WikiMirs: A Mathematical Information Retrieval System for Wikipedia

Xuan Hu, Liangcai Gao[*], Xiaoyan Lin, Zhi Tang
Institute of Computer Science & Technology, Peking University, Beijing, China
{xuan.hu, glc, linxiaoyan, tangzhi}@pku.edu.cn

Xiaofan Lin
A9.com
Palo Alto, CA, USA
xiaofanl@a9.com

Josef B. Baker
School of Computer Science, University of Birmingham
Birmingham, UK
J.Baker@cs.bham.ac.uk

ABSTRACT

Mathematical formulae in structural formats such as MathML and LaTeX are becoming increasingly available. Moreover, repositories and websites, including ArXiv and Wikipedia, and growing numbers of digital libraries use these structural formats to present mathematical formulae. This presents an important new and challenging area of research, namely Mathematical Information Retrieval (MIR). In this paper, we propose WikiMirs, a tool to facilitate mathematical formula retrieval in Wikipedia. WikiMirs is aimed at searching for similar mathematical formulae based upon both textual and spatial similarities, using a new indexing and matching model developed for layout structures. A hierarchical generalization technique is proposed to generate sub-trees from presentation trees of mathematical formulae, and similarity is calculated based upon matching at different levels of these trees. Experimental results show that WikiMirs can efficiently support sub-structure matching and similarity matching of mathematical formulae. Moreover, WikiMirs obtains both higher accuracy and better ranked results over Wikipedia in comparison to Wikipedia Search and Egomath. We conclude that WikiMirs provides a new, alternative, and hopefully better service for users to search mathematical expressions within Wikipedia.

Categories and Subject Descriptors

H.3.3 [**Information Systems**]: Information Search and Retrieval

Keywords

Mathematical Information Retrieval, Mathematical Resource Management, Structure Matching

[*]Liangcai Gao is the corresponding author.

1. INTRODUCTION

Mathematical formulae are commonly used to present essential ideas and concepts formally in many areas, including mathematics, physics and computer science. Nowadays, due to the increased compatibility of web-browsers in rendering formats like MathML and LaTeX along with improved tools for creating them in documents, more and more mathematical resources are produced, presented and distributed in structural formats. As a result, a growing number of these resources are available in information systems including mathematical knowledge management systems, academic digital libraries and digital education systems. In order to make these resources searchable and accessible, Mathematical Information Retrieval (MIR) is required for various reasons: *1)* In mathematical knowledge management systems (e.g., ArXiv, Wolfram), tremendous amount of work on classifying math contents has to be done manually to make resources searchable. Due to the current explosive growth in the area, manual classification is no longer suitable and MIR techniques are needed to achieve effective searching; *2)* In academic digital libraries, math expressions are commonly used to present the essential ideas of research work and a math-aware search engine would provide researchers a brand-new way of finding related works based on the most valuable components of publications; *3)* A growing number of education resources (books, lecture notes and exercises, etc) are being digitalised. Since students initially have limited background knowledge, it is difficult for them to find related mathematical contents in their learning materials with existing search engines. MIR techniques would be beneficial for them to find the needed information effectively via retrieving the math contents directly.

Because mathematical formulae are highly structured and domain specific, conventional search engines, which mainly focus on plain text retrieval, can hardly meet the needs. Although MIR has been discussed for nearly a decade [19], at present there exist few search engines that can provide satisfactory public information retrieval service. The challenges in constructing a MIR system can be mainly attributed to normalization, indexing, matching, query interface, and relevance feedback [19]. The performance of a MIR system is primarily determined by the first three challenges, which will be discussed in detail as follows.

Normalization: In MIR system, the purpose of normalization is to find the equivalent mathematical formulae with different presentations. Different from word stemming and thesaurus operation in text retrieval techniques, normalization should overcome variation in the spatial layout or semantic presentation of the same formula. Since the equivalent transformations of a formula are very flexible in both spatial layout and semantic presentation, normalization becomes rather complicated. For instance, formulae can have the same structure but different variable or constant names (e.g., $a + b$ vs. $x + y$). Formulae can also have equivalent semantic transformation (e.g., $k * (x + y)$ vs. $k * x + k * y$). Objects of normalization can range from identities of variables or constants to semantic calculation equivalence (e.g., order of operands).

Indexing and matching: Since mathematical formulae are naturally expressed in a tree-structure, matches of formulae are supposed to support matching in both text contents and structures. In order to calculate the structural similarities of formulae, attributes of tree structures are commonly considered, such as sub-trees and levels. For example, formulae containing the same main structure with different sub-trees (e.g., $a + b \times y$ vs. $x + y$), and formulae containing different main structure with the same sub-trees (e.g., $a + b \times y$ vs. $b \times y$). Therefore, to support such matching, the structural attributes of formulae need to be extracted and indexed. Indexing all sub-expressions keeps all the structural information of a formula, but the corresponding index space increases explosively. Therefore, it becomes difficult to strike a balance between having sufficient indexing and keeping a reasonably sized index space. Moreover, how to calculate the similarity score according to the attributes of structures remains an open problem, because the motivations of different users of the MIR systems vary according to their background and specific tasks [21].

In this paper, WikiMirs is proposed to facilitate mathematical formula retrieval in Wikipedia. And the system is publicly available[1]. In order to support mathematical formula retrieval considering similarities of both text contents and spatial structures, we propose a presentation tree parser to parse formula structures from a layout presentation markup, such as LaTeX. Moreover, a hierarchical generalization technique is proposed to generate fine-grained sub-trees (index terms) from the presentation tree. Furthermore, the similarity score of formulae is calculated based on both text matching and structure matching at different levels.

It is worth noting that, by now WikiMirs is unable to retrieve formulae which are semantically equivalent to the queried formulae (e.g., $\sqrt{x}$ and $x^{\frac{1}{2}}$), due to the following reasons: Firstly, the conversion of semantic equivalent math expressions requires high level domain specific knowledge, which are difficult to be processed by machine for the moment. The semantics of math expressions themselves can be quite ambiguous. Even people cannot understand them directly without sufficient context (e.g., "ab" can be understood as a variable "ab" or an expression, "a" multiplies "b"). Therefore, we intend to utilise the presentation layout of formulae rather than semantics in MIR in this paper. Secondly, a large amount of layout presentations can be converted into semantic presentations directly, e.g., displayed fraction expressions. In this sense, we believe that the layout presentation of formulae can greatly facilitate MIR. As a result, WikiMirs can provide better MIR service than the traditional search engines or text-based MIR systems through making full use of both the text and structure information in the layout presentations of formulae.

[1]`http://www.icst.pku.edu.cn/cpdp/wikimirs/`

2. RELATED WORK

MIR has attracted much attention from researchers in the past decade, and several related systems or methods on this task have been reported [21]. We compare the existing MIR systems in Table 1 and illustrate the related research on the following three basic aspects of MIR, namely formula presentation, indexing and ranking.

2.1 Formula Presentation

The most well-known markup formats for mathematical formulae are MathML[8], LaTeX and OpenMath[7]. MathML provides two representation forms for formulae, namely Presentation and Content MathML. Presentation MathML is designed to mark up the spatial layout of formula in order to present math contents accurately on screen, whereas Content MathML intends to mark up the semantic meanings of formula. Similarly, OpenMath is proposed to represent mathematical objects with their semantics. LaTeX is designed to be used by authors to create documents with high presentation quality and similarly to Presentation MathML, only the geometric layouts of formulae are coded. Obviously, Content MathML and OpenMath are the ideal representations to support semantic MIR because of the embedded semantics. However, existing mathematical resources are often written in Presentation MathML or LaTeX due to the ease of authoring and high quality rendering when used with display engines such as MathJax [6]. Moreover, LaTeX has been widely used by authors and editors since 1980s. Therefore, although mathematical retrieval techniques using Content MathML or OpenMath [3] can better support semantic retrieval, it is difficult to apply them to existing repositories of mathematical information. In other words, in order to design a practical MIR system, the focus must be on the intrinsic similarity of formulae using their layouts, which has not been well studied by now.

2.2 Indexing

According to the index model, the existing techniques can be generally classified into two categories, namely text-based or tree-based, as classified in Table 1.

The main idea of text-based MIR technique [1, 12, 13, 2, 14, 5] is to convert mathematical formula markups into plain text strings and then tokenize the strings linearly into terms, so that they can be indexed and ranked using existing information retrieval tools like Lucene. Because mathematical formulae are highly symbolic and structured, the transformation from structural formulae into plain text strings mainly focuses on how to encode structures in text strings and normalize different presentations of formulae. Miller et al. convert all non-alphanumeric symbols in LaTeX into alphanumeric symbols and normalize the order of operands into a canonical form [1, 12]. A similar method is proposed by Misutka et al. [2, 14] with improvement via normalization of variables and constants. In these methods, structures of formulae are lost. Aimed at matching formula

Table 1: Comparison of Mathematical Information Retrieval Systems

Systems	Presentation	Normalization	Matching	Ranking	Corpus	Available
Text-based indexing						
DLMF[1, 12]	LaTeX	Order[†]	Exact	*tf-idf*	DLMF	Y
Mathdex[13]	Presentation MathML	Variables	Similar	*tf-idf* with weights	arXiv; Wikipedia	N
EgoMath[2, 14]	Presentation MathML	Variables and constants	Exact	*tf-idf*	Wikipedia	Y
LaTeXSearch[5]	LaTeX	Unknown	Similar	Edit distance	Springer	Y
Tree-based indexing						
MathWebSearch[3, 11]	Content MathML	/	Similar	/	Connexions; Wolfram	Y
MIaS[4, 18]	Presentation MathML	Order; variables and constants	Similar	*tf-idf* with weights	arXMLiv	Y
[17]	LaTeX	Order and variables	Similar	Similarity of matched sets	arXiv	N
Other						
[15]	Content MathML	Variables	Similar	Similarity of feature sets	Constructed dataset	N
[16]	LaTeX	/	Similar	*tf-idf* with learned weights	Math Overflow	N

[†] "Order" denotes the order of operands.

structures in the text-based retrieval model, Miner et al. [13] index n-gram terms which are sub-structures with no more than n successive tags. However, n is only set to five in their system, meaning only sub-structures with a few nodes are indexed. Therefore, high level structure matching of expressions with complex structures cannot be considered. LaTeXSearch [5] is a related commercial text-based system, but its scheme is not reported publicly.

In tree-based methods, attributes of formula tree structures (e.g., sub-expressions [4, 18] or paths [10]) are extracted as index terms. Sojka et al. [4, 18] index all sub-structures of formulae. This method can guarantee high recall rate, but it may cause explosive growth of index space in real-world applications. To solve this problem, Kohlhase et al. [3, 11] apply a substitution tree indexing technique to index sub-structures of semantic formula presentation. A substitution tree represents the structure of all the indexed first-order logic terms. Exact or similar matches can be found by backtracking all nodes of the substitution tree using different strategies. Similarly, Schellenberg et al. [17] employ the substitution tree indexing technique to index layout presentation of formulae.

Besides text-based and tree-based methods, there are some other indexing approaches. Asperti et al. [9] present a meta-data based method, in which math notations are presented by a logic-independent meta-data model, so as to serve a large database. Based on Formal Concept Analysis, Nguyen et al. [15] extract math features for each formula and index them via constructing a mathematical concept lattice of these features.

2.3 Ranking

Ranking is always highly related to indexing because the similarity score calculation has to depend on the indexed contents. Most text-based methods use *tf-idf* to calculate similarities of formulae, which usually only consider the number of terms. Miner et al. [13] introduce weights for terms according to their levels, lengths and complexities. However, as only limited structure information of formulae is indexed in text-based models, the structure matching score is difficult to calculate properly. Some tree-based methods also use the modified *tf-idf* to calculate the matching scores of sub-structures. Using *tf-idf*, Sojka et al. [4, 18] introduce weights to discriminate sub-structure matches in different levels based on the assumption that structures in higher level are more important than those at lower levels.

Besides *tf-idf*, some methods evaluate the similarities of formulae according to the similarity of feature sets. Nguyen et al. [15] index formulae in the mathematical concept lattice structure based on similarities of feature sets of each formula. To search for a formula, the query is inserted into the mathematical concept lattice and similar formulae are ranked according to their distances from the query. In the literature [17], each formula can be presented by a set of sub-expressions with their attributes (e.g., contents, neighbours, etc) presented in 5-tuples. Retrieved formulae are ranked according to the set similarity between formulae. Zanibbi et al. [20] employ content-based image retrieval technique to search formulae in document images. Since this method calculates the similarity score based on visual features of formulae, its performance is sensitive to typesetting styles of formulae.

3. OVERVIEW

The workflow of the proposed system, WikiMirs, is illustrated in Figure 1, which includes four main modules, namely preprocessor, tokenizer, indexer and ranker. The index is constructed offline as described by solid lines in Figure 1. Firstly, the off-line Wikipedia corpus is downloaded as the source dataset of our system, then internal LaTeX markups of mathematical formulae are extracted by the preprocessor. Next, indexing terms are extracted by the tokenizer, where formulae representations are first normalized to support equivalent formula search; then, sub-structures, as well as fuzzy sub-substructures, are

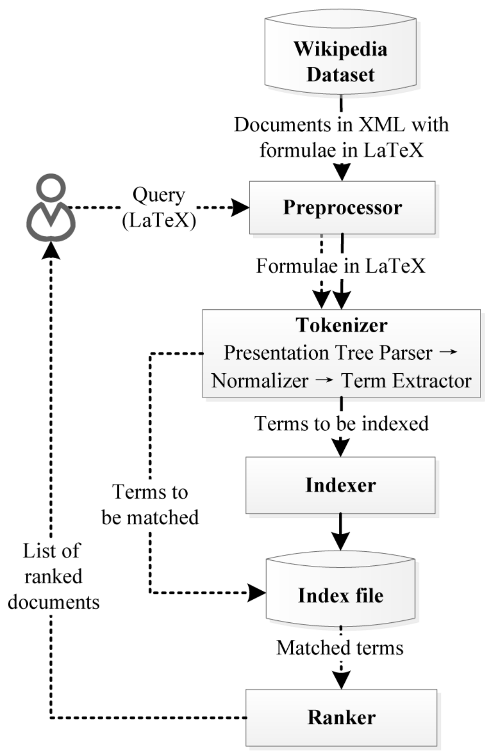

Figure 1: Workflow of WikiMirs (Dotted lines denote online query flows and solid lines denote offline index flows).

extracted hierarchically. Finally, attributes of these terms are calculated and stored in inverted index files by the indexer.

When a user searches a formula online as described by dotted lines in Figure 1, the query will be first preprocessed into the internal LaTeX markup. Then, query terms are generated by the tokenizer following a similar process set in indexing above. Next, these terms are passed into the ranker to find relevant formulae and calculate the corresponding similarity score. Lastly, a list of documents are ranked according to the similarity scores of relevant formulae and returned to the user. In the following sections, we detail the four modules of WikiMirs, the preprocessor, tokenizer, indexer and ranker.

4. PREPROCESSOR

The Wikipedia database[2] presents web pages in a single XML file in which mathematical formulae are presented via LaTeX markup tagged by "<math>" and "</math>". The database is downloaded and then parsed by the preprocessor to extract LaTeX markup of formulae by identifying the tags.

After the LaTeX markup of formulae is extracted, normalization is carried out. This has to be completed as in LaTeX authors can use many different commands and user-defined macros to create formulae that are visually and semantically identical. As documents in Wikipedia are edited by many contributors from all around the world, it is inevitable to introduce various styles and commands of LaTeX into the dataset. For instance, spaces with different widths can be produced using different commands (e.g., "\qquad", "\quad", "\;", "\," etc). To overcome the impacts of typesetting differences, four rules are adopted to normalize the LaTeX markup: *1)* Line breaks inside encoding are filtered; *2)* Spaces before and after encoding are filtered; *3)* Successive multiple spaces are converted into a single space; *4)* Different presentations of space commands are removed.

[2]`http://en.wikipedia.org/wiki/Wikipedia:Database_download`

5. TOKENIZER

5.1 Conventions

In order to propose a more reasonable index scheme, we first analyse the common conventions used in reading or understanding the structures of mathematical formulae before introducing our tokenizer and indexer.

Firstly, because structures of mathematical formulae are naturally hierarchical, they are conventionally read or understood in a hierarchical way. An assumption commonly used in previous work [13, 4, 18] is that higher level structures are more important than those at lower levels. This is because the top level usually determines the category or the skeleton of the formula, whereas the lower level nodes usually store variables or constants whose contents do not matter to the meaning of the structure. Although this convention is true when understanding a single formula, it seems unfeasible when comparing two formulae, because the same sub-structure may be present in both at different levels, thus it is hard to decide which level should be considered in calculating the importance of the match. To decide whether two formulae are similar or not, a more practical way is to compare how many contents they share at the corresponding level. In other words, the closer the levels of the matched sub-expressions are, the more they should contribute to the overall similarity score. For instance, when comparing $a^2+b^2=c^2$ with the following three expressions: $\frac{1}{a^2+b^2}=1$, $a^2+b^2=1$ and a^2+b^2, there is only one matched sub-expression, a^2+b^2. When calculating the score of this match, a higher score should be assigned to $a^2+b^2=1$ in which a^2+b^2 is in the same level with the query formula, whereas $\frac{1}{a^2+b^2}=1$ and a^2+b^2 should be assigned lower scores since the matched a^2+b^2 are in different levels compared with the query expression. According to this convention, in Section 7, a specific measurement will be proposed to calculate the similarity of term levels.

Secondly, when reading mathematical formulae with complicated structures, generalization is commonly used to simplify the formula. More specifically, most of the formulae can be viewed as expressions containing an operator with a list of operands. Generalizing the content of operands as a variable or alias is commonly applied to simplify complex structure into concise expression. For example, the expression $\frac{\sqrt{a+b}}{(c+d)^2}$ can be recognized step by step from $\frac{\sqrt{*}}{*^2}$ to $\frac{*}{*}$. Similarly, when comparing two mathematical formulae, generalized forms of expressions are commonly used to evaluate how similar two formulae appear. However, generalized forms of formulae are rarely used in existing MIR methods. In Section 5.4, a method is proposed to extract generalized sub-structures of formulae, so as to be served in finding relevant formulae.

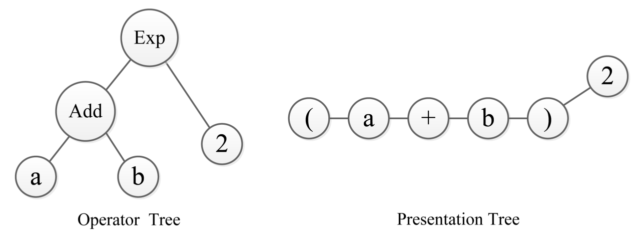

Figure 2: Operator tree and presentation tree of $(a+b)^2$ [19].

5.2 Presentation Tree Parser

The presentation tree parser aims at parsing LaTeX markup into an internal presentation tree, which is an intermediate format and is processed by downstream components, namely the normalizer and term extractor.

5.2.1 Presentation Tree

Different from an operator tree which is well-known for presenting the semantic meaning of formula directly, a presentation tree describes the spatial layout of formulae. In the presentation tree, spatial relations (e.g., horizontal, subscript, superscript, etc) are utilized to describe the geometric layouts of symbols in a formula. The presentation tree is also named as a Symbol Layout Tree in some literature [19]. Comparison of the operator tree and the presentation tree of a sample expression is given in Figure 2.

In this paper, the presentation tree of formula is presented according to the LaTeX markup. This definition can also be applied to Presentation MathML or other formula presentation formats with little modification. Basically, there are two ways to present a formula using LaTeX markup: *1)* Typically, expressions are presented by predefined operators (e.g., "\frac", "\sqrt", etc) with their operands explicitly encoded inside a pair of tags, "{}". In this paper, operators, which present their operands explicitly, are named as *explicit operator* and referred to as *ex-optr* hereafter. Their operands are referred to as *ex-opnd*. *2)* Other expressions composed of binary operators (e.g., "+", "÷") or delimiters (e.g., "(", ")") are presented linearly with their operands implicitly encoded. Each operand and each operator are placed linearly from left to right without any tag denoting operands. We call operators which present their operands implicitly as *implicit operators* and refer them as *im-optr* hereafter. Their operands are referred to as *im-opnd*. These two types of operators can be easily distinguished through checking the pair of tags, "{}" in the LaTeX markups.

According to the two types of operators defined, two templates are defined as follows to describe the expressions in LaTeX. Expressions with *ex-optr* can be parsed using Template 1 and expressions with *im-optr* can be parsed using Template 2. The mathematical formulae presented in LaTeX can be parsed by these templates or the combination of them via checking up the types of operators encountered from left to right in its LaTeX markup. Each operand (*ex-opnd* or *im-opnd*) can be a sub-expression, a variable or a constant. It should be noted that custom commands defined in user-defined macros are now processed in the same way with the pre-defined commands in LaTeX. The general presentation tree template is shown in Figure 3.

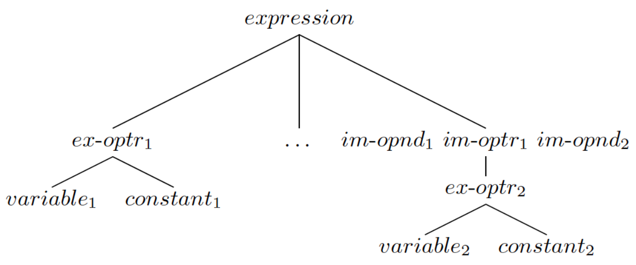

Figure 3: Presentation tree template.

Template 1:

$$ex\text{-}optr\ \{ex\text{-}opnd_1\}\{ex\text{-}opnd_2\}\ldots\{ex\text{-}opnd_n\}$$

Template 2:

$$im\text{-}opnd_1\ \ im\text{-}optr_1\ \ im\text{-}opnd_2\ \ im\text{-}optr_2\ \ \ldots\ \ im\text{-}optr_n\ \ im\text{-}opnd_{n+1}$$

5.2.2 Tree Construction

After parsing LaTeX markups of formulae using the above templates, they can be further built into presentation trees. Firstly, if an expression conforms to Template 1, the *ex-optr* is taken as a node and its operands inside "{}" tags become its children. Otherwise, the expression should conform to Template 2. In this case, there may exist more than one *im-optrs* and the priorities of these *im-optrs* are unknown without semantic analysis. Thus, it can not be decided which *im-optr* should be the node of the current level. Therefore, when the expression conforms to Template 2, the whole expression itself will be considered as a node and will not grow as one operator node with several operand children. Secondly, the operands in Template 1 and Template 2 are taken as individual expressions and parsed recursively following the above process. It is worth noting that the expression conforming to Template 2 will have children only if at least one of its operands conforms to Template 1. The tree building process continues until no more children nodes can be added into the existing tree.

An example of the presentation tree of a specific formula (see Figure 4(a)) is shown in Figure 4(c) whose LaTeX markup is described in Figure 4(b).

5.3 Normalizer

The role of the normalizer is to normalize the variables, constants and order of operands. Challenges of employing the normalizer towards LaTeX markup include: *1)* Different from MathML, in which identifying variables and constants is quite straightforward, since variables and constants are usually tagged explicitly with special tags (e.g., "ci"), in LaTeX there exist no explicit commands to denote them. *2)* Similar to Presentation MathML, LaTeX markups do not contain semantic information of formulae, so it is difficult to decide the priorities of operators without additional semantic analysis. As a result, equivalent transformation of the operand order cannot be carried out, either.

Given that, techniques on semantic understanding of mathematical formulae have not been well studied, and this paper does not leverage semantic analysis to overcome the aforementioned problems. Alternately, we propose a normalizer for variables and constants based on the

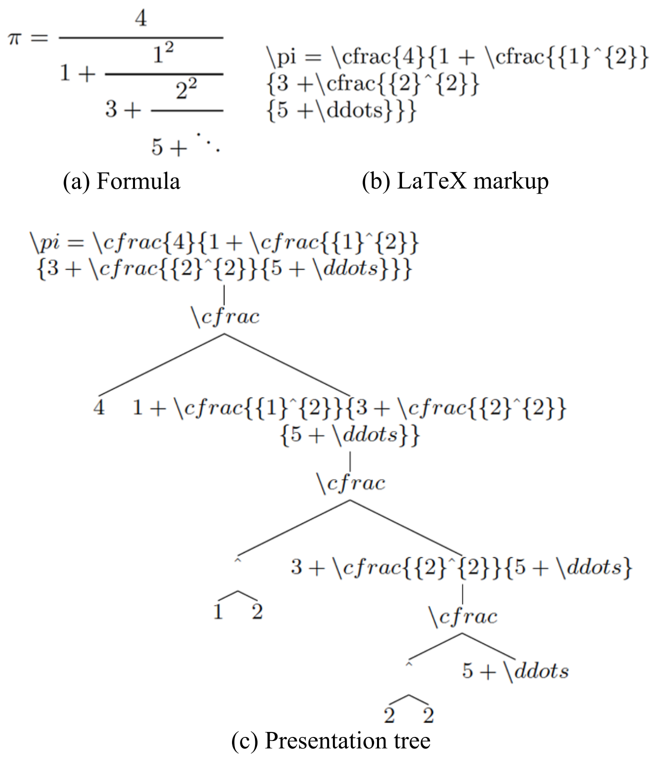

Figure 4: An example of presentation tree.

presentation tree parsed in the previous stage as explained in Section 5.2. Given a presentation tree, all leaf nodes are examined by the normalizer. If a leaf node does not confirm to Template 2, it should be identified as a variable or a constant. In this case, a unified alias will replace the content of this leaf node. If a leaf node confirms to Template 2, the ordinary presentation will be kept.

5.4 Term Extractor

As aforementioned, formulae in LaTeX are parsed into the presentation tree and then normalized. Based on the normalized presentation tree, fine-grained terms are extracted using a hierarchical generalization technique. In this way, matches in different levels, as well as matches between generalized expressions, can be taken into consideration for the similarity score calculation.

The term extractor will generate two categories of terms, namely *original terms* and *generalized terms*. The *original terms* are generated directly from the original sub-expressions of the formulae. The *generalized terms* are generated from the fuzzy sub-expressions. The *generalized terms* is proposed to describe the sketch of the expression.

For each term, three attributes are extracted and recorded in a triple, namely $\{content, level, is\_fuzzy\}$. The content of the term is recorded in the *content* field. The level of the term in the formula presentation tree is stored in the *level* field. *is_fuzzy* field denotes whether the term is a *generalized term* or not.

The term generation algorithm is illustrated in Algorithm 1. The presentation tree is traversed using depth-first search. For each expression (exp) in a different level (lev), terms are recursively generated. The original term is generated as shown in line 2 of Algorithm 1. The original LaTeX markup ($LaTeX\_markup(exp)$) of the expression is extracted as the *content*. The generalized term is extracted as shown in line 5. The content of this term is generated using Algorithm 2 to be detailed later. After the *original term* and *generalized term* are extracted, operands of the expression are traversed. If the $operand_i$ is a sub-expression rather than a variable or constant, the term generation algorithm will be recursively called with $operand_i$ and $lev + 1$ as inputs.

Algorithm 1 Term generation

```
1:  procedure TERM_OF_EXPRESSION($exp, lev$)
2:      $term\_set \leftarrow (LaTeX\_markup(exp), lev, false)$
3:                                          ▷ Extract original term
4:      if GEN_LATEX_MARKUP(exp) is not “” then
5:          $term\_set \leftarrow$ (GEN_LATEX_MARKUP($exp$), $lev, true$)
6:      end if
7:                                          ▷ Extract generalized term
8:      for $operand_i$ := each operand in $exp$ do
9:          if $operand_i$ is not a variable or constant then
10:             TERM_OF_EXPRESSION($operand_i, lev + 1$)
11:         end if
12:     end for
13: end procedure
```

In order to record the rough outline of an expression without the details of its sub-expressions, Algorithm 2 is proposed. The main idea is to reserve the structure of the top level of the expression and neglect the contents of its sub-expressions. As described in Algorithm 2, to generate the generalized LaTeX markup of an expression (exp), the operands are traversed. If the operand is a sub-expression rather than a variable or constant, all the operands of this sub-expression should be removed and replace by “{}”. In this way, a generalized term is created to record the dominating structure of the expression.

Algorithm 2 Generalized LaTeX markup generation

```
1:  procedure GEN_LATEX_MARKUP($exp$)
2:      $is\_exist \leftarrow false$
3:      $gen\_str \leftarrow$ “”
4:      for $operand_i$ := each operand in $exp$ do
5:          if $operand_i$ is not a variable or constant then
6:              replace all operands of $operand_i$ with “{}”;
7:              $gen\_str \leftarrow gen\_str\,\hat{}\,operand_i$
8:              $is\_exist \leftarrow true$
9:          else
10:             $gen\_str \leftarrow gen\_str\,\hat{}\,operand_i$
11:         end if
12:     end for
13:     if $is\_exist$ then return $gen\_str$
14:     else return “”
15:     end if
16: end procedure
```

Taking the expression in Figure 4 as an example, the terms generated are illustrated in Table 2.

6. INDEXER

To run queries more efficiently we employ the inverted index data structure, which is commonly used in text-based

Table 2: Terms of Expression in Figure 4

Terms			Expression
Content[†]	Level[‡]	Fuzzy	
\pi=\cfrac{C}{1+ \cfrac{{C}^{C}}{3+ \cfrac{{C}^{C}}{5+ \ddots}}}	1	false	$\pi = \cfrac{4}{1+\cfrac{1^2}{3+\cfrac{2^2}{5+\ddots}}}$
\pi=\cfrac{}{}	1	true	$\pi = \frac{*}{*}$
1+\cfrac{{C}^{C}} {3+\cfrac{{C}^{C}} {5+\ddots}}	2	false	$1+\cfrac{1^2}{3+\cfrac{2^2}{5+\ddots}}$
1+\cfrac{}{}	2	true	$1+\frac{*}{*}$
{C}^{C}	3	false	1^2
3+\cfrac{{C}^{C}} {5+\ddots}	3	false	$3+\cfrac{2^2}{5+\ddots}$
3+\cfrac{}{}	3	true	$3+\frac{*}{*}$
{C}^{C}	4	false	2^2
5+\ddots	4	false	$5+\ddots$

[†] "C" denotes the normalized alias of constant.
[‡] Level of root is defined as 1.

Table 3: Data Structure of the Inverted Index File

Term	iff	$formula_1$	...	$formula_j$
t_1	$iff(t_1)$	$tf(t_1, formula_1)$ $tl(t_1, formula_1)$	...	$tf(t_1, formula_j)$ $tl(t_1, formula_j)$
...	...	...	...	...
t_i	$iff(t_i)$	$tf(t_i, formula_1)$ $tl(t_i, formula_1)$	...	$tf(t_i, formula_j)$ $tl(t_i, formula_j)$

search engine, to index formula terms. For each term, a list of formulae which contain this term is recorded. A formula (f) "contains" a term (t) denotes that the *content* field of one of the terms extracted from f is exactly the same with that of t. The structure of the inverted index file is presented in Table 3. Term level $tl(t_i, formula_j)$ denotes level of t_i in $formula_j$. However, since there may be several occurrences of t_i in $formula_j$ and their term levels may also be different, $tl(t_i, formula_j)$ is defined as a 64-bit integer in which the k-th bit denotes whether t_i appears in the k-th level of the formula. Similar to the traditional text-based information retrieval model, for each term (t_i), $iff(t_i)$ is employed to describe the inverted formula frequency of t_i as defined in Equation 1:

$$iff(t_i) = \log \frac{Number\ of\ formulae}{1 + Number\ of\ formulae\ containing\ t_i}. \quad (1)$$

$tf(t_i, formula_j)$ describes the frequency of t_i occurring in $formula_j$ as defined in Equation 2:

$$tf(t_i, formula_j) = \frac{Number\ of\ t_i\ in\ formula_j}{Number\ of\ terms\ in\ formula_j}. \quad (2)$$

As previously stated, the tokenizer extracts two types of terms, *original terms* and *generalized terms*. In the indexer, *original terms* and *generalized terms* are indexed in two inverted index files separately according to the *is_fuzzy* field of terms. This is because separated index files can support flexible query demands efficiently. If an exact query is requested, only the index file of the *original terms* should be looked up, since the matches in the index file of *generalized terms* will not contribute to the overall similarity score calculation and the time of checking the generalized terms can be saved.

7. RANKER

When users search a formula via entering the LaTeX markup of the formula, WikiMirs first preprocesses the query into normalized LaTeX markups as described in Section 4. Then, the query is tokenized into terms by the tokenizer as introduced in Section 5. After the terms are generated, they are used to look up all the matched terms in the inverted index file. For each term t generated from the query Q, a similarity score is calculated as defined in Equation 3:

$$score(Q, F) = \sum_{t \in Q} tf(t, F) \times iff(t) \times W_{lev}(t, Q, F) \times W_{gen}(t), \quad (3)$$

$iff(t)$ and $tf(t, F)$ are defined in Equation 1 and Equation 2 respectively. The similarity score between the query and a formula is calculated by summing up the similarity score of each matched term. A weight of level, $W_{lev}(t, Q, F)$, is introduced to evaluate the distance of the matched terms on different levels, as defined in Equation 4:

$$W_{lev}(t, Q, F) = \frac{1}{1 + \min_i\{|level(t, Q) - level_i(t, F)|\}}. \quad (4)$$

In this equation, $tf(t, F)$ is stored in the index file and $tf(t, Q)$ is obtained by the tokenizer. It is worth noting that there may be several occurrences of the term t in the formula F and they might be at different levels of the formula F. Therefore, different level distances may be obtained through calculating distances between the query term level ($level(t, Q)$) and each matched term level ($level_i(t, F)$). In our system, a minimum among these level distances is taken as the level distance between the query term t and the formula F. In order to ensure that the exactly matched terms would always get higher similarity score than those who are only approximately matched, the weight of generalization, $W_{gen}(t)$, is also defined in Equation 5:

$$W_{gen}(t) = \begin{cases} 1 & \text{if } t \text{ is not generalized} \\ 0.5 & \text{if } t \text{ is generalized} \end{cases}. \quad (5)$$

8. EXPERIMENTAL RESULTS

In this section, the query efficiency of the proposed system, WikiMirs, is evaluated. In addition, we compare the accuracy of WikiMirs with two existing search engines; Wikipedia Search and Egomath [2].

8.1 Dataset

The dataset used in WikiMirs is the Wikipedia dataset. The version being used is the 2012-10-01 dump with a size of approximately 8.7 GB compressed and 30 GB uncompressed. This dataset contains approximately 13 millions web pages and 495,958 mathematical formulae. Compared with other mathematical document corpuses (e.g., ArXiv), which are relative more domain-specific or academic, Wikipedia is widely used by users with different background. Because there is no benchmark dataset for evaluation in MIR research community, it is difficult to compare performances of MIR systems. However, since Wikipedia is a publicly available dataset and some reported work [14] does use it, performance evaluation can be done through using the same Wikipedia dataset.

8.2 Query Efficiency

To evaluate the query efficiency of our proposed method, growth trends of the index file size and the query time with increasing amount of indexed formulae are evaluated. Table 4 depicts the sizes of index files with increasing amount of indexed formulae. It can be seen that as the number of indexed formulae grows at ten times scale, the size of index file increases steadily. Table 5 illustrates the query times with increasing amount of indexed formulae. The minimum, maximum and average query time over ten test queries are tested with an increasing number of indexed formulae. On average, the query time is less than 1 second when the index size is less than 420 MB. The steady trend of the index file size and query time growing with the number of indexed formulae implies that the WikiMirs has good scalability for real-world realistic applications with a much larger dataset.

Table 4: Sizes of Index Files with an Increasing Number of Indexed Formulae

Number of formulae	1,000	10,000	100,000	495,958
Size (MB)	0.26	1.83	21.41	419.55

Table 5: Query Times with an Increasing Number of Indexed Formulae

Number of formulae	1,000	10,000	100,000	495,958
Min (ms)	0.48	0.48	0.51	1.1
Max (ms)	58.04	115.34	922.42	1431.41
Average (ms)	9.76	31.56	222.57	510.65

8.3 Accuracy

To evaluate the accuracy of the results found by WikiMirs, experiments are carried out to compare WikiMirs with Wikipedia Search and Egomath. In addition, it is worthy to note that the traditional search engines perform rather poorly in searching formulae on Wikipedia according to our testing. In fact, so poorly that it seems meaningless to compare with Google and Bing. Therefore, we choose to compare the proposed system with Wikipedia Search and Egomath rather than those traditional search engines. Wikipedia Search is a search engine built in Wikipedia, and it can be used to locate content on Wikipedia based on plain text retrieval techniques. Egomath is a text-based math search engine on Wikipedia. To compare their accuracies, we first analyse the retrieval results of several sample queries, then compare the *Precision* and *Discounted Cumulative Gain* (*DCG*) of their top-k retrieval results.

8.3.1 Case Study

In this section, specific expressions are selected from different categories as sample queries to test in Wikipedia Search, Egomath and WikiMirs. Table 6 shows six sample queries with their retrieved expressions, ordered by their similarity scores with the queries.

Taking the first query ($\frac{x^2-1}{x^2+1} - \frac{2ix}{x^2+1}$) as an example, Wikipedia Search finds relevant formulae according to distinguishable terms in the query (e.g., "\frac" or "xˆ2"), so formulae with these terms are ranked first. High frequencies of the particular terms (e.g., "\frac") will cause irrelevant results to rank highly, such as the fourth and fifth formulae. In contrast to Wikipedia Search, which obtains high recall but low accuracy, Egomath returns only one result for this query. A portion of this retrieved formula is exactly matched with the query. WikiMirs finds the exactly matched expression followed by a list of partially matched expressions which share similar sub-structures with the query formula.

Let us look at the the fourth query ($\sum_{i=0}^{\infty} a_i X^i$). Wikipedia Search ranks the results according to frequency of particular terms in the query (e.g., "\sum" and "xˆ"). Egomath returns no result mainly because of its exact matching policy. WikiMirs finds the expressions with similar structures with the query and ranks them according to structure similarities. The upper bounds of "$\sum$" in these expressions are "n" rather than "∞", but this difference should not matter in the structure matching. Owing to the generalization technique presented in our tokenizer, this difference is ignored properly and therefore fuzzy matched formulae are found.

8.3.2 Performance of the Top-k Results

Figure 5 shows the performance of Wikipedia Search, Egomath and WikiMirs based on 20 test queries. To evaluate the performance results, average *Precision* and *Discounted Cumulative Gain* (*DCG*) are calculated respectively, based on the top 5, 10 and 20 retrieved expressions over all queries. *Precision* is a set-based evaluation metric which is used to evaluate the accuracy of retrieved results without considering the difference of positions of results in the ranked list. In order to take the order of results into consideration, *DCG* is introduced. Using a graded relevance scale of the formula in the ranked list, *DCG* measures the relevant score of a formula based on its position in the result list. In other words, more contribution is made if a relevant formula is ranked higher in the result list, otherwise less contribution is made.

For each test query, the top k results are retrieved and then labelled manually by subjects to denote whether the retrieved formula is relevant to the query or not. For each query, a list of labels of the top-k results are obtained. Here, the list is named rel in which the i-th element (rel_i) denotes whether the i-th retrieved formula is relevant to the query ($rel_i = 1$) or not ($rel_i = 0$). Based on the list rel, P of the top-k retrieved results can be calculated using Equation 6:

$$P_k = \sum_{i=1}^{k} \frac{rel_i}{k}. \tag{6}$$

DCG of the top-k retrieved results can be calculated using Equation 7:

$$DCG_k = \sum_{i=1}^{k} \frac{2^{rel_i} - 1}{\log_2(i+1)}. \tag{7}$$

It is shown from Figure 5 that WikiMirs outperforms Wikipedia Search and Egomath in P_5, P_{10} and P_{20} by more

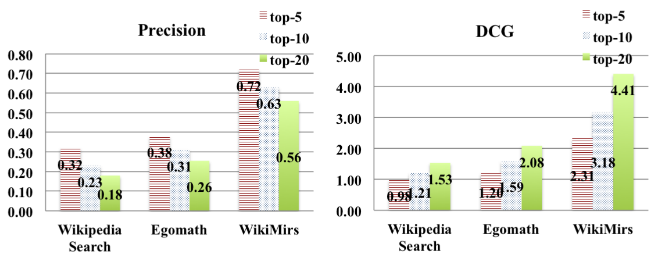

Figure 5: Comparison of Precision & DCG of the top-5, top-10 and top-20 results.

than 30%. It is also seen that DCG_5, DCG_{10} and DCG_{20} of WikiMirs are about twice higher than those of Egomath and Wikipedia Search, showing that WikiMirs ranks the relevant results more properly. Many irrelevant formulae are found and ranked highly by Wikipedia Search because it retrieves results according to textual contents solely. In Egomath, although some features of math formulae are considered, due to the exact matching technique adopted, limited amount of results are found and sometimes only less than 5 results are returned. This is the main reason causing low precision and DCG in Egomath.

9. CONCLUSIONS

In this paper, we propose WikiMirs, a tool to facilitate searching mathematical formulae, which consists of four major components, preprocessor, tokenizer, indexer, and ranker. In order to support both the textual and structural matching of formulae, we propose a tokenizer to produce fine-grained sub-structures of mathematical formulae, extracted using a hierarchical generalization technique from the presentation tree. The ranker calculates the similarity between formulae using both text matching and structure matching at different tree levels. Experimental results show that WikiMirs has both satisfactory query efficiency and good scalability. Experiments on accuracy have been carried out from several different perspectives. They show that WikiMirs retrieves more accurate results and ranks formulae better in Wikipedia compared to two existing search engines, Wikipedia Search and Egomath. In order to facilitate the wide adoption of MIR in real-world digital libraries, it is important to study the gap between the performance of MIR and the user's expectations. Therefore, we plan to conduct a survey of user requirements of the digital libraries. Moreover, utilising the contexts and semantics of mathematical formulae, as well as the metadata of the documents to improve the performance of mathematical information retrieval is another meaningful research direction.

10. ACKNOWLEDGMENTS

This work is supported by the National Natural Science Foundation of China (No.61202232) and National Key Technology R&D Program of China (No.2012BAH40F01). We also thank the anonymous reviewers for their valuable comments.

11. REFERENCES

[1] http://dlmf.nist.gov/.
[2] http://egomath.projekty.ms.mff.cuni.cz/.
[3] http://search.mathweb.org/.
[4] https://mir.fi.muni.cz/mias/.
[5] http://www.latexsearch.com/.
[6] http://www.mathjax.org/.
[7] http://www.openmath.org/.
[8] http://www.w3.org/math/.
[9] A. Asperti, F. Guidi, C. Coen, E. Tassi, and S. Zacchiroli. A content based mathematical search engine: Whelp. *Types for Proofs and Programs*, pages 17–32, 2006.
[10] Y. Hijikata, H. Hashimoto, and S. Nishida. Search mathematical formulas by mathematical formulas. *Human Interface and the Management of Information. Designing Information Environments*, pages 404–411, 2009.
[11] M. Kohlhase and I. Sucan. A search engine for mathematical formulae. In *Artificial Intelligence and Symbolic Computation*, pages 241–253. Springer, 2006.
[12] B. Miller and A. Youssef. Technical aspects of the digital library of mathematical functions. *Annals of Mathematics and Artificial Intelligence*, 38(1):121–136, 2003.
[13] R. Miner and R. Munavalli. An approach to mathematical search through query formulation and data normalization. *Towards Mechanized Mathematical Assistants*, pages 342–355, 2007.
[14] J. Mišutka and L. Galamboš. Extending full text search engine for mathematical content. *Towards Digital Mathematics Library. Birmingham, United Kingdom, July 27th, 2008*, pages 55–67, 2008.
[15] T. Nguyen, S. Hui, and K. Chang. A lattice-based approach for mathematical search using formal concept analysis. *Expert Systems with Applications*, 2011.
[16] T. T. Nguyen, K. Chang, and S. C. Hui. A math-aware search engine for math question answering system. In X. wen Chen, G. Lebanon, H. Wang, and M. J. Zaki, editors, *CIKM*, pages 724–733. ACM, 2012.
[17] T. Schellenberg, B. Yuan, and R. Zanibbi. Layout-based substitution tree indexing and retrieval for mathematical expressions. In *Proceedings of SPIE*, volume 8297, page 82970I, 2012.
[18] P. Sojka and M. Líška. Indexing and searching mathematics in digital libraries. *Intelligent Computer Mathematics*, pages 228–243, 2011.
[19] R. Zanibbi and D. Blostein. Recognition and retrieval of mathematical expressions. *International Journal on Document Analysis and Recognition*, pages 1–27, 2012.
[20] R. Zanibbi and B. Yuan. Keyword and image-based retrieval of mathematical expressions. In *IS&T/SPIE Electronic Imaging*, pages 78740I–78740I. International Society for Optics and Photonics, 2011.
[21] J. Zhao, M.-Y. Kan, and Y. L. Theng. Math information retrieval: user requirements and prototype implementation. In *Proceedings of the 8th ACM/IEEE-CS joint conference on Digital libraries*, pages 187–196. ACM, 2008.

Table 6: Comparison of the Top-5 Retrieval Results of the Sample Queries

Queries	Systems	Top-5 results
$\frac{x^2-1}{x^2+1}-\frac{2ix}{x^2+1}$ (Sub-structure match)	Wikipedia Search	1) $\frac{x^2+1}{(x+2)(x-1)\left(x^2+x+1\right)}$; 2) x^n-1; 3) $\frac{d}{dx}arctanx=\frac{1}{x^2+1}$; 4) $E_{KL}=\frac{1}{2}\left(\frac{U_K}{X_L}+\frac{U_L}{X_K}+\frac{U_M}{X_K}\frac{U_M}{X_L}\right)$; 5) $B_{16}\left(x\right)=x^{16}-8x^{15}+20x^{14}-\frac{182}{3}x^{12}+\frac{572}{3}x^{10}-429x^8+\frac{1820}{3}x^6-\frac{1382}{3}x^4+140x^2-\frac{3617}{510}$
	Egomath	1) $g\left(x\right)=\frac{x^2-1}{x^2+1}-\frac{2ix}{x^2+1}$; No more results.
	WikiMirs	1) $g(x)=\frac{x^2-1}{x^2+1}-\frac{2ix}{x^2+1}$; 2) x^2+1; 3) $Imf\left(f\left(x\right)\right)=\frac{2x}{x^2+2}$; 4) $\frac{1}{x^2+1}$; 5) $\left(\arctan\left(x\right)\right)'=\frac{1}{x^2+1}$
$\frac{\tan\frac{A-B}{2}}{\tan\frac{A+B}{2}}$ (Sub-structure match)	Wikipedia Search	1) $\tan\frac{\alpha}{2}\tan\frac{\beta}{2}+\tan\frac{\beta}{2}\tan\frac{\gamma}{2}+\tan\frac{\gamma}{2}\tan\frac{\alpha}{2}=1$; 2) $\frac{a-b}{a+b}=\frac{\tan[\frac{1}{2}(\alpha-\beta)]}{\tan[\frac{1}{2}(\alpha+\beta)]}$; 3) $\frac{a-b}{a+b}=\frac{\tan[\frac{1}{2}(\alpha-\beta)]}{\tan[\frac{1}{2}(\alpha+\beta)]}$; 4) $\frac{\tan\frac{A-B}{2}}{\tan\frac{A+B}{2}}=\frac{a-b}{a+b}\frac{\tan\frac{A-C}{2}}{\tan\frac{A+C}{2}}=\frac{a-c}{a+c}\frac{\tan\frac{B-C}{2}}{\tan\frac{B+C}{2}}=\frac{b-c}{b+c}$; 5) $\frac{\frac{d[D]}{dt}}{\frac{d[C]}{dt}}=\frac{k_2K[A]}{k_1[A]}=\frac{k_2K}{k_1}$
	Egomath	No result.
	WikiMirs	1) $\frac{\tan\frac{A-B}{2}}{\tan\frac{A+B}{2}}=\frac{a-b}{a+b}\frac{\tan\frac{A-C}{2}}{\tan\frac{A+C}{2}}=\frac{a-c}{a+c}\frac{\tan\frac{B-C}{2}}{\tan\frac{B+C}{2}}=\frac{b-c}{b+c}$; 2) $A-B$; 3) $A+B$; 4) $\frac{1}{AB}=\frac{1}{A-B}\left(\frac{1}{B}-\frac{1}{A}\right)=\frac{1}{A-B}\int_B^A\frac{1}{z^2}dz$; 5) $\left(0.9\left(\frac{A+B}{2}+\frac{A-B}{2}\sin\left(4\right)\pi f_pt\right)+0.1\sin\left(2\right)\pi f_pt\right)\times75kHz$
$\sqrt{2+\sqrt{2+\sqrt{2}}}$ (Sub-structure match)	Wikipedia Search	1) $\sin(84\frac{3}{8}^\circ)=\frac{1}{2}\sqrt{2+\sqrt{2+\sqrt{2+\sqrt{2}}}}$; 2) $x=\sqrt{2+\sqrt{2+\sqrt{2+\sqrt{2+\cdots}}}}$; 3) $\frac{\sqrt{2}}{2}\cdot\frac{\sqrt{2+\sqrt{2}}}{2}\cdot\frac{\sqrt{2+\sqrt{2+\sqrt{2}}}}{2}\cdot\cdots=\frac{2}{\pi}$; 4) $\pi=2\times\frac{2}{\sqrt{2}}\times\frac{2}{\sqrt{2+\sqrt{2}}}\times\frac{2}{\sqrt{2+\sqrt{2+\sqrt{2}}}}\times\frac{2}{\sqrt{2+\sqrt{2+\sqrt{2+\sqrt{2}}}}}\times\cdots$; 5) $2+\sqrt{2}+\sqrt{6}$
	Egomath	1) $\pi=2\times\frac{2}{\sqrt{2}}\times\frac{2}{\sqrt{2+\sqrt{2}}}\times\frac{2}{\sqrt{2+\sqrt{2+\sqrt{2}}}}\times\frac{2}{\sqrt{2+\sqrt{2+\sqrt{2+\sqrt{2}}}}}\times\ldots$; 2) $\frac{2}{\pi}=\frac{\sqrt{2}}{2}\cdot\frac{\sqrt{2+\sqrt{2}}}{2}\cdot\frac{\sqrt{2+\sqrt{2+\sqrt{2}}}}{2}\ldots$; No more results.
	WikiMirs	1) $\sin(78\frac{3}{4}^\circ)=\frac{1}{2}\sqrt{2+\sqrt{2+\sqrt{2}}}$; 2) $\sin(84\frac{3}{8}^\circ)=\frac{1}{2}\sqrt{2+\sqrt{2+\sqrt{2+\sqrt{2}}}}$; 3) $\pi=2\times\frac{2}{\sqrt{2}}\times\frac{2}{\sqrt{2+\sqrt{2}}}\times\frac{2}{\sqrt{2+\sqrt{2+\sqrt{2}}}}\times\frac{2}{\sqrt{2+\sqrt{2+\sqrt{2+\sqrt{2}}}}}\times\cdots$; 4) $\frac{2}{\pi}=\frac{\sqrt{2}}{2}\cdot\frac{\sqrt{2+\sqrt{2}}}{2}\cdot\frac{\sqrt{2+\sqrt{2+\sqrt{2}}}}{2}\cdots$; 5) $2+\sqrt{2}$
$\sum_{i=0}^{\infty}a_iX^i$ (Fuzzy match)	Wikipedia Search	1) $\int x^m\exp\left(ix^n\right)dx=\int\sum_{l=0}^{\infty}\frac{i^lx^{m+nl}}{l!}dx=\sum_{l=0}^{\infty}\frac{i^l}{(m+nl+1)}\frac{x^{m+nl+1}}{l!}$; 2) $\frac{1}{(1-x)^2}=\sum_{n=1}^{\infty}nx^{n-1}$; 3) $e_1=\sum_i x_i$; 4) $\sum_{i=1}^{n}w_i{x_i}^p{}^{\frac{1}{p}}\leq\sum{}_{i=1}^{n}w_i{x_i}^q{}^{\frac{1}{q}}$; 5) $\frac{1}{NT}\int_{NT}\left[\sum_{n=-\infty}^{\infty}x[n]\cdot\delta(t-nT)\right]e^{-i2\pi\frac{k}{NT}t}dt$
	Egomath	No result.
	WikiMirs	1) $\left(\sum_{i=0}^{n}a_iX^i\right)+\left(\sum_{i=0}^{n}b_iX^i\right)=\sum_{i=0}^{n}\left(a_i+b_i\right)X^i$; 2) $\left(\sum_{i=0}^{n}a_iX^i\right)+\left(\sum_{i=0}^{n}b_iX^i\right)=\sum_{i=0}^{n}\left(a_i+b_i\right)X^i$; 3) $f\left(x\right)=\sum_{i=0}^{n}a_ix^i;g\left(x\right)=\sum_{i=0}^{m}b_ix^i$; 4) $i=0$; 5) $\ldots=\sum_{i=0}^{N}\left(N\right)_iA^{-i}-\sum_{i=0}^{N}\left(N-1\right)_iA^{-i}$
$\log\frac{z}{(1-z)^2}$ (Sub-structure match & Fuzzy match)	Wikipedia Search	1) $\arcsin\left(z\right)=-i\log\left(iz+\sqrt{1-z^2}\right)$; 2) $\Gamma\left(z\right)\Gamma\left(z+\frac{1}{2}\right)=2^{1-2z}\sqrt{\pi}\Gamma\left(2z\right)$; 3) ${}_2F_1(a,1-a;c;z)=\Gamma(c)z^{\frac{1-c}{2}}(1-z)^{\frac{c-1}{2}}P_{-a}^{1-c}(1-2z)$; 4) $\mathrm{Li}_{-1}\left(z\right)=\sum_{k=1}^{\infty}kz^k=\frac{z}{(1-z)^2}$; 5) $\frac{1}{1-z}\sum\left(k-1\right)\frac{z^k}{k}=\frac{z}{(1-z)^2}-\frac{1}{1-z}\log\left(\frac{1}{1-z}\right)$
	Egomath	No result.
	WikiMirs	1) $\frac{1}{2}\exp\left(\frac{1}{2}\log\left(\frac{1}{(1-z)^2}\right)\right)+\frac{1}{2}\exp\left(\frac{1}{2}\log\left(1+z\right)^2\right)$; 2) $=\frac{a^2}{1-z}+\frac{az}{(1-z)^2}+\frac{az}{(1-z)^2}+\frac{z}{(1-z)^2}+\frac{2z^2}{(1-z)^3}=\frac{a^2}{1-z}+\frac{(2a+1)z}{(1-z)^2}+\frac{2z^2}{(1-z)^3}$; 3) $\mathrm{Li}_{-1}\left(z\right)=\sum_{k=1}^{\infty}kz^k=\frac{z}{(1-z)^2}$; 4) $\log\left(\frac{1}{\lvert x\rvert}\right)$; 5) $=\frac{a^2}{1-z}+\frac{(2a+1)z}{(1-z)^2}+\frac{2z^2}{(1-z)^3}$
$\sum_{n=0}\frac{f^{(n)}(a)}{n!}(x-a)^n$ (Sub-structure match & Fuzzy match)	Wikipedia Search	1) $N(x)=\sum_{n=0}^{\infty}H(x-E_n)$; 2) $\sum_{n=0}^{\infty}\frac{f^{(n)}(a)}{n!}(x-a)^n$; 3) $\left(\sum_{n=0}^{\infty}a_nx^n\right)\left(\sum_{n=0}^{\infty}b_nx^n\right)=\sum_{i=0}^{\infty}\sum_{j=0}^{\infty}a_ib_jx^{i+j}=\sum_{n=0}^{\infty}\left(\sum_{i=0}^{n}a_ib_{n-i}\right)x^n=\sum_{n=0}^{\infty}c_nx^n$; 4) $r_6(n)=\frac{\pi^3n^2}{2}\left(\frac{c_1(n)}{1}-\frac{c_4(n)}{8}-\frac{c_3(n)}{27}-\frac{c_8(n)}{64}+\frac{c_5(n)}{125}-\frac{c_{12}(n)}{216}-\frac{c_7(n)}{343}-\frac{c_{16}(n)}{512}+\ldots\right)$; 5) $T_f(z)=\sum_{k=0}^{\infty}\frac{(z-c)^k}{2\pi i}\int_\gamma\frac{f(w)}{(w-c)^{k+1}}dw=\frac{1}{2\pi i}\int_\gamma\frac{f(w)}{w-c}\sum_{k=0}^{\infty}\left(\frac{z-c}{w-c}\right)^kdw=\frac{1}{2\pi i}\int_\gamma\frac{f(w)}{w-c}\left(\frac{1}{1-\frac{z-c}{w-c}}\right)dw=\frac{1}{2\pi i}\int_\gamma\frac{f(w)}{w-z}dw=f(z)$
	Egomath	No result.
	WikiMirs	1) $\sum_{n=0}^{\infty}\frac{f^{(n)}(a)}{n!}(x-a)^n$; 2) $\left(\sum_{n=0}^{\infty}a_nX^n\right)\left(\sum_{n=0}^{\infty}b_nX^n\right)=\sum_{n=0}^{\infty}c_nX^n$; 3) $(\sum_{n=0}^{\infty}2^nx_n,\sum_{n=0}^{\infty}2^ny_n)\leftrightarrow\sum_{n=0}^{\infty}2^{2n}x_n+\sum_{n=0}^{\infty}2^{2n+1}y_n$; 4) $u=\sum_{n=0}^{\infty}f_i^{(n)}u_n=\sum_{n=0}^{\infty}e_i^{(n)}v_n$; 5) $n=0$