An Approach to Math-Similarity Search

Qun Zhang and Abdou Youssef

Department of Computer Science
The George Washington University
Washington DC, 20052, USA

Abstract. The unique structural syntax and the variety of semantic equivalences of mathematic expressions make it a challenge for a keyword-based text search engine to effectively meet the users' search needs. Many existing math search solutions focus on exact search where the notational matching determines the relevance rank, while the structural similarity and mathematical semantics are often missed out or not addressed adequately. One important research question is how to effectively and efficiently find math expressions that are similar to a user's query, and how to do relevance ranking of hits by similarity. This paper focuses on (1) conceptualizing similarity between mathematical expressions, (2) defining metrics to measure math similarity, (3) utilizing those metrics for math similarity search, and (4) evaluating performance to validate advantage of the proposed math similarity search. Our results show that the performance of math-similarity search is superior to that of keyword-based math search.

Keywords: math search, similarity search, similarity metric, similarity ranking, Strict Content MathML.

1 Introduction

More and more math knowledge has become available on the Web, and search is a gate to such vast treasure of digital mathematics content [11]. Even though Information Retrieval technology has reached maturity, math retrieval is still in its nascent stages, and many challenges remain. Those challenges are due in part to the significant difference of math knowledge from other textual documents. A math expression is often written in a symbolic language with several levels of abstraction, and often contains rich structural information. Additionally, notational ambiguities, and syntactical and semantic equivalences, make math knowledge harder to search. Furthermore, similarity search in math needs to capture not only the taxonomically similar operation or function names but also the hierarchically similar structures (we will use the term "function" to encompass both "function" and "operator"). For example, $x^2 + y^2 + z^2$ is expected by the user to match $a^2 + b^2 + c^2$ due to the structural similarity of the two expressions. The great inference on the structural aspect and semantic aspects of math expressions calls for a search engine that is capable of detecting and measuring similarity between mathematical constructs.

S.M. Watt et al. (Eds.): CICM 2014, LNAI 8543, pp. 404–418, 2014.

Most "first-generation" math search systems are full text-search-based math search systems that treat math objects as linear strings. However, this approach often misses out the structural information of the math expressions, and makes it nearly impossible to find a semantically similar math expression. On the other hand, there are XML-based math search solutions that identify the common sub-paths between the query expression and the candidate expressions. However, XML-based search methods often limit search to exact matches without systematically measuring the structural similarity or the semantic similarity between the query expression and the candidate expression. Similarity search enables users to find additional knowledge, discover latent relationships to different fields, and compensate for false recognition [13].

In this paper, we will lay out certain fundamental facts about math-similarity search to find, for a given user query math expression, the math expressions that are structurally and semantically similar to the query. The specific goals of this paper are:

1. Conceptualize math similarity in a way that makes it possible to measure and utilize similarity in math search;
2. Develop and study math similarity metrics to measure the similarity between two math expressions;
3. Develop algorithms for computing math-similarity metrics;
4. Leverage the NIST Digital Library of Mathematical Functions [1] to build "ground truth" of math queries and corresponding matching expressions with human experts' knowledge input;
5. Implement a ranking comparison metric to benchmark the results of a math search against the "ground truth".

The rest of the paper starts with a brief summary of the related work in Section 2. It then elaborates our research work in Section 3, and draws conclusions in Section 4.

2 Background

Existing math search engines can be categorized as text-based and structure-based. Text-based math search engines extend full-text search to achieve math awareness by transforming math expressions into either equivalent linear text tokens or expanded bags of text tokens. Miller, Youssef, et al. [6], [14], [15] developed the first generation of an equation-based math search system as part of the DLMF project at NIST. They developed an innovative TexSN (i.e. Textualization, Serialization/Scoping, and Normalization) process to convert math to text, and built a math search engine on top of existing text search technology. However the conversion process loses considerable structural, and captures little semantics. Additionally, its relevance ranking leaves room for improvement. Because it is one of the few deployed math search engines that are available for us, we leverage it for performance evaluation.

Some other text-based math search engines include Mathdex [7], EgoMath [8], and MIaS [11], ActiveMath [5]. They all took advantage of the mature and optimized text search engines that are already available. But like the DLMF they are forced to transform math expressions into the form that the text search engine can effectively process, leading to the destruction of much of the native structures of the expressions, and thus preventing truly structural or similarity search from taking place. Structure-based math search systems, on the other hand, use a radically different approach based on emerging XML-based technologies and markup languages. Those math search systems analyze the structure inherent in the content representations, and statistically identify the math expressions that have the most common sub-structures with the query expressions.

Kohlhase et al. [4] implemented MathWebSearch which leverages the semantic information that resides in the structured math equation written in MathML or OpenMath. With the adoption of the unique substitution tree indexing technique, it provides the full support of alpha-equivalence matching and sub-equation matching. However, MathWebSearch does not provide relevance ranking or similarity search.

Other structure-based math search engines include DFS & BFS Index of MathML DOM [2], Waterloo Math Retrieval System [3]. They often leverage the metadata to extract semantic annotations. But most of them either simply rank the candidate hits by basic statistical methods such as count of the occurrences of the matching sub-structures, or not pay enough attention to the matching function to calculate the similarity score between the math expressions [13].

The paramount challenge of math search is to identify relevant results by finding expressions that are similar to a query expression while allowing for difference in variable names, order, and structure. However, the lack of a definition for similarity between math expressions, and the inadequacy of exact-match searching, makes the problem of math search even harder [3]. To the best of our knowledge, there are very few efforts in math similarity search for MathML encoded expressions; Yokoi and Aizawa [13]'s work is by far the only significant one. They introduced a similarity measure that is based on the "Subpath Set" of Content MathML syntactic trees. A "Subpath Set" is defined as "the paths from the root to the leaves and all the sub-paths of those paths". Trees whose "Subpath Sets" overlap with each other are considered to be similar. The significance of their approach is that, rather than the notational similarity of tokens that the conventional math search engines evaluate, they focused on the structural similarity of MathML expressions, which we do as well. But they miss the semantic aspect in the similarity measure. Due to the numerous variations of Content MathML expressions to express one math expression, without sufficient normalization it is hard for the search engine to find semantically equivalent expressions which only differ syntactically from the query expression. Additionally, little performance evaluation was done in the aspect of ranking.

In the latest W3C release of MathML, MathML 3, a subset of Content MathML is defined: Strict Content MathML. This uses a minimal, but sufficient, set of

elements to represent the meaning of a mathematical expression in a uniform and unambiguous structure [12].

Strict Content MathML requires only 10 XML Elements to be understood by MathML 3 processors, namely: *m:apply, m:bind, m:bvar, m:csymbol, m:ci, m:cn, m:cs, m:share, m:semantics, m:cerror,* and *m:cbytes.* This provides a great economy for implementation. On the other hand, MathML 3 assigns semantics to content markup by defining a mapping from arbitrary Content MathML to Strict Content MathML, and W3C even laid out a nine-step algorithm [12] to transform an arbitrary Content MathML expression into a Strict Content MathML counterpart. We limit our work to math expressions that can be encoded with Strict Content MathML. Given all these special characteristics of Strict Content MathML, it is chosen for the MathML search implementation in our research.

3 Our Approach to Math-Similarity Search

To the best of our knowledge, there is no solution available to address the similarity measurement of the Strict Content MathML expressions. This motivated us to start the research effort by addressing similarity and taking the structure-based approach to implementing semantics-sensitive math-aware similarity search with native math language MathML as query input.

3.1 Research Problem

Our research problem is defined as follows: Given a math expression that is encoded in Strict Content MathML, identify a list of structurally and semantically similar math expressions from a library of Strict Content MathML encoded math expressions, and sort the list by similarity according to some similarity measure. Specifically, the tasks of our research include: identify conceptual factors to math similarity, deduce math similarity metric, implement the math similarity metric, evaluate and refine the math similarity metric.

3.2 Math Similarity Factors

Influenced by the Multidimensional Relevance Metric proposed by [15], we came up with the vector model based multidimensional similarity metric which takes all the factors into consideration during similarity measurement. The following five factors are identified and evaluated:

1. **Taxonomic Distance of Functions** Taxonomy defines the hierarchical groups, i.e. taxa, to be referenced for grouping individual items. Taxonomic Distance is a measure of taxonomic similarity between two mathematical terms. In a taxonomy, it is intuitive to assign more similarity to two terms belonging to the same category than to terms belonging to different categories. In our search, terms that belong to the same Content Dictionary (CD)

are attributed a higher similarity value than terms that belong to different CDs.
For future consideration, even within the same Content Dictionary, some finer-granularity hierarchy could be superimposed to further differentiate the functions for the more precise similarity measurement.

2. **Data Type Hierarchical Level** The node of a MathML expression is of a data type, such as a constant number, a variable, a function (e.g. multiplication, log, etc.), or a function of function (e.g. integral, diff, etc.). Different data types contribute different levels of significance to the math expression. To illustrate, here is an example, Query Q: $a+2$, one of the expression E_1: $a+3$, and another expression E_2: $\log_a 2$. Expression E_1 "matches" Q at the function level, while E_2 "matches" Q at the variable and constant level. Intuitively, similarity at the function level is more important than at variable or constant level. Thus E_1 is more similar to Q than E_2 is. By reference to the Common LISP types design, we organize these different data types into a partially ordered hierarchy of types defined by the subset relationship [10]. That is, variables and constants are at the lowest level, function is at the higher level, and function of function is at the highest level. The premise is that the higher the data type is in the hierarchy, the higher the significance of that element is to the whole expression. Note that there are more data type levels in data type which can be considered in future work, but in this work we limit ourselves to two levels: function level, and argument level.

3. **Match-Depth** Naturally each MathML expression is expressed in an XML tree structure. The nodes at the higher level of the MathML expression tree decide how the expression starts, and largely determine the nature of the whole expression. Further down the tree, the nodes depict the characteristics of the expression in more detail and more locality. We claim that the similarity at the higher level matters more than at the lower level. In other words, the more deeply nested the query is in an expression, the less similarity there is between the query and that expression. An example is given in Fig. 1. Tree-wise, Q "matches" E_1 at a higher level in the tree than it does

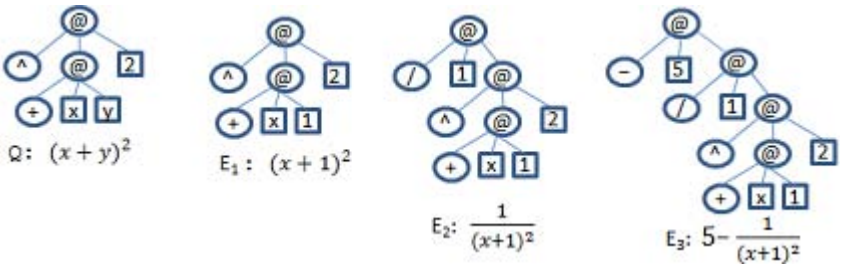

Fig. 1. Illustration of Math Similarity Factor: Depth

to E_2, and Q "matches" E_2 at a higher level in the tree than it does to E_3. Intuitively, E_1 is more similar to Q than E_2 is, and E_2 is more similar to Q than E_3 is. This illustrates that high-level matches correspond to stronger similarity than lower-level matches.

To incorporate the match-depth element into our similarity metrics, we propose to represent match-depth as a similarity-decaying multiplicative factor. It is a decaying factor because the bigger the depth, the smaller the multiplicative factor should be in order to cause the similarity to be smaller. One can utilize different models for this decay factor, such as exponential decay, linear decay, quadratic decay, or constant decay. The different models produce different degrees of penalty for depth difference. As for which model to choose for math similarity search, it depends on the type of application of the math search. For those knowledge discovery oriented math search applications, the structural similarity of math expressions is more important, thus the exponential decay model can be a good choice.

4. **Query Coverage** In actual use, how much of the query expression Q is "covered" in the returned expression E is very important. The following example gives an intuitive illustration: There is a query Q: $(x + y)^2$, an expression E_1: $(x + y)^2 + (x - y)^2$, and another expression E_2: $x + y$. Q is intuitively more similar to E_1 than to E_2. Generally, the higher the query coverage is, the higher the significance is.

5. **Formula vs. Expression** If an expression has at the root level a relational operator (e.g., $=, \geq$), it is treated as a "formula"; otherwise, a "non-formula". Typically in math content, formulas are more significant and more informative than non-formula expressions, and therefore more weight should be given to the former than to the latter.
Note that, strictly speaking, this is not really a similarity factor, instead it is a relevance ranking factor. But it is incorporated into our similarity measure, because our similarity measure is our relevance ranking formula.

This concludes all the factors that are considered for similarity measure. Next a similarity metric is defined to take those five factors into account for math similarity measure.

3.3 Math Similarity Metric

We take parse trees as the primary model representing math expressions, and focus especially on Strict Content MathML parse trees. The notion of similarity between two math expressions will be defined in terms of their corresponding parse trees T_1 and T_2, and the similarity measure between them, denoted $sim(T_1, T_2)$, will be defined and computed recursively based on the height of the Strict Content MathML parse tree as explained next.

1. **For two trees T_1 and T_2 of same height 0** In this case, both trees T_1 and T_2 are singleton leaves, the similarity $sim(T_1, T_2)$ is defined as:

(a) If T_1 and T_2 are constants
   i. $sim(T_1, T_2) = 1$, if $T_1 = T_2$.
   ii. $sim(T_1, T_2) = \delta$, if $T_1 \neq T_2$, where $0 \leq \delta < 1$.
   δ is one of the parameters that are optimized experimentally.

(b) If T_1 and T_2 are variables
   i. $sim(T_1, T_2) = 1$, if $T_1 = T_2$.
   ii. $sim(T_1, T_2) = \epsilon$, if $T_1 \neq T_2$, where $0 \leq \epsilon \leq 1$.
   Because the choice of symbol used for a variable name is immaterial in most cases, ϵ is simply set to 1 as the initial value in our implementation prior to the optimization process. Our research focuses on the context-free evaluation; otherwise, similarity of two variables can depend on not only value, but location and role, which can be an interesting topic for future work.

(c) If T_1 and T_2 are functions, the taxonomic distance is leveraged to measure the similarity between the two functions.
   i. $sim(T_1, T_2) = 1$, if T_1 and T_2 are the same function.
   ii. $sim(T_1, T_2) = \mu$, if T_1 and T_2 are are functions of same category in the taxonomy, where $0 < \mu < 1$. μ is one of the parameters that are optimized experimentally.
   iii. $sim(T_1, T_2) = 0$, if T_1 and T_2 are functions that belong to different categories in the taxonomy.

(d) If T_1 and T_2 belong to different data types
   i. $sim(T_1, T_2) = \theta$, if one tree is a constant and the other is a variable, where $0 \leq \theta < 1$.
   ii. $sim(T_1, T_2) = 0$, if one tree is a function and the other is a constant or variable.

2. **For two trees T_1 and T_2 of same height** $h \geq 1$ In this case, the trees T_1 and T_2 are composed of function apply operator @ as root, a left-most child node representing function, followed by a list of argument nodes which are sub-trees, as illustrated in Fig. 2. Naturally the similarity between T_1 and T_2 is affected by the similarity between the two function node f_1 and f_2, and by the similarity between the two lists of argument nodes. p is the number of argument nodes in T_1, while q is the number of argument nodes in T_2. We treat T_1 as the query expression, T_2 as an expression in the database. Because functions are more important than arguments, the similarity between T_1 and T_2 is defined as a weighted sum:

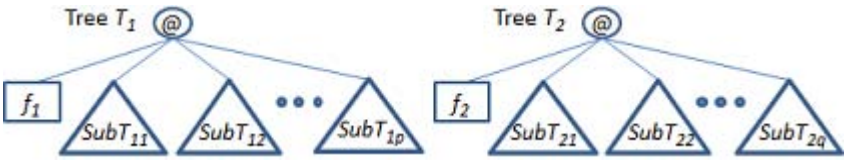

Fig. 2. Illustration of two trees T_1 and T_2 of same height $h \geq 1$

$sim(T_1, T_2) = \alpha \cdot sim(f_1, f_2) + \beta \cdot sim(\{SubT_{11}, SubT_{12}, \ldots, SubT_{1p}\}, \{SubT_{21}, SubT_{22}, \ldots, SubT_{2q}\})$,
where α and β are weighting factors that capture the significance of the similarity contribution from each child node of the tree. Weighting factor $\alpha = \frac{\omega}{p+\omega}$, and $\beta = \frac{1}{p+\omega}$, where ω is boost value for the leftmost child being a function data type as opposed to argument. We take $\omega > 1$. Using p instead of q takes the query coverage factor into account.

The similarity between the two lists of argument nodes,
$sim(\{SubT_{11}, SubT_{12}, \ldots, SubT_{1p}\}, \{SubT_{21}, SubT_{22}, \ldots, SubT_{2q}\})$,
is a compound value,
$0 \leq sim(\{SubT_{11}, SubT_{12}, \ldots, SubT_{1p}\}, \{SubT_{21}, SubT_{22}, \ldots, SubT_{2q}\}) \leq p$.
The measure of the similarity between the two lists of argument nodes depends on the commutative nature of the functions.

(a) If f_1 and f_2 are non-commutative functions, the order of the arguments is observed. The similarity between the two lists is the sum of the similarities between the corresponding available pairs of argument nodes with one from each tree:
$sim(\{SubT_{11}, SubT_{12}, \ldots, SubT_{1p}\}, \{SubT_{21}, SubT_{22}, \ldots, SubT_{2q}\}) = \sum_{i=1}^{min(p,q)} sim(SubT_{1i}, SubT_{2i})$

(b) If f_1 and f_2 are commutative functions, an argument node in T_1 can be paired with any argument node in T_2. To find the best pairing between the 2 lists of argument nodes, the permutations of the argument nodes are taken into consideration, which can be very costly to compute. In this research, we apply the Greedy Approximation algorithm as described in Fig. 3 to find a solution that is close to the optimum similarity value.

```
Greedy_Similarity (T1, T2)
{
  sim(SubT1 , SubT2) = 0;
  P = { i | 1 ≤ i ≤ p };
  Q = { j | 1 ≤ j ≤ q };
  while ( P ≠ Ø and Q ≠ Ø ) {
    sim(SubT1 , SubT2) = sim(SubT1 , SubT2) + max {sim(SubT1i , SubT2j) | i ∈ P, j ∈ Q};
    P = P – { i };
    Q = Q – { j };
  }
  sim (T1 , T2) = (ω·sim(f1 , f2) + sim(SubT1 , SubT2) ) / (p + ω);
  return sim (T1 , T2);
}
```

Fig. 3. Greedy Algorithm to find Similarity of Two Trees with Commutative Functions

In this case, the similarity between the two lists of arguments is defined as:
$sim(\{SubT_{11}, SubT_{12}, \ldots, SubT_{1p}\}, \{SubT_{21}, SubT_{22}, \ldots, SubT_{2q}\})$
$= \max\left\{(\sum_{i=1}^{p}(sim(SubT_{1i}, SubT_{2t(q,p,i)})))\right\}$, where $t(q,p,i)$ is the i-th element of a p-permutation of q.
$\approx \sum_{i=1}^{min(p,q)} \max\ (\ \{sim(SubT_{1i}, SubT_{2\varphi(i)})\}$, by applying greedy approximation, $\varphi(i) = 1, 2, \ldots, q$ and $\varphi(i) \notin \{\varphi(1), \varphi(2), \ldots, \varphi(i-1)\}$.
It is noted that other approximation algorithms can be used to replace the proposed Greedy algorithm for more optimum approximation and/or less computational complexity. This is not to be addressed in this research, but deferred for future research.

(c) If f_1 is commutative function, and f_2 is non-commutative function, or vice versa, we argue that this case should be the same as the above case with both f_1 and f_2 are commutative functions. Because for example, if we have query tree $Q : 5 - 2$, expression $E_1 : 5 + 2$, and expression $E_2 : 2 + 5$, then we should have $sim\ (Q, E_1) = sim\ (Q, E_2)$ which we will not have if we do not "permute" the sub-trees of the tree with commutative function. Thus, in this case, the similarity between the two lists of arguments is defined the same as the case with both functions being commutative. The example in Fig.4 is given for illustration.

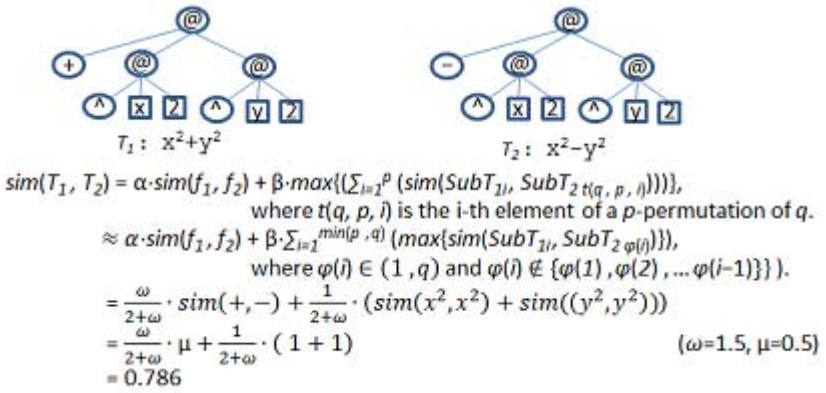

Fig. 4. T_1 with commutative function, and T_2 with non-commutative function

3. **For two trees T_1 and T_2 of different heights** If one of the two trees, say T_1, is of $height(T_1) = h$, and the other tree T_2 is of $height(T_2) \geq h + 1$, then the match between T_1 and T_2 can be at the highest level of T_2, or nested in the T_2, and the best match of these two possibilities is taken. In other words, to measure the similarity between T_1 and T_2, not only the similarity between T_1 and T_2 at their root level is evaluated, but also the similarity between entire tree T_1 and each single sub-tree of T_2, that is $sim(T_1, \text{Sub}T_{2j})$, in this

case because the match is nested, the match-Depth Penalty is applied. Then we choose whichever the larger value as the final similarity measure. Thus, in this case, the similarity between the two trees T_1 and T_2 is defined as shown in Fig 5.

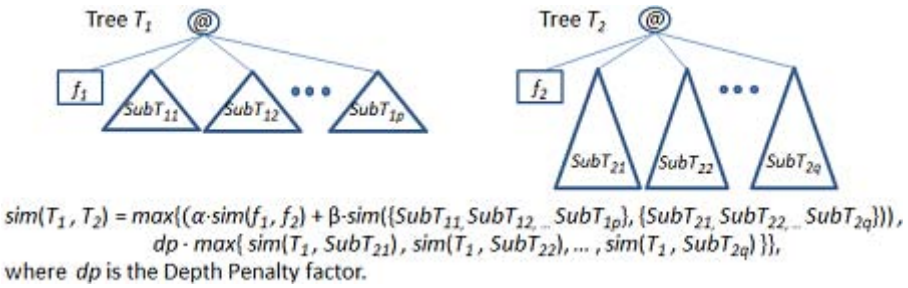

Fig. 5. Similarity Metric for two trees T_1 and T_2 of different heights

We recursively keep comparing the first tree T_1 with the sub-trees of the second tree T_2 , till the two trees under evaluation are of the same height, in which case the similarity metric is already defined.

3.4 Performance Evaluation

To our best knowledge, there is no standard benchmark Strict Content MathML encoded documents set together with a set of standard sample queries that can be used to evaluate MathML search engine's performance. This makes it a challenge to quantitatively compare the performance of versions math similarity metrics as well as various math search engines.

1. **Evaluation Methodology** As the DLMF math digital library and search engine are among the few available and easily accessible, this research leverages the DLMF as the source for mathematical expressions repository, and we compare our similarity search approach to the DLMF search system. To our knowledge there is no Strict Content MathML encoding of the DLMF; therefore, a significant subset of the DLMF is hand-crafted into Strict Content MathML encoding in this research. The methodology of how we build the dataset and evaluate the performance of the proposed similarity metrics is depicted below.

   On the one hand, the queries with varying degrees of mathematical complexity and length were selected. Table 1 lists the test queries that we used. For each query in the test set, we identify the expected relevant expressions from DLMF source repository, and further rank them manually by a group of human experts, which are then named as "ground truth". On the other hand, each query expression is compared with the expressions in the DLMF

Table 1. The Test Queries

Query ID	Query Expression	Query ID	Query Expression
Q1	e^z	Q2	$\tan(z)$
Q3	$\int_a^b f(x)dx$	Q4	$f(z_0) = \frac{1}{2\pi i}\int_c \frac{f(z)}{z - z_0}\, dz$
Q5	$\ln(1+z) = z - \frac{z^2}{2} - \frac{z^3}{2} - \cdots$	Q6	$\frac{d}{dx}\int_a^x f(t)dt = f(x)$
Q7	$\int_c f(z)\, dz = 0$	Q8	$f(z) = c_0 + c_1 z + c_2 z^2 + \cdots$
Q9	$\lim \frac{\sin(z)}{z}$	Q10	$A\cosh(az) + B\sinh(az)$
Q11	$sin^2 x + cos^2 x = 1$	Q12	$\sin(x+y) = \sin x \cos y + \cos x \sin y$
Q13	$\int \sin(x)\, dx = -\cos(x)$	Q14	$\cosh(x) \leq (\frac{\sinh(x)}{x})^3$
Q15	$\sinh(z) = \frac{e^z - e^{-z}}{2}$	Q16	$\delta(x - a)$
Q17	$a^2 + b^2$	Q18	$\frac{a(1-x^n)}{1-x} = a + ax + ax^2 + \cdots + ax^{n-1}$
Q19	$\sum a_j\, b_j \leq (\sum a_j{}^p)^{1/p} (\sum b_j{}^q)^{1/q}$	Q20	$F(x) = \frac{1}{\sqrt{2\pi}}\int f(t)e^{ixt}\, dt$
Q21	$\Gamma(z)$	Q22	$\det[a_{ij}]$
Q23	$a \cdot b = \sum a_j\, b_j$	Q24	$\sqrt{a^2 + b^2}$
Q25	$e^{i\pi} + 1 = 0$	Q26	$e^x < \frac{1}{1-x}$
Q27	$\int \frac{dx}{1 + e^x}$	Q28	$(f * g)(t)$
Q29	$\lim_{x \to \infty} x^a e^{-x}$	Q30	$\frac{d}{dx}\arctan(\sin(x^2))$
Q31	$sin^2 x + cos^2 x$	Q32	$\sin(x + y)$
Q33	$\int \sin(x)\, dx$	Q34	$a + ax + ax^2 + \cdots + ax^{n-1}$
Q35	$\sum a_j\, b_j \leq$	Q36	$\int_c \frac{f(z)}{z - z_0}\, dz$
Q37	$\sin(u) + \sin(v)$	Q38	$\cos(u) + \cos(v)$
Q39	$\frac{d}{dz}\text{arccot}(z)$	Q40	$\lim_{n \to \infty} \left(1 + \frac{1}{n}\right)^n$

repository, and a similarity value is computed with the proposed similarity metric. Afterwards, this list of expressions is ordered by the similarity measurement.

Up to this point, for any given query, there are three hit lists: one from "ground truth", one from DLMF site returned by DLMF search, and another ranked by the proposed similarity metric. In order to quantitatively evaluate the performance, this research proposes to compare the three lists of results to figure out the correlation between the proposed Math-Similarity Search (MSS) result list and the "ground truth" list, and the correlation between the DLMF search result list and the "ground truth" list. Our comparison is done with respect to recall and relevance ranking.

To evaluate the quality of the relevance ranking, the two classical rank correlation coefficient metrics, namely, Kendall's *tau* (τ) and Spearman's *rho* (ρ) are used. In statistics, τ is used to measure the extent of agreement between two lists of measurements, while ρ is the standard correlation coefficient of statistical dependence between two variables. In general, the magnitude of τ is less than the value of ρ. τ focuses more on the relative order of the hits (which came before which), whereas ρ focuses more on absolute order (where each hit ranked). Both metrics are implemented in this research to complement each other in the ranking comparison.

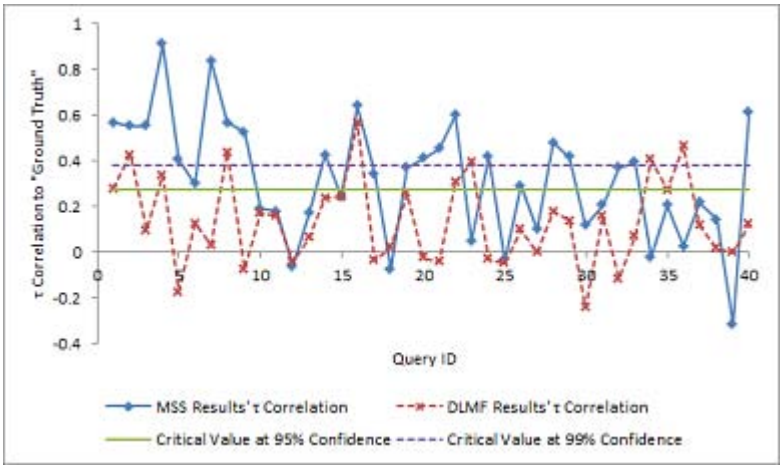

Fig. 6. τ Correlation Analysis of MSS Results vs. DLMF Results over 40 Queries

2. **Performance Evaluation Results** The performance evaluation of the proposed search shows that both the recall and the ranking based on our proposed similarity metric align better with the "ground truth" than that of DLMF search. Figure 6 and Fig. 7 indicate that the search results of most of the 40 queries in our evaluation that are returned by the proposed MSS

search have better correlation to "ground truth" than those of DLMF, with respect to τ metric and ρ metric. That validates the advantage of the proposed MSS over the DLMF search with respect to relevance ranking.

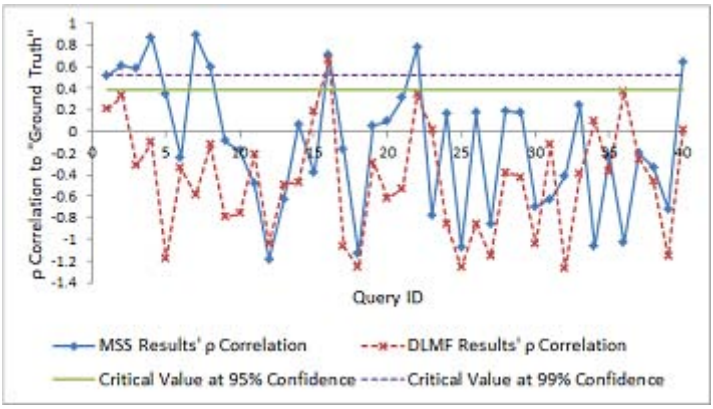

Fig. 7. ρ Correlation Analysis of MSS Results vs. DLMF Results over 40 Queries

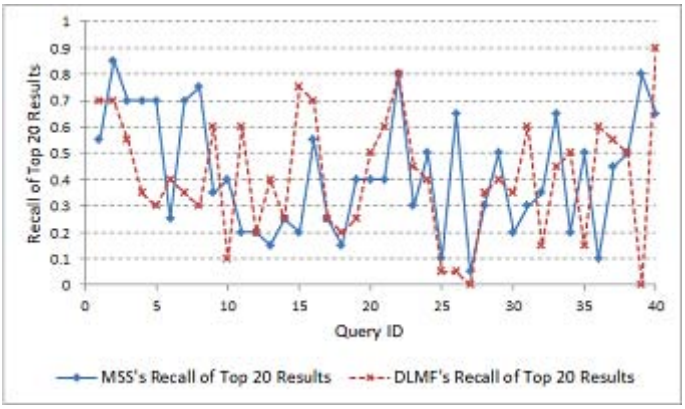

Fig. 8. Recall of MSS Top 20 Results vs. DLMF's Top 20 Results over 40 Queries

Figure 8 and Fig. 9 indicate that with respect to the recall of the top 20 results, the MSS does not differ significantly from the DLMF search. However, with respect to the recall of the top 10 results, the MSS search shows better performance than the DLMF search does.

Fig. 9. Recall of MSS Top 10 Results vs. DLMF's Top 10 Results over 40 Queries

4 Conclusions and Future Work

In order to effectively and efficiently find math expressions that are similar to a user's query, this paper conceptualizes math similarity between mathematical expressions with more weight to structural similarity and mathematical semantics than the mere notational matching that many existing math search solutions focus on. Further, this paper proposes a semantic-sensitive math-similarity metric to measure the math similarity. With the availability of Strict Content MathML which represents math in disambiguated uniform structure, an algorithm is developed to compute the math similarity between any two Strict Content MathML encoded math expressions. Additionally, a "ground truth" of math queries and corresponding matching expressions is constructed by leveraging DLMF, and is used as a benchmark for performance evaluation. Comparing with the existing non-similarity based math search techniques, primarily the DLMF math search, the proposed math-similarity search does show the performance advantage with respect to both recall and relevance ranking.

However, many parameters of the proposed similarity metric are yet to be optimized, including taxonomic distance values (e.g. μ, θ) between functions, function nodes type booster value ω, depth penalty decay model and its parameters, query coverage factor, etc. We plan to address them in the near future. Other future directions include: (1) Address normalization in the context of Strict Content MathML. (2) Cover in the similarity search the remaining elements of Strict Content MathML that are not covered in this research, such as "*m:bind*" and "*m:share*". (3) Leverage the sample queries and benchmark dataset that are to be produced from the NTCIR-11 [9] ongoing math task, for more thorough and more objective performance evaluation.

References

1. The Digital Library of Mathematical Functions (DLMF), the National Institute of Standards and Technology (NIST), http://dlmf.nist.gov/
2. Hashimoto, H., Hijikata, Y., Nishida, S.: Incorporating Breadth First Search for Indexing MathML Objects. In: IEEE International Conference on Systems, Man and Cybernetics, SMC 2008 (2008)
3. Kamali, S., Tompa, F.: A New Mathematics Retrieval System. In: Proceedings of the 19th ACM International Conference on Information and Knowledge Management, CIKM 2010. ACM, New York (2010)
4. Kohlhase, M., Sucan, I.: A Search Engine for Mathematical Formulae. In: Calmet, J., Ida, T., Wang, D. (eds.) AISC 2006. LNCS (LNAI), vol. 4120, pp. 241–253. Springer, Heidelberg (2006)
5. Libbrecht, P., Melis, E.: Methods to Access and Retrieve Mathematical Content in ActiveMath. In: Iglesias, A., Takayama, N. (eds.) ICMS 2006. LNCS (LNAI), vol. 4151, pp. 331–342. Springer, Heidelberg (2006)
6. Miller, B., Youssef, A.: Technical Aspects of the Digital Library of Mathematical Functions. Annals of Mathematics and Artificial Intelligence 38(1-3), 121–136 (2003)
7. Miner, R., Munavalli, R.: An Approach to Mathematical Search Through Query Formulation and Data Normalization. In: Kauers, M., Kerber, M., Miner, R., Windsteiger, W. (eds.) MKM/Calculemus 2007. LNCS (LNAI), vol. 4573, pp. 342–355. Springer, Heidelberg (2007)
8. Miutka, J., Galambo, L.: Mathematical Extension of Full Text Search Engine Indexer. In: Proceedings of Information and Communication Technologies: From Theory to Applications, ICTTA 2008, IEEE Catalog number CFP08577, Syria, pp. 207–208 (2008)
9. The 11th National Institute of Informatics Testbeds and Community for Information access Research Workshop (2013-2014), http://ntcir-math.nii.ac.jp/
10. Reddy, A.: Features of Common Lisp (2008), http://random-state.net/features-of-common-lisp.html
11. Sojika, P., Liška, M.: The Art of Mathematics Retrieval. In: Proceedings of the ACM Conference on Document Engineering, DocEng 2011, Mountain View, CA, pp. 57–60 (2011)
12. Mathematical Markup Language (MathML) Version 3.0 (3rd edn.), World Wide Web Consortium, http://www.w3.org/TR/MathML3/
13. Yokoi, K., Aizawa, A.: An Approach to Similarity Search for Mathematical Expressions using MathML. In: Towards a Digital Mathematics Library, Grand Bend, Ontanrio, Canada, pp. 27–35. Masaryk University Press, Brno (2009)
14. Youssef, A.: Information Search and Retrieval of Mathematical Contents: Issues and Methods. In: The ISCA 14th Int'l Conf. on Intelligent and Adaptive Systems and Software Engineering (IASSE 2005), Toronto, Canada, July 20-22 (2005)
15. Youssef, A.S.: Methods of Relevance Ranking and Hit-content Generation in Math Search. In: Kauers, M., Kerber, M., Miner, R., Windsteiger, W. (eds.) MKM/Calculemus 2007. LNCS (LNAI), vol. 4573, pp. 393–406. Springer, Heidelberg (2007)