Answer Retrieval for Mathematics Questions

Sachin Sourav Munda

Answer Retrieval for Mathematics Questions

DISSERTATION SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

Master of Technology
in
Computer Science

by

Sachin Sourav Munda
[Roll No: CS-2025]

under the guidance of

Dr. Mandar Mitra.
Professor
Computer Vision and Pattern Recognition Unit

Indian Statistical Institute
Kolkata-700108, India

July 2022

To my family and my guide

CERTIFICATE

This is to certify that the dissertation entitled **"Answer Retrieval for Mathematics Questions"** submitted by **Sachin Sourav Munda** to Indian Statistical Institute, Kolkata, in partial fulfillment for the award of the degree of **Master of Technology in Computer Science** is a bonafide record of work carried out by him under my supervision and guidance.

Dr. Mandar Mitra
Professor,
Computer Vision and Pattern Recognition Unit,
Indian Statistical Institute,
Kolkata-700108, INDIA.

Acknowledgments

My deepest thanks to all the teachers of Indian Statistical Institute, for their valuable suggestions and discussions which added an important dimension to my research work.

Finally, I am very much thankful to my parents and family for their everlasting supports.

Last but not the least, I would like to thank all of my friends for their help and support. I thank all those, whom I have missed out from the above list.

Sachin Sourav Munda
Indian Statistical Institute
Kolkata - 700108 , India.

Chapter 1

Introduction

1.1 Mathematical Information Retrieval (MathIR)

Information Retrieval (IR) deals with retrieving relevant documents from a collection, given some input queries. MathIR, or Mathematical Information Retrieval, is one of the applications of IR that aims to develop "math-aware" search engines that are capable of searching for mathematical formulae or expressions in scientific documents. Mathematical formulas have a crucial role in engineering and scientific documents to express ideas and concepts. Searching for any mathematical formula or expression using text-based search engines is typically ineffective because they are unable to identify and handle special characters and structures found only in math formulas. This led to the growth of the *MathIR* community, and the development of "math-aware" search engines that can handle mathematical formulas.

During 2013–2016, **NTCIR** (NII Testbeds and Community for Information access Research) organised the MathIR Task to aid the development of such math-aware search engines [4, 3, 19]. The NTCIR MathIR task* involves retrieving useful documents from a corpus of either Wikipedia or arXiv articles in response to queries consisting of one or more formulas with or without keywords. The MathIR task was held under the aegis of NTCIR three times. We use the dataset created for NTCIR-12†, the third and final iteration of the task, as a benchmark for evaluation.

The ARQMath‡ (Answer Retrieval for Question on Math) Lab series is a more recent initiative that also aims to provide an evaluation framework for math-aware search engines [18, 5]. ARQMath has been organised under the aegis of CLEF§ (Conference and Labs of the Evaluation Forum), and was held in 2020, 2021 and 2022.
ARQMath defines a MathCAQ (Mathematical Community Question Answering) task

*https://ntcir-math.nii.ac.jp/
†http://research.nii.ac.jp/ntcir/ntcir-12/index.html
‡https://www.cs.rit.edu/~dprl/ARQMath/
§https://clef2022.clef-initiative.eu/

that makes use of data from the Math StackExchange[¶] (MSE) site.

1.2 Formula Representations

These days, there are many documents where formulas are written. These formulas are written in various formats, e.g., LaTeX and **MathML**. LaTeX is a widely used linearized encoding for representing the syntax of formulas in scientific and technical documents. On the other hand, **MathML** is predominantly used for mathematical content in websites. It is also used in computer algebra systems and print typesetting. It is a markup language (ML) based on XML. It encodes formulas using two tree-like representations.

Presentation MathML

```
<mrow>
  <mrow>
    <msup>
      <mi>x</mi>
        <mn>2</mn>
      </msup>
      <mo>+</mo>
    <mrow>
      <mn>4</mn>
      <mo>&InvisibleTimes;</mo>
      <mi>x</mi>
    </mrow>
    <mo>+</mo>
    <mn>4</mn>
  </mrow>
    <mo>=</mo>
    <mn>0</mn>
</mrow>
```

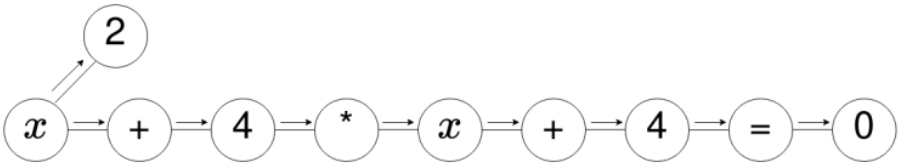
Content MathML

```
<apply>
  <eq/>
    <apply>
      <plus/>
        <apply>
          <power/>
          <ci>x</ci>
          <cn>2</cn>
        </apply>
        <apply>
          <times/>
          <cn>4</cn>
          <ci>x</ci>
        </apply>
        <cn>4</cn>
     </apply>
     <cn>0</cn>
</apply>
```

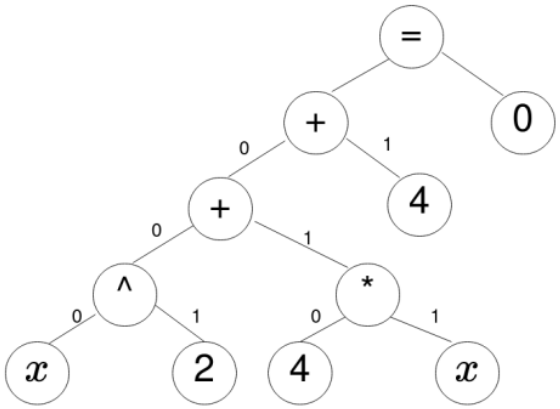
Table 1: Representations for the formula $x^2 + 4x + 4 = 0$. Presentation MathML uses presentational tags that "... generally start with "m" and then use "o" for operator "i" for identifier "n" for number, and so on. The "mrow" tags indicate organization into horizontal groups." Content MathML uses semantic tags that "... take into account such concepts as "times", "power of" and so on".

[¶] https://math.stackexchange.com/

(a) $x^2 + 4x + 4 = 0$



(b)Symbol Layout Tree (SLT)



(c)Operator Tree (OPT)

Figure 1.1: SLTs are used to represent structures, and show positions of variables and operators. OPTs are used to represent the mathematical meaning of a formula.

- **Presentation MathML.** This is used to capture the Symbol Layout Trees (SLT), which is a tree-like structure used to represent the visual structure of mathematical expressions. The main features and drawbacks of Presentation MathML are listed below.
  - It encodes visual information.
  - Compared to LaTeX, it provides rich visual information.
  - It does not encode semantic mathematical meaning.

- In a certain case, SLT representation is ambiguous. A node will have a variable with respect to one formula if its semantic is the same then it will act like an operator in other formula. eg.$x - y = 0$ and $x + (-y) = 0$ have the same meaning but the SLT representation will generate ambiguity. example given in the figure below.

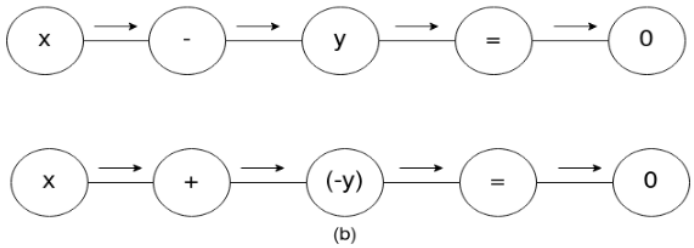


Figure 1.2: (a)$x - y = 0$ SLT (b)$x + (-y) = 0$ SLT

- **Content MathML** is used to capture the hierarchical structure of a mathematical formula, in a format that also represents the mathematical semantics of the formula. The main features and drawbacks of Content MathML are listed below.

  - It encodes semantics, i.e., the mathematical meanings of formulas.
  - Although there are common tools for converting LaTeX to MathML (like LaTeXML[‖]), translation errors can still happen, particularly for Content MathML, which needs the interpretation of semantics from visual data.
  - There is high degree of ambiguity during encoding if there is single or parallel markup. Content MathML if have parallel markup < *semantic* > tags it will create ambiguity. Example if our single query has multiple formula then the query Content MathML file will have two or more < *semantic* > tags which lead to ambiguity during encoding.
  - Operator Trees (OPTs) are mathematically unambiguous. Since OPTs have variables at leaf nodes and operators at internal nodes, the parsing will maintain the operation order.

Both forms of MathML are tree-based encodings. Both Presentation MathML and Content MathML bring advantages and disadvantages to math-aware search engines. LaTeX encoding is also linearized, and can be used for in existing text-based search systems, but LaTeX can only represent SLTs or structural information.

[‖] https://dlmf.nist.gov/LaTeXML/

1.3 Vector Representation for formulas

We embed formulas into vector representations. Formulas represented in a vector are used to calculate similarity between the formulas that is further used for retrieval. Mathematical notations give a greater advantage since we are using operator trees to detect semantic relationships which words or text cannot provide. These semantic relationships are beneficial for the construction of embedding models.

Our primary goal is to check different kinds of embedding models that are suitable for formula retrieval. We ignore surrounding text when constructing formula embeddings. DeepWalk is an embedding model which uses different ways of traversing a graph with a focus on trees. To linearize the formula embedded in the tree, the tree is traversed either breadth-first, depth-first, or via a random walk. After traversing the graph, we generate tuples and their tokens, which are further used in embedding models. Normally, graph embedding models take each node individually, but when constructing embedding models for mathematical formulae, we further categorise nodes, e.g., some are variables and some are constants.

1.4 Problem Statement

The problem is to build better embedding models for formula retrieval. We are using the NTCIR12 MathIR WikiCorpus $v2.1.0$** data set provided by NTCIR.

There are reasonably effective search engines for text, but we cannot directly apply these search engines to formulas, since formulas are more dependent on their structure and layout. There have not been many attempts to embed formulas, in contrast to the proliferation of word embeddings. The distributed bag of words (PV-DBOW) [13], a variation of the doc2vec approach, was first introduced in early research on formula embedding by Thanda et al. [2]. Utilizing binary expression trees, they gave each formula a real-valued vector so that formulae with comparable structures would be close to one another in vector space.

As far as we are aware, previous research has not concentrated on embedding isolated formulae, which simply takes formula structure and similarity between symbols and operations into account. A formula is mapped to a tree-like hierarchical structure, that is then linearised, and used to generate formula embeddings. There are further used to compute similarities between formulas.

**https://www.nii.ac.jp/dsc/idr/en/ntcir/ntcir.html

Chapter 2

Background

Previous approaches to formula retrieval are based on three types of indexing: text-based, tree-based and using SLTs and OPTs.

- **Text-Based Indexing**: Math formulas are converted to a sequence of tokens (like text). For retrieval, traditional TF-IDF models [14] or word embedding based techniques [1] may be used.
- **Tree-Based Indexing:** Formulas are represented using tree-like representations (either SLTs or OPTs), which are then linearized. Tuples extracted from the linear representation are used as indexing units [16, 10, 22].
- **Spectral Approaches:** As retrieval primitives, we use paths in OPTs/SLTs, i.e., features are extracted from tree while in the previous case, the tree is linearized. While traversing the tree, paths from the root to internal nodes and to leaves are stored. While constructing these paths, it is important to detect operator commutativity.

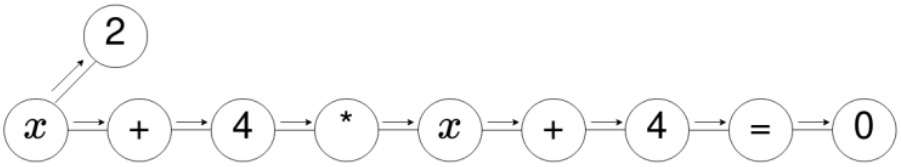
2.1 Representation of Math Formula

In order to encode mathematical formulae, they are represented as SLTs and OPTs. SLTs are constructed from the LaTeX representation of formulae. There are certain cases, e.g., $x - y = 0$ and $x + (-y) = 0$, where a symbol can be variable in one particular node, while the position of the same node can be a operator with same structure. This shows that SLTs can be ambiguous.

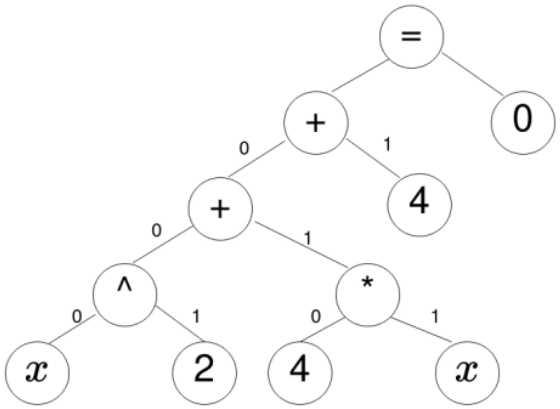
For Content MathML we use OPTs (Operator Tree). Since the operators are internal nodes, and variables are leaves, these OPTs are unambiguous. Since OPTs are unambiguous, we can construct a way to convert SLTs to valid OPTs, but we have to take care of syntax of the operator, its precedence and their respective associativity

as shown in figure below.

(a) $x^2 + 4x + 4 = 0$



(b)Symbol Layout Tree (SLT)



(c)Operator Tree (OPT)

Figure 2.1: SLTs are used to represent structures, and show positions of variables and operators. OPTs are used to represent the mathematical meaning of a formula.

For our work, we use the Tangent-s [8] formula search engine, which constructs SLTs and OPTs from formula representations. Nodes in the SLTs and OPTs here reflect distinct symbols and explicit aggregates like function parameters in the manner given below.

2.1.1 Nodes

Nodes in SLTs and OPTs are assigned individual symbols, with optional additional arguments, and are of the form: (**Type!**value). The list of different node types is given below.

- (**N!**n): nodes with numbers.
- (**V!**v): identifiers / variables.
- (**T!**t): text fragments, e.g., **'lim'** and **'such that'**.
- (**F!**) : fractions.
- (**R!**) : radicals.
- (**W!**) : explicitly specified white space.
- (**M!**f rxc): tabular expression or matrices, possibly surrounded by parentheses or similar delimiters. Here, r and c represent, respectively, the number of rows and columns in the structure, and f represents the fence characters.
  - Matrix (M!M)
  - Set (M!S)
  - List (M!L-)
  - Delimited (M!D-)
  - Matrix Row (M!R!)
  - Case (M!C!)
- (**C!**) : constants.
- (**T!**) : Text.
- (**t!**) : use a subtree to define operation.
- (**E!**) : error.
- (**-!**) : unknown type.
- (**$!**) : temporary nodes.
- (**U!**) and (**O!**) : **U!**=commutative **O!**=non-commutative operator in OPTs. (Since SLTs are ambiguous, they don't have these operators.)

2.1.2 SLT Edges

We define edge labels based on the spatial relationship between the nodes (symbols) that the edge connects. Consider an object (node or symbol) O. We define seven types of edge labels to define these relationships.

- **element**('e') : refer to the next element appearing after O in row-major order inside a structure represented by **M!**.
- **next**('n') : to reference that adjacent object appear right of O and on the same line.
- **pre-above**('SUP') : refer leftmost object of superscript of O.
- **below**('b') : refer leftmost object on a lower line starting at position below O.
- **pre-below**('SUB') : refer leftmost object of subscript of O.
- **within**('w') : refer to radicand if O is root, or first element appearing in row-major order in O if it is a structure represented by **M!**.
- **above**('a') : refer leftmost object on a higher line starting at position above O.

Below figure show the SLT Edges examples.

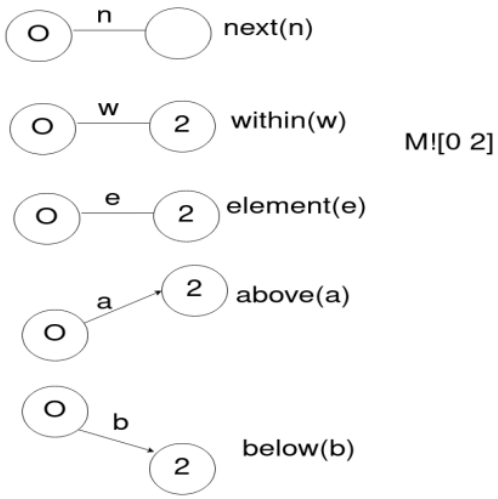


Figure 2.2: Examples of SLT Edges

2.1.3 OPT Edges

In OPTs, edge labels indicate argument position. For commutative operators (e.g., '+'), edges are unlabelled or labelled 0. For non-commutative operators, arguments are indexed from 0, as shown in the figure below.

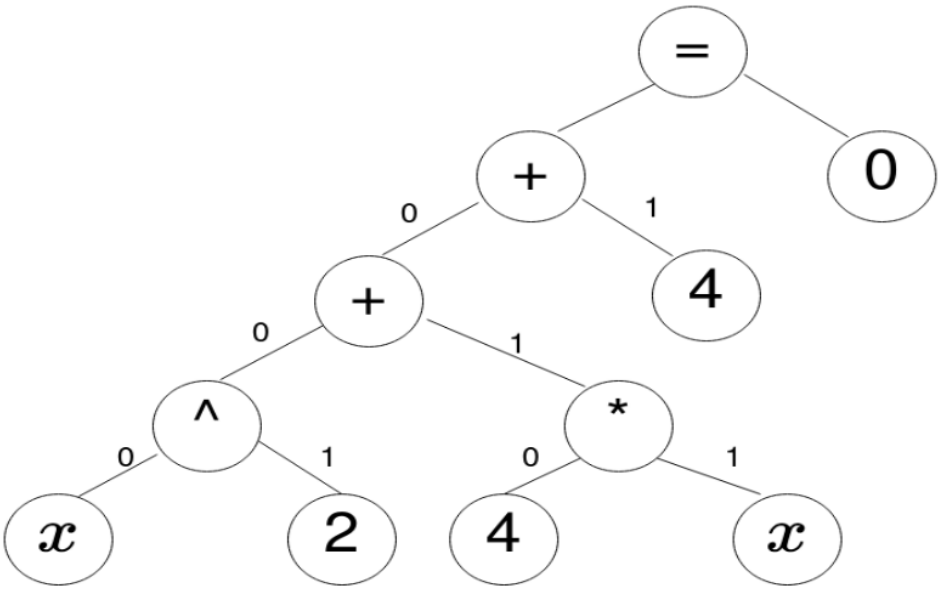


Figure 2.3: Operator Tree (OPT) of $x^2 + 4x + 4 = 0$

2.2 Word2vec

In word2vec [21], semantically similar words get similar vector values. Two architecture are commonly used in word2vec: CBOW (Continuous Bag of Words) and Skip-Gram model. The main disadvantage of word2vec is that it cannot handle out of vocabulary words: if the query contains an invalid fragment / token, or if a token in the user query is unknown to the model (because it is not available in the training set), it simply cannot be represented. Thus, the system will not give accurate results. Therefore we go for FastText[15].

2.3 FastText

FastText is a n-gram embedding model derive from word2vec. The disadvantage of word2vec is that if there is a paragraph in which each word are the smallest unit vector then the internal word structure is ignored.
Advantage of fastText is that it is written in c++, support multiprocessing of vectors at the time of training, it is a shallow neural network, it uses hierarchical softmax which reduces computational cost.

Let a word w has a set of n-grams ζ_w, Let a dictionary of n-grams with size G, To calculate the vector representation of word w it will be sum of n-gram vector representation. To calculate score of a pair of context word.

$$s(w, c) = \sum_{g \in \zeta_w} z_g^T v_c \tag{2.1}$$

z_g= n-gram g vector representation. v_c context window size represented by w vector representation.

FastText uses words as a bags of n-grams, value of n from 1 to length of word. Example n-gram, n=[2,3], lets take a word "fast" it is represented as fa,as,st,fas,ast and fast in this n-gram will have minimum and maximum number we use different number during training so it's a hyper-parameter. For embedding layer we take the target word and it's respective n-gram vector to calculate mean this will be done at each step during the training.
Based on the calculated error target vector are updated uniformly. After n-gram vectors are generated sum of word vectors are embedded in n-gram vectors.

Chapter 3

Baselines

3.1 Approach0

Approach0 [23] it uses OPT (Operator Tree) for structure of formula using LaTeX formula, which generally have small expression to retrieves formulas. OPT represent semantic similarity of math formulas. Example (x^{-1} and $1/x$) below figures shows that these two formulas will have same OPT.
OPT represent use of operators to operands in an expression. For indexing sub-trees

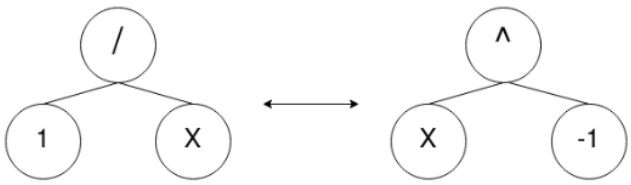


Figure 3.1: $1/x$ OPT and x^{-1} OPT

it uses leaf root path indexing as used in MCAT [12] in which leaf-root path act as retrieval units or "keywords" of formulas. SLT captures spatial relation to represent structure but they cannot capture semantic equivalence like operator associativity(to determine $1+x^2$ and x^2+1 are equivalent) and commutativity. It will find sub-tress that is identical in structure and those nodes that are similar have identical tokens. Better results are achieved by assigning higher weights to operands than operators since sometimes same operands with different operator means the same. It does not improve recall in query expansion of math having similar formulas(e.g. expand $1/z$ for z^{-1}).

3.2 Tangent-s

Tangent-s [9] uses both SLTs and OPTs for formula representation using three-layer model. Earlier Tangent-3 [11][17] use two stage SLT model to retrieve formula. Better formula results are obtained using the combination of SLTs and OPTs and then linearly combining their respective similarity scores.
For formula retrieval Three-layer retrieval is used.

- **Candidate Selection**: Formula Query constructed with SLTs/OPTs having symbols tuples, having matching query.
- **Matching of Structure**: Greedy algorithm is used to get maximum match between query and candidates. OPTs ignore the commutative operators matches example $x + y = 0$ and $0 = y + x$ therefore we use pair-wise matching which is computed in polynomial time it allow the perfect match of $x + y = 0$ and $0 = y + x$.
- **Linear Regression**: NTCIR-12 contain relevance judge data using that we train least squares linear regressor to combine previous two scores to produce a final rank.
- **SLT and OPT are combined** : For each representation symbol pair are retrieved having index the top-k candidate obtained from the index are combined to get a single list.

Chapter 4

Experimental Setup

4.1 Dataset

NTCIR-12 MathIR Wikipedia Formula Browsing Task [20]. It contains 5,90,000 mathematical formulas contain in English Wikipedia. The collection contain 'math' articles contained in $< math >$ tags, while 'text' are not.
Corpus contains:-

- *MathTagArticles*: 10% collection contains $< math >$ tag.
  - 31,742 article
  - 551,675 formulae
- *TextArticles*: 90% contains without $< math >$ tag.
  - 285,925 articles
  - 12,271 formulae

We consider isolated formulas as queries for the Wikipedia data.

4.2 Test Queries:

Relevant formulas are those that are thought to be comparable to the query formula in appearance and/or mathematical content. The Wiki-formula task has no keywords.

Concrete queries in which we use formulas used in Wikipedia articles. Wildcard queries it is a placeholder (eg. $*1 * x = 0$ we can place any value in position of 1) for subexpressions (eg. when we forget to put a particular symbol in a formula).

Table 4.1

Data-set	All Topics	Concrete Only	Wildcard Only
NTCIR12-MathIR-WikiCorpus-v2.1.0	40	20	20

4.3 MathML Formula Representation

Formulae translated into MathML i.e Math XML* encoding, LaTeXML† convert each formula from LaTeX to MathML, it produce three representations for each formula:

1. Presentation MathML (SLT) it is shown with $< semantic >$ tag.
2. Content MathML (OPT) is is shown with $< annotation - xmlencoding = "MathML - Content" >$ tag.
3. LaTeX string (symbol layout for appearance), demarcated by: $< annotation - xmlencoding = "application/x - tex" >$.

4.4 Document Processing

4.4.1 Extraction of MathML form the HTML file

For the query and document containing only formula we will first extract the MathML formula from their respective HTML file which contain text as well as formula.

Below is the HTML file for the formula **Monic polynomial:** $ax^2 + bx + c = 0$

```
<html lang="en">
<head>
<meta charset="utf-8"/>
<title charset="1985">Monic polynomial</title>
<script src="https://cdn.mathjax.org/mathjax/latest/MathJax.js?
    config=TeX-AMS-MML_SVG.js" type="text/javascript">
</script>
</head>
<body>
<h1>Monic polynomial</h1>
<hr/>
<p>In <a class="uri" href="algebra" title="wikilink">algebra</a>
    </p>
<p>
```

*http://www.w3.org/Math/
†http://dlmf.nist.gov/LaTeXML/

```
<h4 id="polynomial-equation-solutions">Polynomial equation
    solutions</h4>
<p>In other respects, the properties of monic polynomials and of
    their corresponding monic <a href="polynomial_equation" title=
    "wikilink">polynomial equations</a></p>
<p>
<math display="block" id="Monic_polynomial:1">
 <semantics>
  <mrow>
   <mrow>
    <mrow>
     <mpadded lspace="5pt" width="+5pt">
      <mi>a</mi>
     </mpadded>
     <msup>
      <mi>x</mi>
      <mn>2</mn>
     </msup>
    </mrow>
    <mo>+</mo>
    <mrow>
     <mi>b</mi>
     <mi>x</mi>
    </mrow>
    <mo>+</mo>
    <mi>c</mi>
   </mrow>
   <mo>=</mo>
   <mn>0</mn>
  </mrow>
.
.
.
.
 </semantics>
</math>
</body>
</html>
```

Let query be **Monic polynomial:** $ax^2 + bx + c = 0$
Below is the example of a query MathML file after processed. Mainly tags like $< semantics >$ is a container element associating mathematical annotations. $< mi > a < /mi >$ containing variable name, $< mn > 2 < /mn >$ containing numbers, $< mo > + < /mo >$ containing operators.

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="application/xhtml+xml;
    charset=UTF-8" />
</head> <body>
```

```
<div>
<math id="doc-REPLACE_ID_NUMBER:0" xmlns="http://www.w3.org/1998/Math/MathML" alttext="unknown" display="block">
<semantics>
      <mrow>
            <mrow>
              <mrow>
                <mrow>
                  <mi>a</mi>
                  <msup>
                    <mi>x</mi>
                    <mn>2</mn>
                  </msup>
                </mrow>
                <mo>+</mo>
                <mrow>
                  <mi>b</mi>
                  <mi>x</mi>
                </mrow>
                <mo>+</mo>
                <mi>c</mi>
              </mrow>
              <mo>=</mo>
              <mn>0</mn>
            </mrow>
            <annotation-xml encoding="MathML-Content">
              <apply>
                <eq/>
                <apply>
                  <plus/>
                  <apply>
                    <times/>
                    <ci>a</ci>
                    <apply>
                      <csymbol cd="ambiguous">superscript</csymbol>
                      <ci>x</ci>
                      <cn type="integer">2</cn>
                    </apply>
                  </apply>
                  <apply>
                    <times/>
                    <ci>b</ci>
                    <ci>x</ci>
                  </apply>
                  <ci>c</ci>
                </apply>
                <cn type="integer">0</cn>
              </apply>
            </annotation-xml>
            <annotation encoding="application/x-tex">\ ax^{2}+bx+c=0</annotation>
```

```
53 </semantics>
54 </math>
55 </div>
56 </body> </html>
```

Chapter 5

Proposed Work

5.1 Embedding using FastText

Tangent CFT (Tangent-s Combined with fastText). In order to embed mathematical formulas as we analysed from above the fastText embedding model is good to embed.

1. Mathematical formulas are linearize so that we can apply text based embedding model.
2. The linearized formulas are then applied with n-gram embedding model.

5.1.1 Generation of Tuple Sequence

1. Using Tangent-s the LaTeX or MathML format formulas are converted into SLT and OPT encodings.
2. To generate tuple sequence depth-first traversal is used.
3. A pairs of symbols tuples are generated using Tangent-s with their relative position.
4. A tuples $((s_1, s_2, R, FRP)$ for traversing tree in depth first we will store during the traversal of the node with the following details which is used to build tuples. In SLT queries are parsed from the root for tuples generation.
   - s_1 To determine the ancestor symbol.
   - s_2 To determine the descendant symbol.
   - R in order to get the relation between the symbols we use edge label sequence from s_1 to s_2.

- FRP (Full relative path), the whole relative path indicating s1's location along its route from the tree's root. eg. In the figure given below FRP of variable x to 0 is nnnn.
- w (window size) to control maximum path between path length of the symbols.
- EOB(End-of-Base) flag to match small expression system will create dummy pairs at the last symbol on each baseline and null.

In OPTs EOB tuple actually represents end of root-leaf path thus EOB flag is generated for all arguments in leaves of OPT.

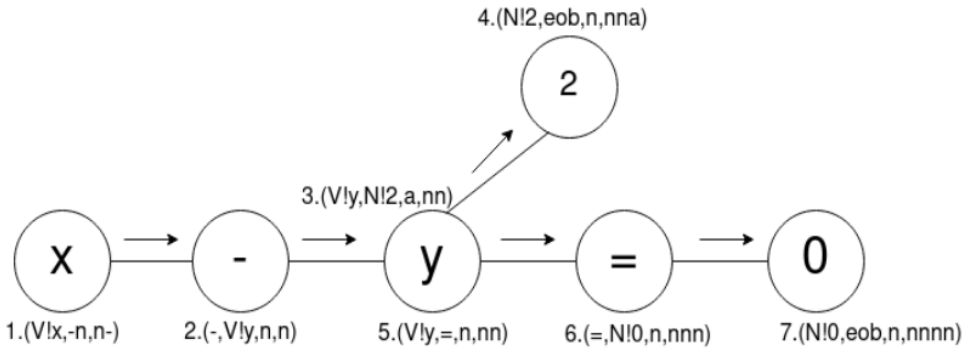


Figure 5.1: SLT representation with tuples of $x - y^2 = 0$

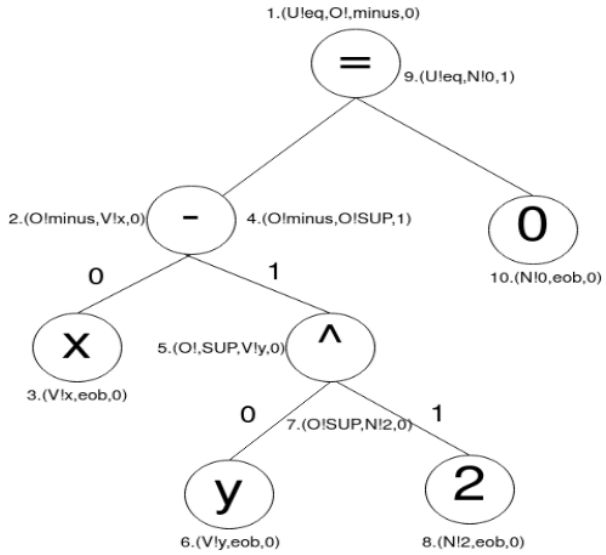


Figure 5.2: OPT representation with tuples of $x - y^2 = 0$

$$x - y^2 = 0$$

SLT tuples	OPT tuples (FRP omitted)
(**V!** x, -, n, -)	(**U!** eq,**O!** minus,0)
(-, **V!** y, n, n)	(**O!** minus, **V!** x, 0)
(**V!** y, **N!** 2, a, nn)	(**V!** x, eob, 0)
(**N!** 2, eob, n, nna)	(**O!** minus, **O!** SUP, 1)
(**V!** y, =, n, nn)	(**O!** SUP, **V!** y, 0)
(=, **N!** 0, n, nnn)	(**V!** y, eob, 0)
(**N!** 0, eob, n, nnnn)	(**O!** SUP, **N!** 2, 0)
	(**N!** 2, eob, 0)
	(**U!** eq, **N!** 0, 1)
	(**N!** 0, eob, 0)

Figure 5.3: SLT and OPT Tuples generated after traversing their respective tree for the formula $x - y^2 = 0$.[13]

5.1.2 Tuples converted to Tokens:

Now we have tuple sequence after converting the formula representation using trees. To use n-gram embedding we need 'words' and 'character' in our representation. We consider tuple as a word. Each token(character) encoded with unique identifier. In order to differentiate between nodes and edges different token identifier are used for nodes and edges.

To get to know the formula structure tokenizing is essential, separating node from value provide detail structure of the formula.

- Example: Two formula sharing same structure but have different variable or constants
  $x_2 - y_2 = 0$ $a_2 - b_2 = 0$
  These two must have higher similarity score. If tokens are treated as characters then we can have higher similarity score.

FRP does not provide essential information its just give argument position of each operation therefore we use tokenized formula without FRP as a input of embedding model.

Example of tokenized tuple.
From Table-1:
(V!y, eob, 0) tokenized to 'V!', 'y', 'eob' and '0'
(O! SUP, V! y, 0) tokenized to 'O!', 'SUP', 'V!', 'y' and '0'

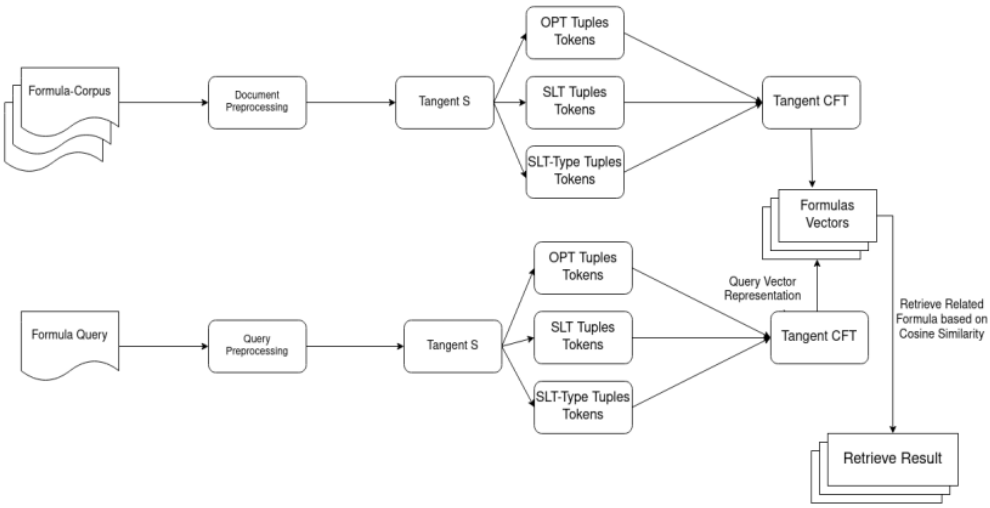


Figure 5.4: Overview of Formula Embedding to Retrieve Similar Formulas.

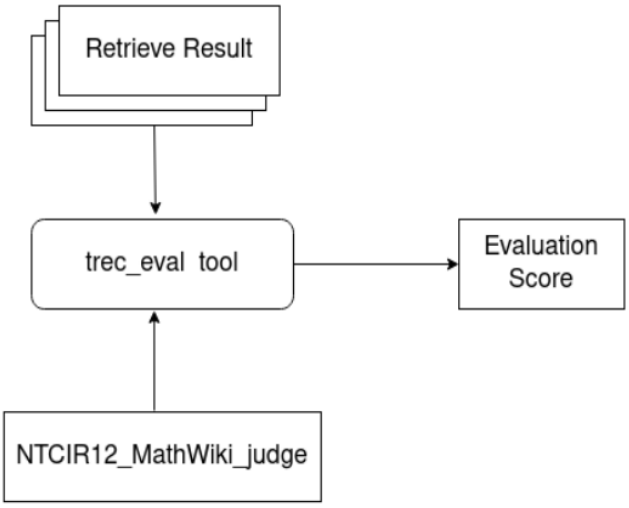


Figure 5.5: After retrieved result *trec_eval* tool is used where it's take two file judge file and the result file.

Given a results file and a set of judged results, trec_eval * is the standard tool employed by the TREC† (Text Retrieval Community) community to assess an ad hoc retrieval run.

5.1.3 Embedding fastText to Formulas

As described fastText with n-gram embedding is applied to embed mathematical formula. Context window size (hyper-parameter) 'T' are the nearby tuples in the linearized sequences.
T=5 (default value) and d = dimensional vector (defined before training). Formula(F) respective vector representation with set of n tuples $'T'_F$ given by:-

$$formulaVec(F) = \frac{1}{n} \sum_{t \in T_F} tupleVec(t) \tag{5.1}$$

Now we have a vector representations of each formula in the collection, in order to get vector representation of a given query we train our model. Now since we have vector representation of query we calculate cosine similarity of query with vector representation of formula on the corpus.
Similarity of two formulas use cosine similarity on their vector representation.
similarity between the formula query 'q' denoted by vector $'V'_q$ and a formula 'f' in the collection with vector denoted by $'V'_f$ is measured

$$sim(q, f) = \frac{V_f . V_q}{|V_f| . |V_q|} \tag{5.2}$$

Higher cosine similarity result in good rank.

5.1.4 Combining SLT and OPT

SLT captures visual representation while OPT captures formula semantics we combine these two to capture both OPT and SLT to have semantics as well as appearance of formula in a single vector. SLT type - it will focus on node type and ignore it's nodes value from SLT.
example ((V!y,N!3,a) will be considered as (V,N,a)). it is helpful in case name of the two variable in two formulas are same but they still have the same structure.
At last sum up the three vector representations: SLT, SLT-Type and OPT.

*https://github.com/usnistgov/trec_eval
†https://trec.nist.gov/

5.1.5 Training

There are total 4 steps in training:

1: **Embedding of SLT**

- We will train the FastText model on SLT tuples tokens which takes the train data with vector size=300, window=20, skip gram=1, Hierarchical Softmax, negative samples=30, epochs=40, min and max word =1, n-gram=1.

- After that we will save the fastText model of SLT.

2: **Embedding of OPT**

- We will train the FastText model on OPT tuples tokens which takes the train data with vector size=300, window=5, skip gram=1, Hierarchical Softmax, negative samples=20, epochs=20, min=3 and max=6 words, n-gram=1.

- After that we will save the fastText model of OPT.

3: **Embedding of SLT Type**

- We will train the FastText model on SLT Type tuples tokens which takes the train data with vector size=300, window=5, skip gram=1, Hierarchical Softmax, negative samples=20, epochs=30, min=3 and max=6 words, n-gram=1.

- After that we will save the fastText model of SLT Type.

4: **Combining SLT,OPT and SLT Type model to get Final Result**
Our embedding are built on their OPT and SLT representations, with the OPT capturing the semantics of the formula and the SLT concentrating on its visual structure. As a result, by merging the OPT and SLT vector representations, we might be able to encapsulate the semantics and appearance of a formula in a single vector. From the SLT representation, we trained a different model called SLT-Type that only takes into account the node types and ignores its values. The tuple $(V!x, N!2, a)$, for example, will be regarded as (V,N,a). When variables in two formulas have different names but have the same basic structure, this may be helpful.

We add up the three vector representations of a formula SLT, SLT-Type, and OPT to get its final vector form.

Chapter 6

Results

For comparison we use bpref (binary preference-based measure)[7] it uses information from judged documents. It process how frequently relevant document retrieved before non-relevant documents.
For overall score of relevant formula we use Harmonic mean. Harmonic mean is the combination of full and partial bpref so that the retrieval system can retrieve relevant formula.

- f : query with relevant formulas.
- d : bpref detects the first d non-relevant formulas

d judged non-relevant formulas are compared with f judged relevant formulas.

$$\text{Harmonic mean bpref} = \frac{2PF}{(P+F)} \tag{6.1}$$

F=Fully relevant formulas P=Partially Relevant formulas.

6.1 Evaluation Measures:

Over 5,90,000 mathematical formulas are used for the NTCIR12-MathIR-WikiCorpus-v2.1.0 which contains Wikipedia pages containing mathematical formulas.40 queries have been found. The NTCIR-12 Wiki corpus is used to sample 20 equations, and then each formula has one or more sub-expressions eliminated or substituted with wildcard symbols.
Depending on relevance, each judge assigns each hit a score of 0, 1, or 2. (with 2 indicating highly relevant). The two judges' combined scores determine the final relevance rating, which ranges from 0 to 4. Scores of three or four are deemed entirely relevant, one or two are deemed somewhat relevant, and zero is deemed unimportant.

bpref:(Binary Preference-based Measure:
It uses the details from the documents to be judged. An organiser provide the details of the judged documents. It is a function of how much a relevant documents are retrieved before the non-relevant documents.

$$bpref = \frac{1}{R}\sum_{r} 1 - \frac{|\text{n ranked ranked higher than r}|}{R} \tag{6.2}$$

P@K:(Precision @ K)
Precision after k documents have been retrieved. Set a threshold k so that documents ranked below k are ignored now compute relevance percentage on top k documents.

ndcg (Normalized Discounted Cumulative Gain)
Cumulative Gain is calculated in which we take the sum of all the gains till k documents.

$$CG(k) = \sum_{i=1}^{k} G_i \tag{6.3}$$

In order to handle the ordering of the ranked documents and to penalize higher ranked documents positioned at the last we use Discounted Cumulative Gain.

$$DCG(k) = \sum_{i=1}^{k} \frac{G_i}{\log_2(i+1)} \tag{6.4}$$

But DCG add up which increase the length of recommended documents which will give preference to only top documents but it will not deal with length of the documents.
To handle above problem we use ideal Discount Cumulative Gain (IDCG) which ranks documents relevance up to k position.

$$IDCG(k) = \sum_{i=1}^{|I(k)|} \frac{G_i}{\log_2(i+1)} \tag{6.5}$$

we now normalize the value from range of 0 to 1. This will termed as NDCG

$$NDCG(k) = \frac{DCG(k)}{IDCG(k)} \tag{6.6}$$

6.2 Result:

Score based on BPREF:
Tangent-CFT score better in partial relevant BPREF than Tangent-s cause as it

Retrieved Results	Partial BPREF	Full BPREF	Harm. Mean BPREF
Approach0	0.59	**0.67**	0.63
Tangent-s	0.59	0.64	0.61
Tangent-CFT	**0.71**	0.59	**0.65**

Table 6.1: Results on NTCIR12-MathIR-WikiCorpus-v2.1.0 data set(**Avg. bpref@1000**)

uses fastText for embedding the SLT,OPT and SLT Type tuples tokens where the embedded vectors still preserve the structural representation of the formula but not all the relevant documents are structures are exactly preserved but partially some of the documents relevance document structure are preserved which gives 12-14% greater score compared to Tangent-s where we uses matching and scoring for SLT and OPT and concatenated score is generated result in single vector then we apply linear regressor to calculate relevance score.
There is a 4%-7% difference between Tangent-CFT and Tangent-s, Approach0 in Full BPREF score the main reason behind this is that Approach0 which uses matching sub-tree of OPT from root-leaf path therefore since it uses only tree it has to go only one path therefore have greater BPREF score on Full document while Tangent-s match using SLT and OPT for score. Since Approach0 and Tangent-s uses only tree for the similarity it is more accurate compare to Tangent-CFT which uses fastText and after that cosine similarity which is a abstract vector representation.

Score based on P@K:

Retrieved Results	Relevant				Partially Relevant			
	P@5	P@10	P@15	P@20	P@5	P@10	P@15	P@20
Tangent-s	0.4400	0.3150	0.2583	0.2162	0.7000	0.6075	0.5550	0.511
Tangent-CFT	0.4600	0.3200	0.2667	**0.2400**	0.9100	0.8500	0.8267	**0.802**

Table 6.2: Results on NTCIR12-MathIR-WikiCorpus-v2.1.0 data set

Tangent-CFT score higher than Tangent-s in all precision score showing Tangent-CFT is faster than Tangent-s.

Score based on Average nDCG@K

Retrieved Results	Concrete Only	
	nDCG@5	nDCG@20
Tangent-s	0.8136	0.7980
Tangent-CFT	**0.8509**	**0.8280**

Table 6.3: Results on NTCIR12-MathIR-WikiCorpus-v2.1.0 data set

Due to the small number of expressions that are to be judged per topic and the fact that our new conditions yield a large number of unrated outcomes in the top-20, we have adopted a different method to further examine the rankings generated. We used each stage of our retrieval model to re-rank all judged formulas for each topic, and we compared these ranks to the ideal rankings using nDCG@K (k=5,20).

Chapter 7

Conclusion and Future Work

7.1 Conclusion

We presented TangentCFT, used as a embedding model for mathematical formulas. FastText is used for embedding for both SLTs for formula structure, and OPTs for formula semantics. During the Embedding process the tree based formula representation are converted to tuples and then tuples are encoded using n-gram on a node and it's neighbour nodes. The we tokenize the tuples with their respective tree representation OPT tuples, SLT tuples, SLT-Type tuples. Each tokenized tuples are fed to fastText embedding known as TangentCFT that gives us query vector representation it will retrieved related formulas based on cosine similarity.

7.2 Future Work

More data based study with model not limited up-to the resource. We can have more test collection containing more complex query formulas that doesn't give exact match. We can include text along with formulas. We can have query like proof with several steps and match it's similarity with others.
We can have Question related to mathematical question containing formulas and the model can give solution the the related questions.
On the basics of this concept we can have other retrieval systems for chemical diagrams, tables, flowcharts, figures.

Bibliography

[1] Abhinav Thanda, Ankit Agarwal, K.S.A.P., Abhishek Gupta. 2016. A document retrieval system for math queries. (2016), ..:

[2] Abhinav Thanda, Ankit Agarwal, K.S.A.P., for Math Queries. In NTCIR., A.G..A.D.R.S.:

[3] Akiko Aizawa, Michael Kohlhase, I.O., Moritz Schubotz. NTCIR-11 Math-2 Task Overview. In NTCIR-11. National Institute of Informatics (NII), .:

[4] Akiko Aizawa, Michael Kohlhase, and Iadh Ounis. NTCIR-10 Math Pilot Task Overview. In NTCIR-10. National Institute of Informatics (NII), .: (2013)

[5] Behrooz Mansouri, Richard Zanibbi, D.W.O., Anurag Agarwal. Overview of ARQMath-2 (2021): Second CLEF Lab on Answer Retrieval for Questions on Math (Working Notes Version). In CLEF 2021, volume 2936 of CEUR Workshop Proceedings, p....:

[6] Bryan Perozzi, R.A.R., learning of social representations. In Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery, S.S..D.O., ACM., D.M.:

[7] Buckley, C., evaluation with incomplete information. In Proceedings of the 27th Annual International ACM SIGIR Conference on Research, E.M.V..R., in Information Retrieval., D.:

[8] Davila, K., Layout, R.Z.., semantics: Combining representations for mathematical formula search. In Proceedings of the 40th In-ternational ACM SIGIR Conference on Research, in Information Retrieval., D.:

[9] Davila, K., Layout, R.Z.., semantics: Combining representations for mathematical formula search. In Proceedings of the 40th In-ternational ACM SIGIR Conference on Research, in Information Retrieval., D.:

[10] Kamali, S., rank Wm Tompa. 2013. Structural similarity search for mathematics retrieval. In Intelligent Computer Mathematics, LNCS vol. 7961. Springer, ..:

[11] Kenny Davila, Richard Zanibbi, A.K., at the NTCIR-12 MathIR task. In Proc. NTCIR-12. 338-345., F.W.T..T..:

[12] Kristianto, G., T.G.A.A.M.m.r.s.f.N..M.t.: Kristianto, g., topic, g., aizawa, a.: Mcat math retrieval system for ntcir-12 mathir task (06 2016))

[13] Le, Q., representations of sentences, T.M..D., documents. In International Conference on Machine Learning.:

[14] Miller, B.R., aspects of the digital library of mathematical functions. Annals of Mathematics, A.Y..T., Artificial Intelligence vol. 38, 1-3 (2003), ..:

[15] Piotr Bojanowski, Edouard Grave, A.J., word vectors with subword information. Transactions of the Association for Computational Linguistics., T.M..E.:

[16] Radu Hambasan, M.K., at NTCIR-11. In Proc. NTCIR-11. 114-119, C.C.P..M.W.:

[17] Richard Zanibbi, Kenny Davila, A.K., Frank Tompa. 2016. Multi-Stage Math Formula Search: Using Appearance-Based Similarity Metrics at Scale. SIGIR (2016), ..:

[18] Richard Zanibbi, Douglas W. Oard, A.A., Behrooz Mansouri. Overview of ARQMath 2020 (Updated Working Notes Version): CLEF Lab on Answer Retrieval for Questions on Math. In CLEF 2020, volume 2696 of CEUR Workshop Proceedings, .:

[19] Richard Zanibbi, Akiko Aizawa, M.K.I.O.G.T., Kenny Davila. NTCIR-12 MathIR Task Overview. In NTCIR-12. National Institute of Informatics (NII), .:

[20] Richard Zanibbi, Akiko Aizawa, M.K.I.O.G.T., NTCIR., K.D..N..M.T.O.I.:

[21] Tomas Mikolov, Ilya Sutskever, K.C.G.S.C., representations of words, J.D..D., phrases, their compositionality. In Advances in Neural Information Processing Systems.:

[22] Zhong, W., Hui Fang. 2016. OPMES: A Similarity Search Engine for Mathematical Content. In ECIR. Springer, ..:

[23] Zhong, W., for For-mulas Using Leaf-Root Paths in Operator Subtrees. In European Conference on Information Retrieval., R.Z..S.S.S.:

output.pdf

ORIGINALITY REPORT

23%
SIMILARITY INDEX

PRIMARY SOURCES

1	Behrooz Mansouri, Shaurya Rohatgi, Douglas W. Oard, Jian Wu, C. Lee Giles, Richard Zanibbi. "Tangent-CFT", Proceedings of the 2019 ACM SIGIR International Conference on Theory of Information Retrieval, 2019 Crossref	442 words — 7%
2	uwspace.uwaterloo.ca Internet	173 words — 3%
3	Kenny Davila, Richard Zanibbi. "Layout and Semantics", Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2017 Crossref	106 words — 2%
4	scholarworks.rit.edu Internet	83 words — 1%
5	link.springer.com Internet	76 words — 1%
6	Behrooz Mansouri, Richard Zanibbi, Douglas W. Oard. "Learning to Rank for Mathematical Formula Retrieval", Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2021 Crossref	54 words — 1%

7	"Intelligent Computer Mathematics", Springer Science and Business Media LLC, 2021 Crossref	41 words — 1%
8	archive.cnx.org Internet	38 words — 1%
9	www.sciplore.org Internet	36 words — 1%
10	functions.wolfram.com Internet	35 words — 1%
11	mcs.uwsuper.edu Internet	34 words — 1%
12	people.cs.georgetown.edu Internet	33 words — 1%
13	Pankaj Dadure, Partha Pakray, Sivaji Bandyopadhyay. "MathUSE: Mathematical information retrieval system using universal sentence encoder model", Journal of Information Science, 2022 Crossref	22 words — < 1%
14	git.uibk.ac.at Internet	21 words — < 1%
15	ebin.pub Internet	16 words — < 1%
16	www.springerprofessional.de Internet	16 words — < 1%
17	Yu Cao, Shawn Steffey, Jianbiao He, Degui Xiao, Cui Tao, Ping Chen, Henning Müller. "Medical	15 words — < 1%

Image Retrieval: A Multimodal Approach", Cancer Informatics, 2015
Crossref

18	api.crossref.org Internet	15 words — < 1%
19	synapse.koreamed.org Internet	15 words — < 1%
20	"Advances in Information Retrieval", Springer Science and Business Media LLC, 2019 Crossref	14 words — < 1%
21	Sourish Dhar, Sudipta Roy, Arnab Paul. "Scientific document retrieval using structure encoded string with trie indexing", Information Services & Use, 2022 Crossref	12 words — < 1%
22	dl.acm.org Internet	12 words — < 1%
23	fkiquality.com Internet	12 words — < 1%
24	ip.ios.semcs.net Internet	12 words — < 1%
25	api.jquery123.com Internet	11 words — < 1%
26	arxiv.org Internet	11 words — < 1%
27	docs.mathjax.org Internet	10 words — < 1%

www.commsp.ee.ic.ac.uk

28	Internet	10 words — < 1%
29	Falk Scholer. "User performance versus precision measures for simple search tasks", Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval - SIGIR 06 SIGIR 06, 2006 Crossref	9 words — < 1%
30	cutler.github.io Internet	9 words — < 1%
31	inftyreader.org Internet	9 words — < 1%
32	www.w3.org Internet	9 words — < 1%
33	Fangjie Yu, Zhiyuan Zhuang, Jie Yang, Ge Chen. "A Glider Simulation Model Based on Optimized Support Vector Regression for Efficient Coordinated Observation", Frontiers in Marine Science, 2021 Crossref	8 words — < 1%
34	Schubotz, Moritz. "Augmenting Mathematical Formulae for More Effective Querying & Efficient Presentation.", Technische Universitaet Berlin (Germany), 2020 ProQuest	8 words — < 1%
35	tsukuba.repo.nii.ac.jp Internet	8 words — < 1%
36	Leslie F. Sikos. "Web Standards", Springer Science and Business Media LLC, 2011 Crossref	6 words — < 1%

EXCLUDE QUOTES	OFF	EXCLUDE SOURCES	OFF
EXCLUDE BIBLIOGRAPHY	OFF	EXCLUDE MATCHES	OFF