dealing with mathematical expressions represented in Image-based formats that requires proper segmentation of characters like the works of [44] and [45].
- Enhancing the built Arabic Math Editor and expanding the math style to represent more Arabic math content.

Table 11: Sample Complex Queries and their Results

Qry	Math-Tree Search			Normalized Math-Tree search		
	Accuracy	Recall	A↔E	Accuracy	Recall	Ar↔En
$a * b \mid a * c * d$	33%	0.5	105 (Ar) of total 247	40%	0.7	62 (Ar) of total 130
$\int x * x * x + \frac{1}{y^{-2}}$	42%	0.6	153 (Ar) of total 374	54%	0.8	74 (Ar) of total 146
$\frac{1}{\cot(x+1)}$	48%	0.7	58 (Ar) of total 133	60%	0.7	58 (Ar) of total 105
$\frac{\sin(x)}{\cot(x)} + \frac{1}{(x \mid 1)^{-1}}$	60%	0.6	79 (Ar) of total 198	64%	0.9	85 (Ar) of total 173
$\int_{1+2}^{b} \ln(2 * x + x)$	56%	0.9	162 (Ar) of total 374	63%	0.9	111 (Ar) of total 345
$\frac{1}{(ص + ض)^{-1}}$	70%	0.6	75 (En) of total 133	90%	0.9	68 (En) of total 168
$\frac{(ب+1)^{ب}}{(ب+1)جا}$	52%	0.8	75 (En) of total 134	70%	0.8	47 (En) of total 105
$\left((2+س).1\right)لوغ^{-2}$	70%	0.5	75 (En) of total 133	83%	0.6	47 (En) of total 105

$\int \frac{1}{(س*2+س)^{-3}}$	42%	0.9	142 (En) of total 247	90%	0.9	92 (En) of total 192
$\sqrt{1+(2^{س})^{2}}$	60%	0.7	90 (En) of total 160	64%	0.7	71 (En) of total 145

REFERENCES

1. Lakshmi, K. Pushpa Rani, and M. Purushotham Reddy. **A Comparative Study of Navigation Techniques and Information Retrieval Algorithms for Web Mining**. *International Journal of Advanced Trends in Computer Science and Engineering*, Vol. 8, no. 1.3, pp. 10-14, 2019. https://doi.org/10.30534/ijatcse/2019/0281.32019
2. G. Weglarz,. Two worlds of data - **Unstructured and structured**. *Information Management*, Vol. 14, no. 9, p. 19, (2004)
3. M. Shatnawi and Q. Abuein. **A Digital Ecosystem-based Framework for Math Search Systems**. *International Journal of Advanced Computer Science and Applications (IJACSA)*, Vol. 3, no. 3, pp. 78-83. (2012).
4. L. Gao, Y. Wang, L. Hao, and Z. Tang. **ICST Math Retrieval System for NTCIR-11 Math-2 Task**, in *Proc. of the 11th NTCIR Conference, Tokyo, Japan ,9-12.* (2014)
5. Awa2el Website Math review sheets. Available at https://www.awa2el.net/ accessed January 15th, 2020.
6. A. Adeel, H. Cheung, and S. Khiyal. **Math GO! Prototype of A Content Based Mathematical Formula Search Engine**. *Journal of Theoretical and Applied Information Technology*, Vol. 4, no. 10, pp. 1002–1012. (2008).
7. P. Kumar, A. Agarwal, and C. Bhagvati.. **A Structure Based Approach for Mathematical Expression Retrieval**. in *Proc. of the International Workshop on Multi-Disciplinary Trends in Artificial Intelligence 2012, (23-34). Springer Berlin Heidelberg.*
8. P. Libbrecht and E. Melis. **Methods to Access and Retrieve Mathematical Content in ActiveMath**. In *International Congress on Mathematical Software 2006, (331-342). Springer Berlin Heidelberg.* https://doi.org/10.1007/11832225_33
9. M. Altamimi, and A. Youssef. **A Math Query Language with an Expanded Set of Wildcards.** *Mathematics in Computer Science*, Vol. 2, no. 2, pp. 05-231. (2008).
10. M. Shatnawi and A. Youssef. **Equivalence detection using parse-tree normalization for math search.** In *Proc. of the 2nd IEEE International Conference on Digital Information Management 2007, Vol.2, 643–648.*
11. S. Kamali and F. Tompa. **Improving mathematics retrieval.** *Towards a Digital Mathematics Library. Grand Bend, Ontario, Canada, 2009, (37-48).*
12. R. Zanibbi and B. Yuan. **Keyword and image-based retrieval for mathematical expressions.** In *Proc. of Document Recognition and Retrieval XVIII, 2011, Vol. 7874, p. 787401.* https://doi.org/10.1117/12.873312
13. S. Kamali and F. Tompa. **Structural Similarity Search for Mathematics Retrieval.** In *Proc.of the International Conference on Intelligent Computer Mathematics 2013, (pp. 246-262). Springer Berlin Heidelberg.*
14. Y. Qin, H. Karimi, A. Zhang, and Q. Leng. **A Novel Mathematical Formula for Retrieval Algorithm, Mathematical Problems in Engineering,** Vol. 2014, Article ID 859157, (5 pages). 2014.
15. W. Zhong. **A Novel Similarity-Search Method for Mathematical Content in LaTeX Markup and Its Implementation**. M.S. Thesis, http://tkhost.github.io/opmes/thesis-ref.pdf., Accessed 9 Oct 2016.
16. S. Kim, S. Yang, and Y. Ko. **Mathematical Equation Retrieval Using Plain Words as a Query.** In *Proc. of the 21st ACM international conference on Information and knowledge management. 2012, 2407-2410.* https://doi.org/10.1145/2396761.2398653
17. J. Zhao, M. Kan, and Y. Theng. **Math Information Retrieval: User Requirements and Prototype Implementation.** In *Proc. of the 8th ACM/IEEE-CS Joint Conference on Digital libraries, 2008, 187–196.*
18. P. Sojka and M. Líška. **Indexing and Searching Mathematics**. In *Proc. of the International Conference on Intelligent Computer Mathematics, 2011, 228-243.*
19. J. Mišutka and L. Galamboš. **Extending Full Text Search Engine for Mathematical Content**. *Towards Digital Mathematics Library*. Birmingham, United Kingdom, July 2008, pp. 55-67.
20. M. Kohlhase and I. Sucan. A Search Engine for Mathematical Formulae. In *Proc. of the International Conference on Artificial Intelligence and Symbolic Computation, 2006, 241-253. Springer Berlin Heidelberg.* https://doi.org/10.1007/11856290_21
21. T. Schellenberg, B. Yuan, and R. Zanibbi. **Layout-Based Substitution Tree Indexing and. Retrieval for Mathematical Expressions**. In *Document Recognition and Retrieval XIX, Vol. 8297, P.N. 82970I. 2012, International Society for Optics and Photonics.*
22. G. Kristianto G. Topic, F. Ho, and A. Aizawa. **The MCAT Math Retrieval System for NTCIR-11 Math Track,** In *Proc. of the 11th NTCIR Conference, Tokyo, Japan, 2014, pp. 120-126.*
23. M. Líška, P. Sojka, and M. Růžička. **Similarity Search for Mathematics,** http://www.fi.muni.cz/usr/sojka/posters/ruzicka-sojka-liska-ntcir2014.pdf, Accessed 9 Oct 2016.
24. M. Schubotz, A. Youssef, V. Markl, and H. Cohl, **Challenges of Mathematical Information Retrieval in**

the Ntcir-11 Math Wikipedia Task. In *Proc. of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2015, pp. 951-954.*
https://doi.org/10.1145/2766462.2767787

25. T. Watanabe and Y. Miyazaki. **Development of IR Tool for Tree-Structured MathML- based Mathematical Descriptions.** In *Proc. of the International Conference on Computers in Education (ICCE2010), 2010, pp. 310-312.*
26. K. Ma, S. Hui, and K. Chang. **Feature Extraction and Clustering-Based Retrieval for Mathematical Formulas.** In *Proc. of the 2nd International Conference on Software Engineering and Data Mining (SEDM), IEEE, 2010, pp. 372-377.*
27. J. Mišutka, **Scaling Feature Based Mathematical Search Engine for Real-World Document Sets.** http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.417.3031 , 2013, Accessed 4 Jul 2016.
28. D. Stalnaker and R. Zanibbi. **Math Expression Retrieval Using an Inverted Index Over Symbol Pairs.** In: *Proc. of the Document Recognition and Retrieval XXII., 2015, Vol. 9402, p 940207.*
29. M. Líška, P. Sojka, and M. Růžička. **Combining Text and Formula Queries in Math Information Retrieval: Evaluation of Query Results Merging Strategies.** In *Proc. of the First International Workshop on Novel Web Search Interfaces and Systems, 2015, pp. 7-9. ACM.*
30. P. Pakray and P. Sojka. **An Architecture for Scientific Document Retrieval Using Textual and Math Entailment Modules**. In *Proc. of RASLAN 2014 8th Workshop on Recent Advances in Slavonic Natural Language Processing, Karlova Studánka, Czech Republic, 2014, pp. 107-117.*
31. M. Schubotz, A. Grigorev, M. Leich, H. Cohl, N. Meuschke, B. Gipp, and V. Markl, **Semantification of Identifiers in Mathematics for Better Math Information Retrieval.** In *Proc. of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval, 2016, pp. 135-144.*
32. L. Yu, **Image-Based Math Retrieval Using Handwritten Queries**, M.S. Thesis, Rochester Institute of Technology, 2010.
33. R. Zanibbi and D. Blostein. **Recognition and Retrieval of Mathematical Expressions**. *International Journal on Document Analysis and Recognition (IJDAR),* Vol. 15, no. 4, 2012, 331-357.
https://doi.org/10.1007/s10032-011-0174-4
34. A. Youssef and M. Shatnawi. **Math Search with Equivalence Detection Using Parse- Tree Normalization**. In *the 4th international conference on computer science and information technology, 2006.*
35. Q. Abuein and M. Shatnawi. **Expanded Grammar for Detecting (GER) Equivalence in Math Expressions.** *International Journal of Computer Applications,* Vol. 43, no. 15, pp. 44-51, 2012.
36. R. Batyha.. **Building a new framework for Mathematical Expression,** PhD dissertation, Department of information technology and computer science, The Arab Academy for Banking and Financial Sciences, Jordan, 2012.
37. T. Nguyen, S. Hui, and K. Chang. **A Lattice-Based Approach For Mathematical Search Using Formal Concept Analysis,** *Expert Systems with Applications: An International Journal*, Vol. 39, no. 5, pp. 5820-5828, 2012.
38. A. Al-Zubi. **Normalizing Different Representations of an Arabic Math Expression Based on a Context Free Grammar (CFG): Toward an Intelligent MathSearch Engine,** M.S. thesis, Department of information technology and computer science, Jordan University of Science and Technology, Irbid – Jordan, 2011.
39. C. Sasarak, K. Hart, R. Pospesel, D. Stalnaker, L. Hu, R. Livolsi, S. Zhu, and R. Zanibbi. **M-in: A Multimodal Web Interface For Math Search**. In *Symposium on Human-Computer Interaction and Information Retrieval (HCIR), Cambridge, MA. 2012.*
40. I. Abu Doush and S. Al-Bdarneh, **Automatic Semantic Generation and Arabic Translation of Mathematical Expressions on the Web.** *International Journal of Web-Based Learning and Teaching Technologies (IJWLTT),* Vol. 8, no. 1, pp. 1-16, 2013.
https://doi.org/10.4018/jwltt.2013010101
41. S. Yang and Y. Ko. **Mathematical Formula Search using Natural Language Queries.** *Advances in Electrical and Computer Engineering*, Vol. 14, no. 4, pp. 99-104. 2014.
42. Trigonometric Identities. http://www.sosmath.com/trig/Trig5/trig5/trig5.html Accessed 8 Dec 2017.
43. R. Baeza-Yates and B. Ribeiro-Neto. **Modern Information Retrieval**, Addison- Wesley Longman Publishing Co., Inc., Boston, MA, 1999.
44. L. Bany Melhem et al. **Frame Removal For Mushaf Al-Quran Using Irregular Binary Region.** *International Journal of Advanced Trends in Computer Science and Engineering*, Vol. 8, no.1.3, pp. 109-114.
https://doi.org/10.30534/ijatcse/2019/2181.32019
45. M. Abdullah, A. Agal, M Alharthi, and M. Alrashidi. **Arabic Handwriting Recognition Model based on Neural Network Approach.** *International Journal of Advanced Trends in Computer Science and Engineering*, Vol. 8, no.1.1, pp. 253-258.