Mathematical Equation Structural Syntactical Similarity Patterns: A Tree Overlapping Algorithm and Its Evaluation

Evgeny Pyshkin
University of Aizu, Japan
Tsuruga, Ikki-machi, Aizu-Wakamatsu
Fukushima, 965-8580 Japan
E-mail: pyshe@u-aizu.ac.jp

Mikhail Ponomarev
Peter the Great St. Petersburg Polytechnic University
29 Polytechnicheskaya st., 195251 St. Petersburg, Russia
E-mail: ponmike92@gmail.com

Keywords: syntactical similarity, mathematical equations, MathML, tree overlapping

Received: November 14, 2016

In this paper we examine mathematical equations structural syntactical similarity patterns. The major focus of this contribution is an NLP tree overlapping algorithm modification adopted to the case of syntactical similarity of mathematical equations presented in MathML. We describe the software implementation and the tests arranged for the cases of both structural and subexpression based similarity. The paper also contains a discussion of algorithm evaluation problems conditioned by the lack of relevant syntactical similarity centered equation corpora.

Povzetek: Prispevek se ukvarja s strukturnimi sinkaktičnimi vzorci podobnosti matematičnih izrazov.

1 Introduction

Nowadays there are only few models of adopting natural language processing (NLP) algorithms related to syntax similarity to mathematical notations. Unique structural syntax of mathematical equations with a big variety of semantically equivalent constructions provide a non-trivial case for information retrieval [11]. Many reported implementations are focused on finding exact matching of mathematical constructions rather than on recognizing their similarity [9, 8]. Indeed, for a case of mathematical equations, syntactical similarity is defined rather fuzzy by using several structural syntactical similarity patterns. However, a model that would deal with syntactical similarity seems to be very useful while developing searching and classification tools used in education, so as to allow math learners and tutors selecting suitable tasks nailing down a topic presented during a classroom session. An obvious possible use case is accessing a set of relevant mathematical equations to be used for training while a learner is doing the preparation exercises for an examination. Another interesting possibility is searching an equation by its syntactical structure, the latter being often easier to recall compare to exact mathematical formulas.

For the reason that most structural notations used for representing mathematical expressions are in fact based on directed graphs, the syntactical similarity can be defined by using tree structural similarity. Specifically, this work addresses the case when expressions are uniformly presented in MathML, the latter being one of widely used structural XML based notations used in mathematics. In turn, if better structural math equation forms are used, one can expect more efficient and accurate retrieval, in contrast to a frequent use of image based equation representation used on many web sites. At the same time we accept a possible criticism pointing an issue that not an every mathematical expression retrieval difficulty could be addressed under MathML representability constraints.

Within a context of mathematical equations similarity, it is important to mention both meaning related similarity and presentation related similarity (in MathML there are two markup schemes corresponding to these two perspectives: content markup and presentation markup). Semantic (e.g. equation meaning) similarity is out of scope of this study; this work is focused only on presentation similarity which is related to the equation syntax (the form) rather than to its meaning (the contents).

This paper is based on Mikhail Ponomarev and Evgeny Pyshkin, *Adopting Tree Overlapping Algorithm for MathML Equation Structural Similarity*, published in the Proceedings of the 2nd International Conference on Applications in Information Technology (ICAIT-2016) [7]

MathML – Mathematical Markup Language

2 Structural similarity of mathematical equations

Since the structure of mathematical equations can be represented in the form of a tree, mathematical equation similarity may be defined using tree similarity. In [1] similarity of two trees is defined on the base of recursive examination of their subtrees. In [5] the following mathematical expressions similarity patterns are defined:

Mathematical equivalence: Equations E_1 and E_2 are mathematically equivalent if they are semantically (but not obligatorily syntactically) the same, for example $\frac{d(sin(x))}{dx}$ and $(sin(x))'$, $sin^2(x) + cos^2(x)$ and 1 are correspondingly equivalent.

Identity: E_1 and E_2 are identical if they are exactly the same.

Syntactical identity: E_1 and E_2 are syntactically identical if they are identical after normalization (dealing with variable names and numeric values). For example $sin(a)$ and $sin(b)$, $\frac{1}{sin(x)}$ and $\frac{5}{sin(x)}$ are correspondingly syntactically identical.

N–similarity: Normalized equations E_1 and E_2 are n-similar if there is a similarity (in a certain sense) which is $sim(E_1, E_2) \geqslant n$, n being a parametric value determining a threshold. There are two specific N–similarity cases:

1. **Subexpression n–similarity**: There is a subexpression n–similarity for E_1 and E_2, if E_1 and E_2 are n–similar and the corresponding trees both contain the common subtree which in turn contains all the terminal nodes of both trees. Figure 1 shows an example for the case of expressions $sin(x)^2$ and $\frac{sin(x)}{2}$.

2. **Structural n–similarity**: E_1 and E_2 are structurally n–similar if E_1 and E_2 are n–similar (in common sense) and there is a common part in both trees rooted at root nodes of compared trees with the production rules being the same for all the nodes in this part. Figure 2 illustrates this case for the equations $x + \sqrt{sin(a)}$ and $x + \sqrt{2b}$.

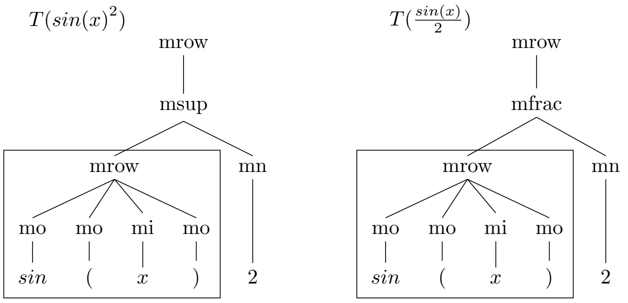


Figure 1: Equations $sin(x)^2$ and $\frac{sin(x)}{2}$ are structurally n–similar for any value of $n \geqslant \frac{18}{26}$.

Note that in order to represent n-values, in Figures 1 and 2 we use the nodes ratio which is a ratio of the number of common nodes in both trees to the number of all nodes in both trees.

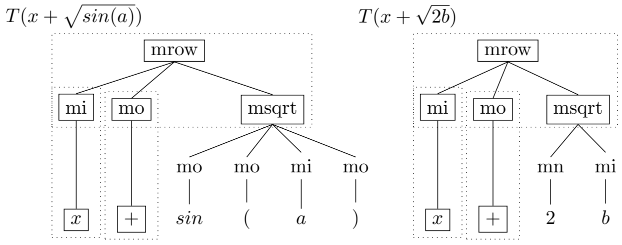


Figure 2: $x + \sqrt{sin(a)}$ and $x + \sqrt{2b}$ are structurally n–similar for any value of $n \geqslant \frac{12}{24}$.

3 Equation similarity evaluation using tree similarity

In order to introduce different approaches used for evaluating mathematical equation similarity based on tree similarity, in this section some sample trees are used: T_0, T_1 and T_2 (shown in Figure 3). In the following text we demonstrate how the similarity between T_0 and T_1 as well as between T_0 and T_2 respectively can be calculated.

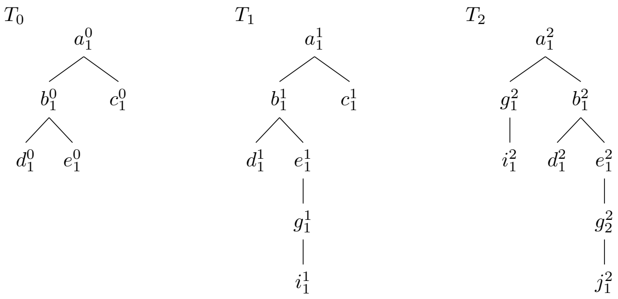


Figure 3: Sample trees T_0, T_1 and T_2.

For each node in Figure 3 its upper index corresponds to the tree which this node belongs to. Thus, all the nodes of T_0 have the upper indexes which are equal to 0, the nodes of T_1 have the upper indexes which are equal to 1, and so on. The lower indexes are used so as to count the equal nodes within a tree. For instance, T_2 contains two equal nodes g, so the first appearance of g is marked by the lower index 1, while the second appearance of g is marked by the lower index 2. These indexes are suppressed in cases, when they are not necessary for algorithm description.

3.1 Tree edit distance

Tree edit distance method uses the definition of similarity (distance) between two trees as a weighted number of edit operations (insert, delete, and modify) required to transform one tree to another (as described, for example, in [10]).

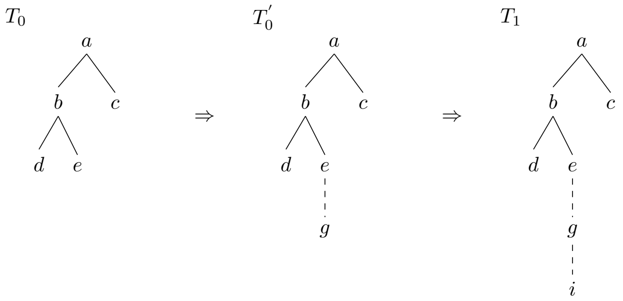


Figure 4: Tree edit transformation: T_0 to T_1.

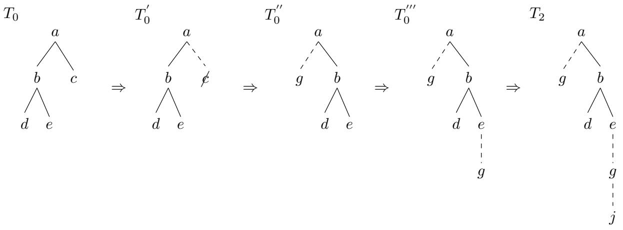


Figure 5: Tree edit transformation: T_0 to T_2.

Assume S is a sequence of edit operations $\{s_1, s_2, \ldots, s_k\}$ for transforming one tree to another. Assume γ is a non-negative distance measure describing node transformation from a to b (defined here as $a \to b$) such as $\gamma(a \to b) \geqslant 0$ and $\gamma(a \to b) = \gamma(b \to a)$.

For an operation sequence S we get the following sum: $\gamma(S) = \sum_{i=1}^{|S|} \gamma(s_i)$.

Then the distance between two equations is defined as follows:

$$\delta(T_1, T_2) = min\{\gamma(S)\} \qquad (1)$$

Figures 4 and 5 illustrate how the transformation distances from T_0 to T_1 and from T_0 to T_2 correspondingly are calculated: a sequence of two insertions is required in order to transform T_0 to T_1; four operations (one deletion and three insertions) are required in order to transform T_0 to T_2.

The node weights (and the operation costs as well) might not be equal, hence there might be different similarity measures based on general edit distance schema. Also, in different algorithms a sequence S_i is searched differently: in addition to the earlier mentioned work [10] there are other implementations described in [2] and [6]. Algorithmic complexity of the above mentioned approaches is summarized in Table 1.

Table 1: Tree edit distance based algorithms.

Algorithm	Time	Memory	Particularities
TED [10]	$O(n^4)$	$O(n^2)$	Good for balanced trees
ODTED [2]	$O(n^3)$	$O(n^2)$	Better in unbalanced trees
RTED [6]	$O(n^3)$	$O(n^2)$	Tree balance insensitive

3.2 Subpath set

Subpath set similarity between two trees is defined as the number of subpaths shared by the trees. Given a tree, its subpaths are defined as a set of all the paths from the root node to the leaves including the partial paths. Subpath based similarity definition for a case of natural language processing can be found in [4]. A possible application of this concept to a case of MathML equations can be illustrated by the algorithm described in [9].

Table 2: Indexing table for T_1 and T_2.

p	$I[p]$	p	$I[p]$
a	$\{1,2\}$	$e \to g$	$\{1,2\}$
b	$\{1,2\}$	$g \to i$	$\{1,2\}$
c	$\{1\}$	$a \to g$	$\{2\}$
d	$\{1,2\}$	$g \to j$	$\{2\}$
e	$\{1,2\}$	$a \to b \to d$	$\{1,2\}$
g	$\{1,2\}$	$a \to b \to e$	$\{1,2\}$
i	$\{1,2\}$	$b \to e \to g$	$\{1,2\}$
j	$\{2\}$	$e \to g \to i$	$\{1\}$
$a \to b$	$\{1,2\}$	$a \to g \to i$	$\{2\}$
$a \to c$	$\{1\}$	$e \to g \to j$	$\{2\}$
$b \to d$	$\{1,2\}$	$a \to b \to e \to g$	$\{1,2\}$
$b \to e$	$\{1,2\}$	$b \to e \to g \to i$	$\{1\}$
		$b \to e \to g \to j$	$\{2\}$

Figure 6 shows a set of common subpaths in the sample trees T_0 and T_1. Hence, in this example subpath based tree similarity $S_s(T_0, T_1) = 11$. Figure 7 illustrates the same issue for a case of the sample trees T_0 and T_2. Similarly, subpath based tree similarity can be calculated as $S_s(T_0, T_2) = 9$.

A more practical case is a set TS of trees to be processed in order to find those which are similar to some given tree T_0. Concerning efficiency issues, a significant improvement may be achieved if, instead of computing the similarity for many pairs, an indexing table $I[p]$ for the whole TS corpus is used [4].

Table 2 provides an example of creating the indexing table for a case of the set of trees consisting of only two trees T_1 and T_2 (shown in Figure 3).

In Table 2 p-columns list all the subpaths from both T_1 and T_2 (i.e. from all the corpus trees); for every subpath the corresponding $I[p]$-cell shows a list of trees where such a subpath exists. Algorithmic complexity of an indexing table based algorithm is $O(L * D^2)$, where L – maximal number of tree leaves, D – maximal tree depth among all the corpus trees [4].

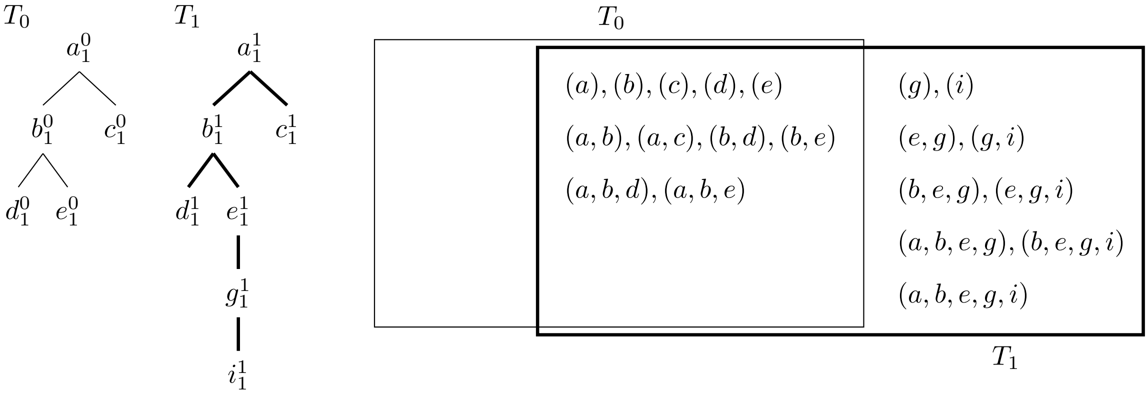


Figure 6: Subpaths in T_0 and T_1.

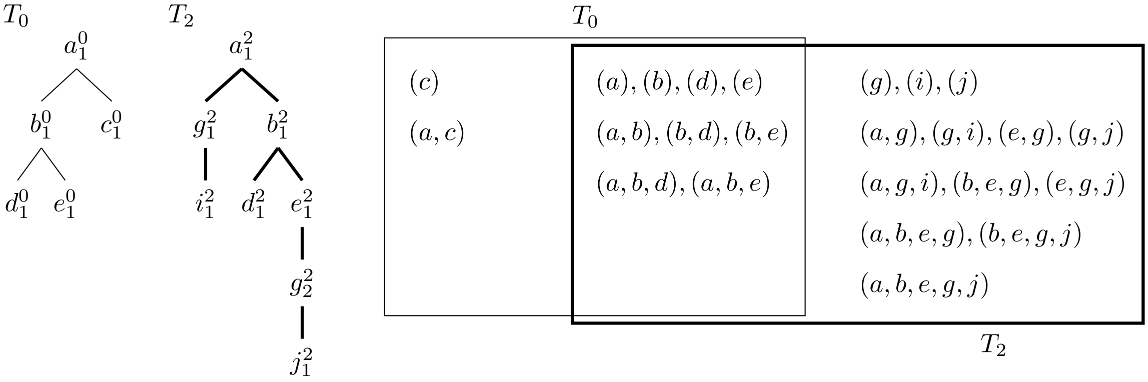


Figure 7: Subpaths in T_0 and T_2.

3.3 Tree overlapping algorithm and its modification for math equation structural similarity

A basic tree overlapping algorithm is described in [4] for a case of sentence similarity which is defined as follows. When putting an arbitrary node n_1 of a tree T_1 on a node n_2 of a tree T_2, there might be the same production rule overlapping in T_1 and T_2. Similarity is defined as a number of such overlapping production rules.

In contrast to the base algorithm from [4] where tree terminals are naturally excluded, for a case of mathematical equations we also include terminal nodes as if they had the same production rules (*Relaxation 1*). Also we relax the strictness of the base algorithm and include the pairs of corresponding nodes which are in the same order among their siblings but do not obligatorily have the same production rules for their child nodes (*Relaxation 2*). Below there is a formal definition of our modification.

Assume $L(n_1, n_2)$ represents a set of overlapping node pairs when putting n_1 on n_2. Assume $ch(n, i)$ is i-th child of node n. The set $L(n_1, n_2)$ is being generated by applying the following rules:

1. $(n_1, n_2) \in L(n_1, n_2)$
2. If $(m_1, m_2) \in L(n_1, n_2)$, then $(ch(m_1, i), ch(m_2, i)) \in L(n_1, n_2)$
3. $L(n_1, n_2)$ includes all the pairs generated recursively by the rule No. 2.

A number $N_{TO}(n_1, n_2)$ of production rules (according to the *Relaxation 1*) is defined as follows:

$$N_{TO}(n_1, n_2) = \left\{ (m_1, m_2) \;\middle|\; \begin{array}{l} m_1 \in nodes(T_1) \\ \wedge\ m_2 \in nodes(T_2) \\ \wedge\ (m_1, m_2) \in L(n_1, n_2) \\ \wedge\ PR(m_1) = PR(m_2) \end{array} \right\} \qquad (2)$$

In equation 2 $nodes(T)$ is a set of nodes (including terminals) in a tree T, while $PR(n)$ is a production rule rooted at the node n.

Figure 8 shows an example of overlapping tree modification algorithm for $N_{TO}(d_1, d_2) = \{(d^1, d^2), (f^1, f^2), (g^1, g^2)\}$.

Assume $P_{WPR}(n_1, n_2)$ is a set of nodes which is represented as a path from (n_1, n_2) to the top last pair of nodes being in the same order among their siblings. Assume n_i and m_i are nodes of a tree T_i, $ch(n, i)$ is i-th child of node n. According to the *Relaxation 2*, P_{WPR} is defined as follows:

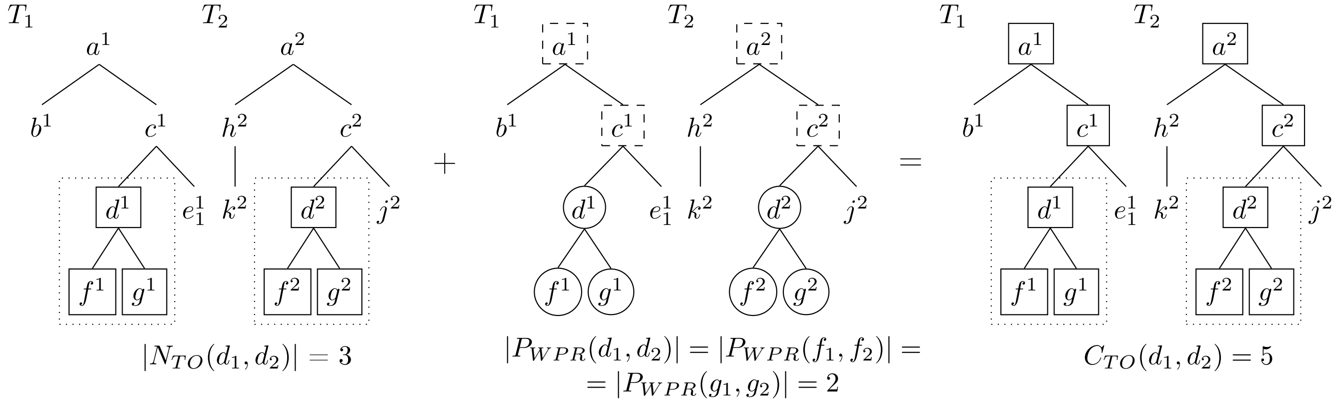

T_1
T_2
a^1 b^1 c^1 d^1 e_1^1 f^1 g^1
a^2 h^2 c^2 k^2 d^2 j^2 f^2 g^2
+
=
$|N_{TO}(d_1, d_2)| = 3$
$|P_{WPR}(d_1, d_2)| = |P_{WPR}(f_1, f_2)| =$
$= |P_{WPR}(g_1, g_2)| = 2$
$C_{TO}(d_1, d_2) = 5$

Figure 8: Modified tree-overlapping algorithm: example.

1. $(n_1, n_2) \notin P_{WPR}$
2. If $PR(parent(n_1)) \neq PR(parent(n_2))$
   $\wedge\, ch(parent(n_1), i) = ch(parent(n_2), i)$
   $\wedge\, ch(parent(n_1), i) = n_1$
   $\wedge\, h(parent(n_2), i) = n_2$,
   $(parent(n_1), parent(n_2)) \in P_{WPR}$
3. $P_{WPR}(n_1, n_2)$ includes only pairs generated by applying rule No. 2.

Then the second component for an integral similarity measure can be defined by using the above introduced P_{WPR} as follows:

$$P_{TO}(n_1, n_2) =$$

$$\left\{ (m_1, m_2) \left| \begin{array}{l} (p_1, p_2) \in N_{TO}(n_1, n_2) \\ (m_1, m_2) \in P_{WPR}(p_1, p_2), \\ \qquad \text{if} \quad top(m_1, m_2) = (n_1, n_2) \end{array} \right. \right\} \tag{3}$$

In equation 3 $top(n_1, n_2)$ is the last pair in set $P_{WPR}(n_1, n_2)$: $top(n_1, n_2) = p_{last}(n_1, n_2)$, $p_{last} \in P_{WPR}$.

Thus, for two nodes, the resulting combined similarity measure is defined as follows:

$C_{TO}(n_1, n_2) = |N_{TO}(n_1, n_2)| + |P_{TO}(n_1, n_2)|$

For the whole trees, we get:

$$S_{TO}(T_1, T_2) = \max_{n_1 \in nodes(T_1), n_2 \in nodes(T_2)} C_{TO}(n_1, n_2) \tag{4}$$

3.4 Software implementation

We developed a software prototype in order to arrange a series of experiments for the above described modification of the tree overlapping algorithm for a case of mathematical equations. Figure **??** gives a hint of how the application user interface is organized.

For displaying mathematical equations defined in MathML the library *net.sourceforge.jeuclid* is used.

4 Experiments

One of the problems we faced while attempting to evaluate the algorithm is that, unlike to the NLP domain, there is no substantial corpus of mathematical equation syntactical similarity classes.

4.1 Test Corpora

For our rather preliminary analysis several experts experienced in teaching mathematics in high schools and lyceums were involved. With their help we selected a number of typical trigonometry problems from the set of tasks used in Russian Unified State Examination [3] The selected equations are listed in Table 3.

With the help of our experts, the expressions were classified according their structural similarity. As a result, two types of equation classification were created: a classification based on equation structural similarity (see Table 4) and a classification based on subexpression similarity (see Table 5).

4.2 Tests

Though corpora presented in Tables 4 and 5 aren't representative enough, they make possible to proceed with some preliminary similarity precision estimation. Let us remind that precision is defined as follows:

$$precision = \frac{TP}{TP + FP} \tag{5}$$

In our tests we assume that in equation 5 TP is the number of k first true positive equations belonging to the same class as the query expression, while FP is the number of t first false positive equations: $k + t = n - 1$, where n is the number of equations belonging to the respective class. The preliminary experiments described in this work may be considered a prove-of-concept example for investigating further necessary improvements of the developed algorithm. In the future tests a standard cross-fold validation

Table 3: Base of test equations.

No.	Equation	No.	Equation
1	$\sqrt{2}\sin(\frac{3\pi}{2} - x)\sin x = \cos x$	16	$2\cos(\frac{\pi}{2} + x) = \sqrt{3}\tan x$
2	$\cos(\frac{\pi}{2} + 2x) = \sqrt{2}\sin x$	17	$\sin 2x + 2\sin^2 x = 0$
3	$2\cos(x - \frac{11\pi}{2})\cos x = \sin x$	18	$2\sin(\frac{7\pi}{2} - x)\sin x = \cos x$
4	$2\sin^4 x + 3\cos 2x + 1 = 0$	19	$2\sin^2 x - \sqrt{3}\sin 2x = 0$
5	$(2\cos x + 1)(\sqrt{-\sin x} - 1) = 0$	20	$\cos 2x - 3\cos x + 2 = 0$
6	$(2\sin x - 1)(\sqrt{-\cos x} + 1) = 0$	21	$2\cos^3 x - \cos^2 x + 2\cos x - 1 = 0$
7	$4\sin^4 2x + 3\cos 4x - 1 = 0$	22	$\cos 2x + 3\sin x - 2 = 0$
8	$\cos 2x = \sin(x + \frac{\pi}{2})$	23	$\sin 2x + \sqrt{2}\sin x = 2\cos x + \sqrt{2}$
9	$2\sqrt{3}\cos^2(\frac{3\pi}{2} + x) - \sin 2x = 0$	24	$3\cos 2x - 5\sin x + 1 = 0$
10	$\cos^2 x - \frac{1}{2}\sin 2x + \cos x = \sin x$	25	$\cos 2x - 5\sqrt{2}\cos x - 5 = 0$
11	$\cos 2x = 1 - \cos(\frac{\pi}{2} - x)$	26	$-\sqrt{2}\sin(-\frac{5\pi}{2} + x)\sin x = \cos x$
12	$\sqrt{\cos^2 x - \sin^2 x}(\tan 2x - 1) = 0$	27	$\frac{2\sin^2 x - \sin x}{2\cos x - \sqrt{3}} = 0$
13	$\tan x + \cos(\frac{3\pi}{2} - 2x) = 0$	28	$\frac{2\sin^2 x - \sin x}{2\cos x + \sqrt{3}} = 0$
14	$\cos x + \cos(\frac{\pi}{2} + 2x) = 0$	29	$4\cos^4 x - 4\cos^2 x + 1 = 0$
15	$\frac{1}{2}\sin 2x + sin^2 x - \sin x = \cos x$	30	$4\sin^2 x + 8\sin(\frac{3\pi}{2} + x) + 1 = 0$

Table 4: Structural Similarity Classification.

No.	Expression	Class
1	$\sqrt{2}\sin(\frac{3\pi}{2} - x)\sin x = \cos x$	①
2	$2\cos(x - \frac{11\pi}{2})\cos x = \sin x$	
3	$2\sin(\frac{7\pi}{2} - x)\sin x = \cos x$	
4	$-\sqrt{2}\sin(-\frac{5\pi}{2} + x)\sin x = \cos x$	
5	$\cos 2x - 3\cos x + 2 = 0$	②
6	$\cos 2x + 3\sin x - 2 = 0$	
7	$3\cos 2x - 5\sin x + 1 = 0$	
8	$\cos 2x - 5\sqrt{2}\cos x - 5 = 0$	
9	$\cos(\frac{\pi}{2} + 2x) = \sqrt{2}\sin x$	③
10	$\cos 2x = \sin(x + \frac{\pi}{2})$	
11	$2\cos(\frac{\pi}{2} + x) = \sqrt{3}\tan x$	
12	$2\sin^4 x + 3\cos 2x + 1 = 0$	④
13	$4\sin^4 2x + 3\cos 4x - 1 = 0$	
14	$4\cos^4 x - 4\cos^2 x + 1 = 0$	
15	$(2\cos x + 1)(\sqrt{-\sin x} - 1) = 0$	⑤
16	$(2\sin x - 1)(\sqrt{-\cos x} + 1) = 0$	
17	$\sqrt{\cos^2 x - \sin^2 x}(\tan 2x - 1) = 0$	
18	$\cos^2 x - \frac{1}{2}\sin 2x + \cos x = \sin x$	⑥
19	$\frac{1}{2}\sin 2x + sin^2 x - \sin x = \cos x$	
20	$\tan x + \cos(\frac{3\pi}{2} - 2x) = 0$	⑦
21	$\cos x + \cos(\frac{\pi}{2} + 2x) = 0$	
22	$\frac{2\sin^2 x - \sin x}{2\cos x - \sqrt{3}} = 0$	⑧
23	$\frac{2\sin^2 x - \sin x}{2\cos x + \sqrt{3}} = 0$	

procedure will be required in order to get trustworthy precision evaluation results.

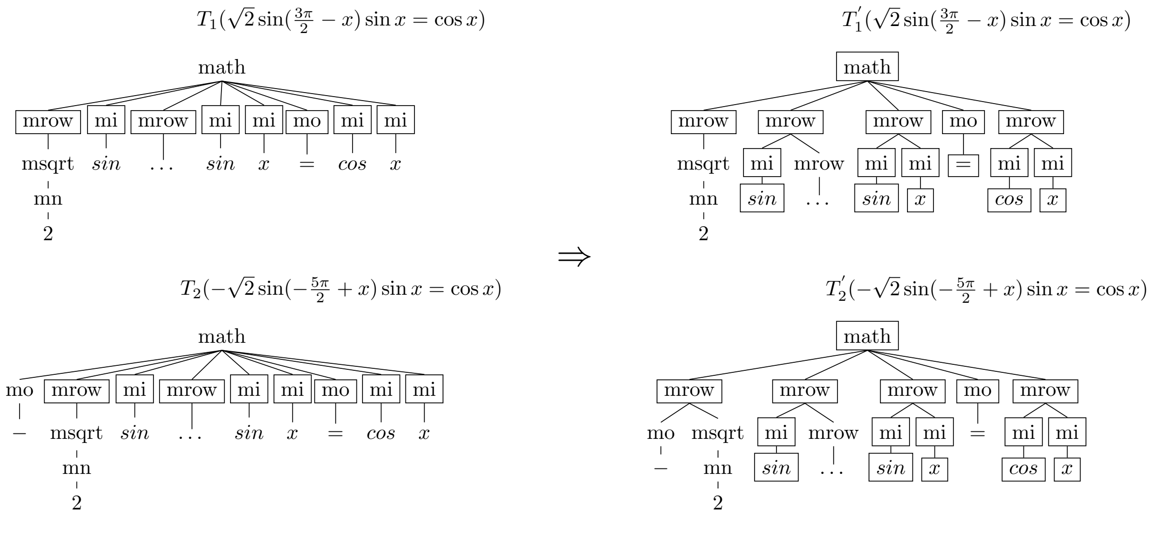
Table 5: Subexpression Based Similarity.

No.	Expression	Class
1	$\cos(\frac{\pi}{2} + 2x) = \sqrt{2}\sin x$	①
2	$\cos x + \cos(\frac{\pi}{2} + 2x) = 0$	
3	$(2\cos x + 1)(\sqrt{-\sin x} - 1) = 0$	②
4	$(2\sin x - 1)(\sqrt{-\cos x} + 1) = 0$	
5	$\sqrt{2}\sin(\frac{3\pi}{2} - x)\sin x = \cos x$	③
6	$2\sin(\frac{7\pi}{2} - x)\sin x = \cos x$	
7	$\frac{2\sin^2 x - \sin x}{2\cos x - \sqrt{3}} = 0$	④
8	$\frac{2\sin^2 x - \sin x}{2\cos x + \sqrt{3}} = 0$	
9	$2\sin^4 x + 3\cos 2x + 1 = 0$	⑤
10	$4\cos^4 x - 4\cos^2 x + 1 = 0$	
11	$\cos^2 x - \frac{1}{2}\sin 2x + \cos x = \sin x$	
12	$\frac{2\sin^2 x - \sin x}{2\cos x - \sqrt{3}} = 0$	
13	$\frac{2\sin^2 x - \sin x}{2\cos x + \sqrt{3}} = 0$	

Figure 10 illustrates the process of structural similarity computation for two expressions from the tiny corpus described earlier. The first expression consists of 34 nodes while the second one has 33 nodes. 20 nodes are equal in both trees. So, $S_{TO} = \frac{20+20}{34+33} = \frac{40}{67} = 0.597$.

4.3 Analysis

Table 6 lists 5 expressions from the base defined in Table 3 which achieve the best scores for the query expres-


$T_1(\sqrt{2}\sin(\frac{3\pi}{2}-x)\sin x = \cos x)$
$T_1'(\sqrt{2}\sin(\frac{3\pi}{2}-x)\sin x = \cos x)$
$T_2(-\sqrt{2}\sin(-\frac{5\pi}{2}+x)\sin x = \cos x)$
$T_2'(-\sqrt{2}\sin(-\frac{5\pi}{2}+x)\sin x = \cos x)$
math
mrow
mi
mo
msqrt
mn
2
sin
cos
x
=
−
...
⇒

$$S_{TO}(T_1, T_2) = \frac{0+0}{34+38} = \frac{0}{72} = 0 \qquad S_{TO}(T_1', T_2') = \frac{17+17}{37+41} = \frac{34}{78} = 0.44$$

Figure 9: Tree structure normalization to avoid a false negative case.

sion $\sqrt{2}\sin(\frac{3\pi}{2}-x)\sin x = \cos x$ (belonging to the class 1 according to Table 4).

Table 6: Query: $\sqrt{2}\sin(\frac{3\pi}{2}-x)\sin x = \cos x$.

Compared expression	Nodes ratio	Similarity
$2\sin(\frac{7\pi}{2}-x)\sin x = \cos x$	60/67	0.896
$2\cos(x-\frac{11\pi}{2})\cos x = \sin x$	40/67	0.597
$2\cos(\frac{\pi}{2}+x) = \sqrt{3}\tan x$	24/63	0.381
$3\cos 2x - 5\sin x + 1 = 0$	12/60	0.200
$\tan x + \cos(\frac{3\pi}{2}-2x) = 0$	10/67	0.149

Two best scores are for the equations which also belong to the same class 1, unlike to the equation $-\sqrt{2}\sin(-\frac{5\pi}{2}+x)\sin x = \cos x$ (No. 4 in Table 4) which wasn't recognized as a similar expression despite of its obvious similarity. To explain this phenomenon we have to go back to MathML equation structure. As you can see from Figure 9 (left side), two compared equations (both belonging to the class 1 of the corpus) have rather similar structure (at least, from human point of view). However, their tree roots have different number of child nodes, hence their production rules are (formally) different too. It means that we have to enhance equation normalization factor (currently limited by only variable names and numerical values): in the above mentioned case the issue can be resolved by restructuring a tree based equation representation as Figure 9 (right side) shows: both trees in the right side are semantically equivalent to those which are in the left side. After such a normalization, syntactical similarity score increases from 0 (in the "left" case) to 0.44 (in the "right" case).

Similar tests were arranged for expressions from other classes as well as for the case of subexpression similarity. Specifically, for a case of subexperssion similarity, Table 7 lists 5 best results for the query $2\sin^4 x + 3\cos 2x + 1 = 0$ (which belongs to the class 5 according the the test corpus from Table 5) against the equations from the base defined in Table 3. In Table 7, nodes ratio means a ratio of common nodes to all nodes in compared trees.

Table 7: Query: $2\sin^4 x + 3\cos 2x + 1 = 0$.

Compared expression	Nodes ratio	Similarity
$4\cos^4 x - 4\cos^2 x + 1 = 0$	10/59	0.169
$4\sin^4 2x + 3\cos 4x - 1 = 0$	10/60	0.167
$\cos^2 x - \frac{1}{2}\sin 2x + \cos x = \sin x$	10/63	0.159
$\frac{2\sin^2 x - \sin x}{2\cos x - \sqrt{3}} = 0$	10/64	0.156
$\frac{2\sin^2 x - \sin x}{2\cos x + \sqrt{3}} = 0$	10/64	0.156

Among 5 best results listed in Table 7, only the second equation $4\sin^4 2x + 3\cos 4x - 1 = 0$ doesn't belong to the class 5 (according to the experts' classification). Let us analyze a possible reason. The experts didn't include this equation to the class 5 due to the difference between $sin^4 2x$ and $\sin^4 x$ subexpressions. They considered this part of equation as more representative from the viewpoint of structural syntactical similarity. However, the subexpressions $\cos 2x$ and $\cos 4x$ were recognized by the algorithm as subexpression based similar equations to the query since both contains the explicit multiplier before x. Similar to the case of structural similarity this issue could be addresses by the equation representation normalization (i.e. introducing an explicit multiplier equal to 1).

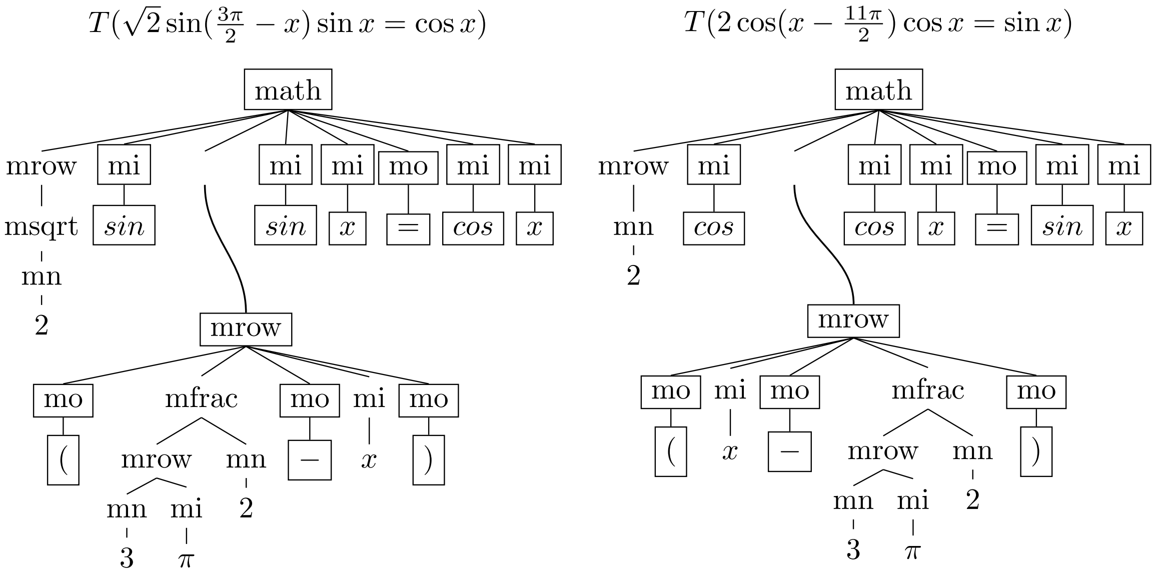


Figure 10: Structural similarity computation: example.

Table 8: Evaluating classification precision for a case of subexpression similarity.

No.	Expression	$\frac{TP}{TP+FP}$
1	$\cos(\frac{\pi}{2}+2x)=\sqrt{2}\sin x$	1/1
2	$\cos x+\cos(\frac{\pi}{2}+2x)=0$	1/1
3	$(2\cos x+1)(\sqrt{-\sin x}-1)=0$	1/1
4	$(2\sin x-1)(\sqrt{-\cos x}+1)=0$	1/1
5	$\sqrt{2}\sin(\frac{3\pi}{2}-x)\sin x=\cos x$	1/1
6	$\sin(\frac{7\pi}{2}-x)\sin x=\cos x$	1/1
7	$\frac{2\sin^2 x-\sin x}{2\cos x-\sqrt{3}}=0$	1/1
8	$\frac{2\sin^2 x-\sin x}{2\cos x+\sqrt{3}}=0$	1/1
9	$2\sin^4 x+3\cos 2x+1=0$	3/4
10	$4\cos^4 x-4\cos^2 x+1=0$	3/4
11	$\cos^2 x-\frac{1}{2}\sin 2x+\cos x=\sin x$	3/4
12	$\frac{2\sin^2 x-\sin x}{2\cos x-\sqrt{3}}=0$	4/4
13	$\frac{2\sin^2 x-\sin x}{2\cos x+\sqrt{3}}=0$	4/4

In sum, based on the results presented in Table 5, for the subexpression similarity sample test corpus the average precision $P=\frac{\frac{1}{1}+\frac{1}{1}+\cdots+\frac{3}{4}+\frac{4}{4}+\frac{4}{4}}{13}=\frac{12.25}{13}=0.94$.

However, such an accuracy achieved for a small test corpus defined in Table 5 may be considered as rather promising but very preliminary evaluation results. Further investigations with using more representative equation corpora are necessary.

5 Conclusion

In this study we adopted a tree overlapping algorithm (used originally in NLP) for mathematical equation syntactical similarity. We implemented the algorithm as a software prototype and arranged a set of experiments with sample test corpora. We discovered that the proposed modification fits well a selection of equations from college-level teaching practice both for the cases of structural and subexpression based syntactical similarity patterns. For the reason that the current implementation has some drawbacks which became evident after the arranged experiments, the further steps towards equation normalization are required in order to achieve better equation classification accuracy.

References

[1] R. Bod. Beyond grammar. *An Experienced-Based Theory of Language. CSLI Lecture Notes*, 88, 1998.

[2] E. D. Demaine, S. Mozes, B. Rossman, and O. Weimann. An optimal decomposition algorithm for tree edit distance. *ACM Transactions on Algorithms (TALG)*, 6(1):2, 2009.

[3] E. Denisova-Schmidt and E. Leontyeva. The unified state exam in russia: problems and perspectives. *International Higher Education*, (76):22–23, 2014.

[4] I. Hiroshi, H. Keita, H. Taiichi, and T. Takenobu. Efficient sentence retrieval based on syntactic structure. In *Proceedings of the COLING/ACL on Main conference poster sessions*, pages 399–406. Association for Computational Linguistics, 2006.

[5] S. Kamali and F. W. Tompa. Improving mathematics retrieval. *Towards a Digital Mathematics Library.*

Grand Bend, Ontario, Canada, July 8-9th, 2009, pages 37–48, 2009.

[6] M. Pawlik and N. Augsten. Rted: a robust algorithm for the tree edit distance. *Proceedings of the VLDB Endowment*, 5(4):334–345, 2011.

[7] M. Ponomarev and E. Pyshkin. Adopting tree overlapping algorithm for mathml equation structural similarity evaluation. In *Proceedings of the 2nd International Conference on Applications in Information Technology (ICAIT-2016)*, pages 17–20. The University of Aizu, The University of Aizu Press, Oct 2016.

[8] K. Sain, A. Dasgupta, and U. Garain. Emers: a tree matching–based performance evaluation of mathematical expression recognition systems. *International Journal on Document Analysis and Recognition (IJDAR)*, 14(1):75–85, 2011.

[9] K. Yokoi and A. Aizawa. An approach to similarity search for mathematical expressions using mathml. *Towards a Digital Mathematics Library. Grand Bend, Ontario, Canada, July 8-9th, 2009*, pages 27–35, 2009.

[10] K. Zhang and D. Shasha. Simple fast algorithms for the editing distance between trees and related problems. *SIAM journal on computing*, 18(6):1245–1262, 1989.

[11] Q. Zhang and A. Youssef. *An Approach to Math-Similarity Search*, pages 404–418. Springer International Publishing, Cham, 2014.