Retrieving Documents With Mathematical Content

Shahab Kamali
David R. Cheriton School of Computer Science
University of Waterloo, Waterloo, ON, Canada
skamali@cs.uwaterloo.ca

Frank Wm. Tompa
David R. Cheriton School of Computer Science
University of Waterloo, Waterloo, ON, Canada
fwtompa@uwaterloo.ca

ABSTRACT

Many documents with mathematical content are published on the Web, but conventional search engines that rely on keyword search only cannot fully exploit their mathematical information. In particular, keyword search is insufficient when expressions in a document are not annotated with natural keywords or the user cannot describe her query with keywords. Retrieving documents by querying their mathematical content directly is very appealing in various domains such as education, digital libraries, engineering, patent documents, medical sciences, etc. Capturing the relevance of mathematical expressions also greatly enhances document classification in such domains.

Unlike text retrieval, where keywords carry enough semantics to distinguish text documents and rank them, math symbols do not contain much semantic information on their own. In fact, mathematical expressions typically consist of few alphabetical symbols organized in rather complex structures. Hence, the structure of an expression, which describes the way such symbols are combined, should also be considered. Unfortunately, there is no standard testbed with which to evaluate the effectiveness of a mathematics retrieval algorithm.

In this paper we study the fundamental and challenging problems in mathematics retrieval, that is how to capture the relevance of mathematical expressions, how to query them, and how to evaluate the results. We describe various search paradigms and propose retrieval systems accordingly. We discuss the benefits and drawbacks of each approach, and further compare them through an extensive empirical study.

Categories and Subject Descriptors

H.3.3 [**Information Storage And Retrieval**]: Information Search and Retrieval– Retrieval models

Keywords

Mathematics retrieval, Search, Math queries, Documents with math content.

1. INTRODUCTION

Many collections of documents contain mathematical expressions. Examples include technical and educational web sites, digital libraries, and other document repositories such as patent collections. Currently, due to the lack of an effective mathematics retrieval system, such rich mathematical knowledge is not fully exploited when searching for such documents. Querying with mathematical expressions, and consequently retrieving relevant documents based on their mathematical content, is not straightforward:

- Mathematical expressions are objects with complex structures and rather few distinct symbols and terms. The symbols and terms alone are usually inadequate to distinguish among mathematical expressions. For example $\sum_{i=1}^{n} i$ and $\sum_{j=1}^{m} j$ are quite similar, but 2^n and n^2 are quite dissimilar even though they share the same symbols.

- Relevant mathematical expressions might include small variations in their structures or symbols. For example, $1 + \sum_{i=1}^{n} i^2$ and $\sum_{j=1}^{n} j^k$ might both be useful matches to a query.

- Each mathematical expression has two sides: *i*) its appearance (or *presentation*) and *ii*) its mathematical meaning (often termed its *content*). The majority of the published mathematical expressions are encoded with respect to their appearance, and most instances do not preserve much semantic information.

As is true for other retrieval systems, a mathematics search engine should be evaluated based on its usefulness, that is, how well it can satisfy users' needs. What makes mathematics retrieval distinct is the difficulty of judging which mathematical expressions are relevant and which are not. For example, a user who is interested in $\sin^2(x)$ might also be interested in $\sin^3(x)$ but not in $\cos^2(x)$. We know of no consensus for similarity of mathematical expressions in general. On the other hand, if we were to limit the search to exact matches only, many relevant expressions will be missed, and the user might need to issue too many queries to find a useful answer. For example if the user is looking for $\sum_{i=1}^{10} \frac{1}{(i+1)^2}$, then pages that contain $\sum_{j=1}^{10} \frac{1}{(j+1)^2}$, $\sum_{i=1}^{n} \frac{1}{(i+1)^2}$, and $\sum_{x=1}^{n} \frac{1}{x^2}$ probably also address her needs.

Mathematics retrieval is still at an early stage. Unfortunately, content-based mathematics retrieval systems [7, 15] are limited to resources that encode the semantics of mathematical expressions, and they do not perform well with presentation markup. The lack of content information within web pages forces a retrieval system to rely mostly on the presentation of expressions, and it is often hard to judge whether a similar-looking expression is relevant to a query.

Some systems rely on the presentation of mathematical expressions [3, 9, 20, 26, 27, 28], but they either find exact matches only or they use models that ignore the whole or parts of the structure and usually return many irrelevant results. They do not define how to measure the relevance of matched mathematical expressions, and there has not been much effort to evaluate such systems in terms of the usefulness of search results.

In this paper we focus on the problem of matching mathematical expressions, and hence we assume that a query consists of a single expression. Systematically addressing this problem is a prerequisite for developing systems that handle more complex queries, such as ones that consist of multiple expressions or a combination of expressions and keywords.

Because mathematical expressions are often distinguished by their structure rather than relying merely on the symbols they include, we describe two search paradigms that incorporate structure:

1. Structural similarity: The similarity of two expressions is defined as a function of their structures and the symbols they share. The similarity is used as an indication of how relevant a document containing an expression is when given another expression as a query. We propose an algorithm based on tree edit distance to calculate the similarity of two expressions. Documents are ranked with respect to the similarity of their contained mathematical expressions to the query.

2. Pattern match: As an alternative approach, a query can be represented as a pattern or template. The added expressivity of such a query language provides the user with tools to specify more details, which allows more accurate and complete results in return. On the other hand, the richness and variety of mathematical concepts implies that the query language is potentially difficult to learn and use.

If the two mentioned approaches perform equally well, the simpler query language is probably preferred; the extra cost of forming a query with the expressive query language is justified only when this expressive power results in higher-quality answers. We discuss the advantages and disadvantages of each approach, and we report on an extensive empirical study to evaluate them in terms of their ability to predict the relevance of pages containing mathematical expressions. We also describe other alternative algorithms (e.g. keyword search only, etc.) and compare them against the proposed algorithms.

The contributions of this paper are as follows:

- We categorize existing approaches to match mathematical expressions, concentrating on two paradigms that consider the structure of expressions.

- We propose a representative system for each search paradigm.

```
<apply>
    <times/>
    <cn>
        2
    </cn>
    <apply>
        <plus/>
        <ci>
            x
        </ci>
        <apply>
            <times/>
            <cn>
                3
            </cn>
            <ci>
                y
            </ci>
        </apply>
    </apply>
</apply>
```

```
<mrow>
    <mn>
        2
    </mn>
    <mfenced>
        <mi>
            x
        </mi>
        <mo>
            +
        </mo>
        <mn>
            3
        </mn>
        <mi>
            y
        </mi>
    </mfenced>
</mrow>
```

Figure 1: Content MathML (left) vs. Presentation MathML (right) for $2(x + 3y)$

- We evaluate and compare the described approaches through detailed user studies in real scenarios.

This is the first attempt to describe and evaluate possible solutions in a principled way. Understanding the effectiveness of approaches to matching mathematical expressions is necessary for evaluating the further development of any mathematics retrieval algorithm. Hence, we believe the result of this study is an important step towards building a useful mathematics retrieval system.

In this paper we focus on the quality of results when matching mathematical expressions. Indexing and other optimization techniques to reduce query processing time or index size is out of the scope of this paper (Elsewhere, we extensively discuss such optimization techniques [14].). In this paper, we also do not address the problem of evaluating mathematical expressions, which is the goal of systems such as Wolfram Alpha [1], or Bing Math [11].

The rest of this paper is organized as follows. After formulating the mathematics retrieval problem more precisely in the next section, we describe related work in more detail in Section 3. In Section 4 we describe an approach based on matching structurally similar expressions, and in Section 5 we describe an approach based on matching expressions to templates. We present the results of our experiments and the comparison of alternative search approaches in Section 6. We finally conclude the paper with an indication of future work.

2. THE FRAMEWORK

In this section we present definitions for some general concepts. We also describe the search problem and any assumptions we make about the query and the results.

2.1 Definitions

Math Expression: A mathematical expression is a finite combination of symbols that is formed according to some context-dependent rules. Symbols can designate numbers (constants), variables, operations, functions, and other mathematical entities.

There are various ways to encode and represent a mathematical expression. Such approaches can be divided into two main groups:

1. Content-based: Semantics of symbols and their interactions are encoded. *Content MathML* [6] and *Open-*

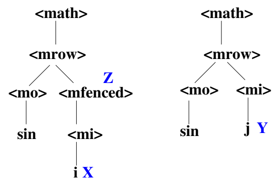

Figure 2: Two trees representing $sin(i)$ (left) and $sin\ j$ (right) in Presentation MathML.

Math [5] are XML-based markup languages that belong to this group.

2. Presentation-based: Expressions are encoded with respect to their appearance. Examples include images of expressions, *Presentation MathML* [6], and LaTeX.

Example 1. *Consider $2(x + 3y)$ as a simple expression. The Content MathML encoding for this expression is shown in Figure 1 (left), and the Presentation MathML is shown in Figure 1 (right). Presentation MathML contains some surface semantic information. For example, <mn> and <mi> indicate that 2 and 3 are numbers and x and y are variables, respectively. However, the multiplication operator is represented by the <times> tag in content markup, but it is invisible and hence not shown in presentation markup. On the other hand, parentheses are not encoded in content markup because they do not carry semantic information. The plus operator is represented by the <plus> tag in content markup, and its operands (x and 3y) are also clearly specified. Using presentation markup, the "+" symbol is shown where it appears in the expression, and even though it is marked as an an operator, its operands are not explicitly indicated.*

Presentation MathML is part of the W3C recommendation that is increasingly used to publish mathematics information on the web, and many web browsers support it [18]. There are various tools to translate mathematical expressions from other languages, including LaTeX, into Presentation MathML. Moreover, Presentation MathML expressions can be processed by parsers and other applications for XML documents. Hence, in this paper we assume mathematical expressions are encoded with Presentation MathML unless otherwise is specified.

DOM Tree: Documents with XML markup can be naturally expressed as ordered labelled trees, also called Document Object Model (DOM) trees. A DOM tree T is represented by $T = (V, E)$, where V represents the set of vertices and E represents the set of edges of T. A label $\lambda(n)$ is assigned to each node n, and Σ is the set of all possible labels. Two examples are shown in Figure 2.

A text document, such as a web page, that contains a mathematical expression is a *document with mathematical content*. The search goal is to retrieve such documents by querying their mathematical content.

2.2 Problem Formulation

Here we present a general definition for the search problem and the query language. Details of the query language and the way a match is defined are specific to a mathematics retrieval system. We describe several possible approaches in the following sections.

Query: The aim of a query is to describe a mathematical expression. Hence, a query is either a mathematics expression, or it specifies a pattern that describes one or more expressions.

Search problem: Given a query, the search problem is to find a list of relevant documents with mathematical content. A document is relevant if it contains an expression that matches the query.

The query and all mathematical expressions are encoded with Presentation MathML. Because forming queries directly with Presentation MathML is difficult, input devices such as pen-based interfaces and tablets [17, 25] or more widely-known languages such as LaTeX could be used instead to enter a query. Automatic tools can then be applied to translate queries to Presentation MathML. Hence, regardless of the user interface, we assume the query is eventually represented in the form of Presentation MathML. Thus, this approach is appropriate for the majority of the available mathematics information on the web.

2.3 Discussion

In the case of text retrieval, syntactic variants of query terms can be matched through stemmers and semantic variants can be matched through ontologies. These and similar tools can improve the results of search systems. Similarly, for mathematics retrieval using mathematical equivalence rules and transforming expressions to canonical forms accordingly (e.g. "$ab + ac$" and "$a(b + c)$") can improve search results. Nevertheless, such approaches are orthogonal to our algorithms and out of the scope of this paper.

Extending the query language to cover more complex cases can increase the usefulness of a search system. For example, allowing a query to consist of multiple mathematical expressions or a combination of mathematical expressions and text keywords can increase its expressive power. Including a (symbolic) mathematics engine to calculate the answer to a mathematical query can also be used to address some users' needs. However, in this paper our primary goal is to study the usefulness of the basic search paradigms and to compare them. While such extensions are potentially useful, the effectiveness of the basic search primitives should be proved first. Hence, in this paper we only focus on the basic search paradigms. An extended query language can then be studied by leveraging our findings.

3. RELATED WORK

In this section, we describe existing algorithms for mathematics retrieval systems in a framework that classifies them based on how they attempt to match expressions.

3.1 Exact Match

Some algorithms assume expressions are available only in images, and they try to match a given query by calculating the similarity of images [30, 31]. In the best case, the performance of such algorithms is similar to *ExactMatch* algorithms, which allow for very limited variation among the expressions returned. We describe and evaluate ExactMatch algorithms further in Section 6.

TexSN [27] is a textual language that can be used to normalize mathematical expressions into canonical forms. After that a search is performed to find mathematical expressions that exactly match a (normalized) query. MathQL [9] and MML Query [3] propose very detailed and formal query languages through which mathematical expressions are presented as sets of symbols. To perform a search, sets of expressions containing specific symbols are selected and then intersected using relational database operations. Einwohner and Fateman [7] propose a data structure and an algorithm for searching integral tables. In this approach, pre-calculated integrals are stored in a table. A requested integral matches an entry in the table if its integrand agrees with that of the table entry up to a choice of parameters, e.g. $\frac{1}{x^2+1}$ matches $\frac{1}{x^2+a}$. We characterize all of these approaches as *NormalizedExactMatch* algorithms, which we describe and evaluate further in Section 6.

As shown in Section 6, ExactMatch and NormalizedExactMatch perform poorly in retrieving web pages with mathematical content.

3.2 Approximate match

3.2.1 Substructure match

Sojka and Liska [26] propose another algorithm that first tokenizes expressions, where a token is a subtree of the expression. Each token is next normalized with respect to various rules (e.g. variables names are removed, number values are removed, or both), and multiple normalized copies are preserved. The resulting collection of tokens is then indexed with a text search engine. A query is similarly normalized (but not tokenized) and then matched against the index. Similarly, Egomath [19] tranforms math expressions into tokens (that represent subexpressions), and uses a text search system to index and query them. Regardless of the tokenization details, some structure information is missed by transforming an expression into bags of tokens, which affects the accuracy of results as shown later in this paper. MathWebSearch [15] is a semantic-based search engine for mathematical expressions. The query language is an extension to OpenMath, with some added tags and attributes, e.g. *mq:not, mq:and, mq:or*. Mathematical expressions are interpreted as prefix terms and are stored in a tree data structure called a substitution tree, where common prefixes are shared. A search is performed by traversing the tree. MathWebSearch can only process and index expressions encoded with Content MathML and OpenMath; presentation-based encoding is not well suited for use by this system. Schellenberg et al. [24] propose extending substitution trees to expressions with LaTeX encoding. Such approaches support exact matching of expressions well, but they support partial matching only when expressions share a common part at the top of the tree. Kamali and Tompa [12] propose to allow the common parts of two expressions to appear anywhere in the trees. We characterize such algorithms as *SubexprExactMatch* algorithms, and we show in Section 6 that their performance remains relatively poor.

3.2.2 Structure Similarity

Pillay and Zanibbi [23] propose an algorithm based on tree edit distance for combining the results of different math recognition algorithms. The goal of this approach is to enhance such algorithms to recognize hand-written expressions. Algorithms for retrieving general XML documents based on tree-edit distance have been proposed [16], and these could be adapted to match XML-encoded mathematical expressions. However, these approaches have not been thoroughly investigated for retrieving mathematical expressions. We propose an algorithm in this *SimSearch* class in Section 4 and show in Section 6 that it has a much better performance than other approaches such as exact match.

An alternative for matching based on structural similarity is to express a query in the form of a template, much as QBE does for querying relational data [33]. We describe how templates may be used to specify precisely where variability is permitted. We review our previous work on *PatternMatch* algorithm [13] in more detail in Section 5 and evaluate its performance in Section 6.

3.2.3 Keyword Similarity

The maturity of keyword search algorithms has motivated some researchers to use them for mathematics retrieval [28, 31]. Such approaches typically represent a mathematics expression as a bag of words, where each word represents a mathematics symbol or function. Youssef [28] proposes an algorithm based on the vector space model to rank mathematical expressions in response to a given query. To try to accommodate the specific nature of mathematics, alternative weighting schemes are considered instead of term frequency and inverse document frequency. Nguyen et al. [21] propose another algorithm that considers a semantic encoding (Content MathML) of expressions. Each expression is represented by several textual tags, after which standard keyword search algorithms are used to search mathematical expressions. This allows supporting queries that contain both keywords and mathematical expressions and using existing IR optimizations. As shown in Section 6, ignoring the structure significantly affects the performance.

3.3 Evaluating Math Retrieval Systems

Very few studies consider the problem of evaluating math retrieval systems in terms of satisfying user needs. This is partly due to the lack of a consensus on the definition of the relevance of math expressions, and partly due to the lack of a clear understanding of users' needs. Zhao et al. [32] report on the interviews of a small group of potential users to ascertain their needs. They conclude that users prefer to use keywords that describe an expression to search for it rather than specifying the expression (e.g. "binomial coefficient" instead of $\binom{n}{k}$). As we mentioned earlier, in many cases an expression is not described with keywords or the user is not aware of such keywords. Moreover, with math expressions more details can be specified (e.g. $\binom{n^2}{n}$). Finally, user-friendly interfaces for entering math expressions, such as pen-based devices, were not widely available at the time of the interview. Some search algorithms that compare images of expressions evaluate their systems in terms of success rate [31, 29]. In such cases, the success rate mostly captures the correctness of recognizing math expressions rather than their relevance. In other words, if an expression or subexpression is returned as the search result, and they exactly match, it is counted as a successful search.

To date, mathematics retrieval systems that perform approximate matching and then rank expressions based on their similarity to a query have not been analyzed in terms of their effectiveness: there are no experimental results comparing their ability to find relevant matches. However, the lack of their popularity may be a sign that in many situations they do not perform well.

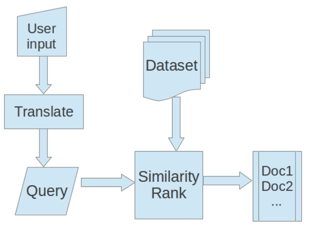

Figure 3: The flow of data in a mathematics retrieval system based on similarity ranking.

4. SIMILARITY SEARCH

The similarity function for mathematics considers only the mathematical content of a document, where each potentially relevant document contains at least one mathematics expression that matches the query. After the user inputs a query through a user interface, it is translated into Presentation MathML to be processed by the ranking algorithm. The result consists of a list of ranked documents sorted with respect to the similarity of their mathematical content to the query. Figure 3 shows the flow of data in this approach.

A general sketch for similarity search is presented in Algorithm 1. The definition of similarity between two mathematical expressions (Line 6) is a key concept that significantly affects such systems.

Just like other information retrieval systems, semantic similarity ranking is generally very useful, as it can better capture the intention of a user. Thus, the limited semantic information that is available (i.e., whether a symbol is a number, a variable, or an operator) should also be considered to calculate similarity in order to broaden the set of potentially matching expressions.

Unfortunately, a ranking function based on more expressive semantic similarity requires that the query and the expressions be semantically encoded using a markup language such as OpenMath or Content MathML. Hence, it requires more effort from the user to form a query semantically and also requires that content markup be used to publish mathematical expressions. As stated earlier, this is generally unavailable for retrieval from the web.

Algorithm 1 Similarity Search

```
1:  Input: Query $q$ and collection D of documents.
2:  Output: A list of documents ranked with respect to
    their similarity to $q$.
3:  Define list $L$ that is initially empty
4:  for each document $d \in D$ do
5:      for each math expression $E$ in $d$ do
6:          Calculate the similarity of $E$ and $q$ and store the
            result
7:      end for
8:      Calculate the similarity of $d$ and $q$ and store the result
        in $L$
9:  end for
10: Sort documents in $L$ with respect to the calculated sim-
    ilarities in descending order
11: return $L$
```

We now propose a similarity function that is based on tree edit distance [4], and define the similarity of a document to a math query accordingly. More specifically, we propose appropriate similarity functions to be used in Lines 6 and 8 of Algorithm 1.

Consider two ordered labelled trees $T_1 = (V_1, E_1)$ and $T_2 = (V_2, E_2)$ and two nodes $N_1 \in V_1 \cup P_\phi$ and $N_2 \in V_2 \cup P_\phi$ where P_ϕ is a special node with label ϵ. An edit operation is a function represented by $(N_1 \to N_2)$ where $(\lambda(N_1), \lambda(N_2)) \in (\Sigma \cup \epsilon) \times (\Sigma \cup \epsilon) - \{(\epsilon, \epsilon)\}$. The operation is a relabelling if $\lambda(N_1),\ \lambda(N_2) \neq \epsilon$. It is a deletion if N_1 is not the root of T_1 and $\lambda(N_2) = \epsilon$, where deleting N_1 makes the children of N_1 become the children of the parent of N_1 in place of node N_1. Finally, the operation is an insertion if $\lambda(N_1) = \epsilon$, where insertion is the mirror image of deletion. A transformation τ from T_1 to T_2 is a sequence of edit operations that transforms T_1 to T_2. To each edit operation $N_1 \to N_2$ we assign a cost $\omega(N_1 \to N_2)$. The cost of a transformation is the sum of the costs of its edit operations. The edit distance of T_1 and T_2 is defined as follows:

$$dist(T_1, T_2) = min\{cost(\tau) | \tau(T_1) = T_2\} \qquad (1)$$

We customize the cost of an edit operation $N_1 \to N_2$ for mathematical expressions as follows:

1. If $\lambda(N_1) = \lambda(N_2)$ then $\omega(N_1 \to N_2) = 0$.
2. If N_1, N_2 are leaf nodes and $\lambda(N_1) \neq \lambda(N_2)$ and $\lambda(parent(N_1)) = \lambda(parent(N_2))$ then $\omega(N_1 \to N_2) = C_{PL}(\lambda(parent(N_1)), \lambda(N_1), \lambda(N_2))$.
3. If N_1, N_2 are leaf nodes and $\lambda(N_1) \neq \lambda(N_2)$ and $\lambda(parent(N_1)) \neq \lambda(parent(N_2))$ then $\omega(N_1 \to N_2) = C_L(\lambda(N_1), \lambda(N_2))$.
4. If N_1, N_2 are not both leaf nodes and $\lambda(N_1) \neq \lambda(N_2)$ then $cost(N_1 \to N_2) = C_I(\lambda(N_1), \lambda(N_2))$.

In the above definition, C_I and C_L, and C_{PL} are static functions that assign values to an edit operation. Their values for various inputs are shown in Table 1. In this table, "<mi>", "<mn>", and "<mo>" represent variables, numbers, and operators respectively; α, β, and γ are constants whose values are set based on the following observations about math expressions (Some math retrieval systems normalize math expressions based on similar observations [19].). Typically, renaming variables affects the semantics less than changing math operators. Similarly, renaming a variable should be less costly than changing a variable to a number, and renaming non-leaf nodes should be more costly that renaming leaf nodes. Therefore, we set $\alpha \leq \beta \leq \gamma$.

Example 2. *Consider nodes X and Y in Figure 2. $X \to Y$ is a relabelling. The label of their parents, <mi>, states that they are variables. According to Table 1, C_{PL}("<mi>", "i", "j") $= \alpha$. Hence, $\omega(X \to Y) = \alpha$. Also, $Z \to P_\phi$ is a deletion and $\omega(Z \to P_\phi) = \gamma$. The edit distance between the two trees is equal to $\alpha + \gamma$.*

Consider two mathematical expressions E_1 and E_2 represented by trees T_1 and T_2. The similarity of the two expressions is calculated as follows:

$$sim(E_1, E_2) = 1 - \frac{dist(T_1, T_2)}{|T_1| + |T_2|} \qquad (2)$$

where $|T|$ is the number of nodes in tree T.

There are many algorithms for calculating the edit distance between two trees. We use RTED [22] to calculate the tree edit distance.

Value of C	Condition
$C_{PL}(``<mi>", x, y) = \alpha$	$x \neq y$
$C_{PL}(``<mn>", x, y) = \alpha$	$x \neq y$
$C_{PL}(``<mo>", x, y) = \alpha$	$x \neq y$ and $\{x, y\} \in \{+, -\}$
$C_L(\epsilon, x) = \beta$	
$C_L(x, \epsilon) = \beta$	
$C_L(x, y) = 2\beta$	$x \neq y$
$C_I(\epsilon, x) = \gamma$	
$C_I(x, \epsilon) = \gamma$	
$C_I(x, y) = 2\gamma$	$x \neq y$

Table 1: Examples of various cost values assigned to edit operations

Assume document d contains mathematical expressions $E_1 \ldots E_n$. The rank of d for a query Q is calculated with the following formula:

$$docRank(d, Q) = max\{sim(E_i, Q) | E_i \in d\} \quad (3)$$

that is, a document's score is equal to the similarity of the most similar expression in that document.

5. PATTERN SEARCH

An alternative to similarity ranking is to specify a template as the query and return expressions that match it as the search result [13]. This allows flexible matching of expressions but in a controlled way (as distinct from the similarity ranking where the user has less control on approximate matching of expressions). For example, the user can specify that she is looking for $[E]^n$ where n is a number and $[E]$ is an expression that contains $\sin(x)$. This capability is not supported by any exact matching algorithm, and the similarity search may not rank relevant expressions high enough. The approach is analogous to querying a collection of strings by specifying a partial context-free grammar, and returning strings that can be parsed by the grammar. Similarly, a template can be defined using wildcards as non-terminals, and regular expressions to describe their relationships. For example, $\sqrt{[V]}$, where $[V]$ is a wildcard that matches any variable, can be used to search for expressions that consist of the square root of a variable.

According to this paradigm, the user has more power to direct the search; hence the results are expected to be more relevant. However, the variety of wildcards and operations that are available to specify a template may result in a complex query language, which requires more effort from the user.

To find a relevant document, the user starts with a pattern to be searched. It may be necessary to tune the query, as the initial pattern may not correctly model the expression the user is looking for or it may be too general or too specific. After the results are shown, she tunes the pattern until she finds a relevant answer. A diagram of the data flow for this search paradigm is shown in Figure 4. Algorithm 2 presents a general sketch for this search paradigm.

In some cases, combining similarity ranking and structured search is useful. For example, assume a user is looking for expressions that contain the square root of an expression E such that E is similar to $sin(x)$. In this case, $\sqrt{sin(x)}$ is a better match than $\sqrt{sin(\frac{x}{2}+1)}$, and while the latter still complies with the pattern, $\sqrt{sin(x)}+1$ is not match. This search paradigm has been adopted in other contexts. Examples include XQuery with full-text capability [2] and other

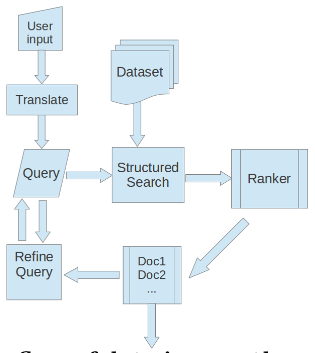

Figure 4: The flow of data in a mathematics retrieval system based on pattern matching.

keyword search features on structured data [10]. Adding this capability to a search system can increase its flexibility to capture relevant results.

In conclusion, this search paradigm is suitable for specialized search where the user can be expected to put more effort to form a query and get better results in return.

Algorithm 2 Structured Search

```
 1: Input: Query $q$ and collection D of documents.
 2: Output: A ranked list of documents that match the query.
 3: Define list $L$ that is initially empty
 4: for each document $d \in D$ do
 5:   for each math expression $E$ in $d$ do
 6:     if $E$ matches $q$ then
 7:       put $d$ in $L$
 8:     end if
 9:   end for
10: end for
11: Sort documents in $L$ with respect to ranking criteria
12: return $L$
```

5.1 A Model Query Language

A query is expressed as a pattern consisting of a mathematical expression augmented with *wild cards*, optional parts, and constraints in the form of *where clauses*. A query matches an expression (Algorithm 2-Line 6) as follows. A wild card represents a slot that will match any subtree of the appropriate type, where $[V_i]$ matches any variable, $[N_i]$ matches any number, $[O_i]$ matches any operator, and $[E_i]$ matches any expression. A wild card's index i is an optional natural number such that if two or more wild cards share the same type and index, they must match identical subtrees. Wild cards with no index are unconstrained.

EXAMPLE 3. *The query $x^{[N_1]} - y^{[N_1]}$ matches $x^2 - y^2$ and $x^5 - y^5$ but not $x^2 - y^3$, whereas either of the queries $x^{[N_1]} - y^{[N_2]}$ or $x^{[N]} - y^{[N]}$ matches all three.*

Optional parts are enclosed by braces and they may appear in some matching expressions.

EXAMPLE 4. *$x^2\{+[N]\}$ matches x^2 and $x^2 + 1$ but not $x^2 + y$ or $x^2 - 1$.*

Constraints can be specified for wild cards in a query using a "where" clause, as follows:

- Number wild cards can be constrained to a specific range or to a domain, which can be specified using a context-free grammar.
- Variable wild cards can be constrained to a restricted set of possible names.
- Operator wild cards can be constrained to a restricted set of operators.
- Expression wild cards can be constrained to contain a given subexpression, which can in turn include further wild cards and constraints.

Example 5.

- *Query "*$[E]^2[O1]3$ **where** $O \in \{+,-\}$*" matches* x^2+3 *and* $(x+1)^2-3$ *but not* $x^2 \times 3$.
- *Query "*$x^{[N1]}$ **where** $1 \le N1 \le 5$*" matches* x^2 *but not* x^9 *or* x^{-1}.
- *Query "*$[E1]-2$ **where** $[E1]$ *contains* x^2*" matches* x^2-2 *and* $\log(x^2+3y)-2$ *but not* $x-2$ *or* y^2-2.
- *Query "*$[E1]$ **where** $E1$ *contains* $\log_2([V])$*" matches all expressions that include a base 2 logarithm of a variable.*

In our experiments we assume a pattern does not contain a similarity constraint. Otherwise, pattern search would be a generalized form of the similarity search approach, which makes it hard to compare them. Moreover, ranking documents with respect to a pattern query that contains multiple similarity constraints is a complex problem that should be addressed after the more basic problem of capturing the similarity of two math expressions (discussed in this paper) is addressed. This problem is a direction of our future work.

5.2 Query Processing

A query is processed by trying to match it against the stored expressions by parsing them with respect to the query. Each document that contains a match is included in the search result.

While similarity ranking is in fact an information retrieval approach to the problem, pattern search resembles a database look-up. Therefore, the result of this search paradigm is a list of documents with expressions that match the query. To rank documents in the list (Algorithm 2-Line 11), a ranking criterion should be considered. In our implementation we sort results with respect to the sizes of the matched expressions in increasing order.

6. EXPERIMENTS

In this section we present the results of our empirical evaluation of the described approaches.

6.1 Alternative Algorithms

In our experiments we consider the following specific algorithms:

- *TextSearch:* The query and expressions are treated as bags of words (nodes' labels in their DOM trees). A standard text search algorithm is used for ranking documents according to a given query[1].
- *ExactMatch:* An expression is reported as a search result only if it matches a given query exactly. Results are ranked with respect to the alphabetic order of the name of their corresponding documents.
- *NormalizedExactMatch:* Some normalization is performed on the query and on the stored expressions: in particular, we ignore specific numbers, variables, and operators by removing all leaf nodes. The normalized expressions are searched and ranked according to the ExactMatch algorithm.
- *SubexprExactMatch:* An expression is returned as a search result if at least one of its subexpressions exactly matches the query. Results are ranked by increasing sizes of their DOM trees.
- *NormalizedSubExactMatch:* Normalization is done on the query and on the stored expressions as for NormalizedExactMatch, and an expression is returned as a search result if one of its normalized subexpressions matches the normalized query.
- *MIaS*: As described in Section 3, subtrees are normalized and transformed into tokens and a text search engine is used to index and retrieve them [26].
- *SimSearch:* Expressions are matched against a query according to the algorithm described in Section 4.
- *PatternSearch:* Expressions are matched against a pattern according to the algorithm described in Section 5. Like SubexprExactMatch, results are ranked with respect to the sizes of their DOM trees.

Note that among the above algorithms, the results of ExactMatch are subsets of the results of TextSearch, NormalizedExactMatch, and SubexprExactMatch.

6.2 Experiment Setup

6.2.1 Data Collection

For our experiments we use a collection of web pages with mathematical content. We collected pages from the Wikipedia and DLMF (Digital Library of Mathematics Functions) websites. Wikipedia pages contain images of expressions annotated with equivalent LaTeX encodings of the expressions. We extracted such annotations and translated them into Presentation MatchML using Tralics [8]. DLMF pages use Presentation MathML to represent mathematical expressions. Statistics summarizing this dataset are presented in Table 2.

6.2.2 Queries

To evaluate the described algorithms we prepared two sets of queries as follows.

- *Interview:* We invited a wide range of students and researchers to participate in our study. They were asked

	Wikipedia	DLMF	Total
Number of pages	44,368	1,550	45,918
Number of expressions	611,210	252,148	863,358
Average size of expressions	28.3	17.6	25.2
Maximum size of expressions	578	223	578

Table 2: Dataset statistics

[1]We used Apache Lucene in our implementation.

to try our system and search for mathematical expressions of potential interest to them in practical situations. They could also provide us with their feedback about the quality of results after each search.

- *Mathematics forum:* People often use mathematics forums in order to ask a questions or discuss math-related topics. Many discussion threads can be described with a query that consists of a single math expression. Usually, by reading the rest of the thread and responses, the exact intention of the user is clear. This allows us to manually judge if a given expression, together with the page that contains it, can answer the information need of the user who started the thread. We manually read such discussions and gathered a collection of queries.

The precise query formulations for PatternSearch were created by one of the authors. Thus the experimental results reflect search environments in which queries are formed reasonably well by an experienced user.

Table 3 summarizes statistics about the queries, where the number of nodes in the query tree is used to represent query size. For the sake of reproducibility and as a basis for further evaluation of various search paradigms, both the dataset and the complete set of queries can be obtained from the authors upon request[2].

6.2.3 Methodology

For each query we use each algorithm to search the dataset, and only consider the top 10 results.

The way we collected queries ensures that a user's information needs are clear, which allows us to judge if a match is actually relevant or not. Searches for which we do not have a user's relevance feedback (i.e., Forum queries) require that we manually judge results. Hence, for Forum queries we consider discussion threads that clearly describe an information need with no ambiguity. For example a discussion thread might start with this question: "prove that $F_n^2 - F_{n-1}F_{n-2} = (-1)^{n-1}$ where F_n is the n^{th} Fibonacci number". A search result page is considered relevant if it satisfies the information need that is inferred from the thread. If a page contains data that can be clearly used to answer the query, we judge it as relevant. Note that a page may contain an exact match to a query, but it still does not answer the information need, hence we assume it is irrelevant (e.g. if a page contains the same expression as in the previous example, but F_n is not a Fibonacci number, or it does not contain any information that helps to prove it.).

As mentioned earlier, for PatternSearch the query may be refined repeatedly unless appropriate results are returned or the query is refined a certain number of times and the user gives up (Figures 4). Hence, unless otherwise specified, the results for PatternSearch are presented with respect to the final refined query. For other algorithms however, refining a query is often not necessary or effective, and the results are shown for the original query.

6.2.4 Evaluation measures

NFR: A search fails if fewer than 10 results are returned (including nothing returned), and none of them is relevant.

[2]The query collections with examples of matching documents are publicly available at *http://db.uwaterloo.ca/mathretrieval/queries.xhtml.*

	Interview	Math Forum	Total
Number of queries	45	53	98
Average size of queries	14.2	23.8	19.4

Table 3: Query statistics

Non-Failure-Rate (NFR) is the number of searches that do not fail divided by the total number of searches:

$$NFR = \frac{|\{q \in Q | searching\ q\ does\ not\ fail\}|}{|Q|} \quad (4)$$

MRR: The rank of the first correct answer is a representative metric for the success of a mathematics search. Hence, for each search we consider the *Reciprocal Rank* (RR), that is, the inverse of the rank of the first relevant answer. For example if the first correct answer is ranked second, the reciprocal rank of the search is $\frac{1}{2}$. The *Mean Reciprocal Rank* (MRR) is the average reciprocal rank for all queries:

$$MRR = \frac{1}{|Q|} \sum_{q \in Q} \frac{1}{C(q)} \quad (5)$$

where Q is the collection of queries, and $C(q)$ is the rank of the first relevant answer for query q.

If no relevant document is among the top 10 results, we optimistically assume the search did not fail, and the rank of the first relevant document is 11. If a search fails, we do not include it for calculating MRR.

Rewrite-Rate: It often happens that a search is not successful, and a user must rewrite the query to find a relevant result. For each search algorithm, starting from an initial query, we logged how many times a query was rewritten to obtain a relevant answer among the top 10 results. We assume that the user gives up after five tries, and the search fails or no relevant result is found. The average number of rewrites for all queries is the *rewrite rate* of an algorithm.

Other measures such as *Mean Average Precision* (MAP) could alternatively be considered. However, for our data and query collections, MRR seems to be a better choice. In most cases, there are a few (often one) relevant documents for the query. Hence, MRR can better reflect the accuracy of algorithms. Recall that some of the baselines (e.g. pattern search and substructure search) deploy database operators to perform a search, while some other ones (e.g. structural similarity and keyword search) use IR techniques. Hence, because such approaches are inherently different, it is important to consider measures that fairly compare them. MRR and NFR together provide an indicative measure of the accuracy of such algorithms.

6.3 Evaluation Results

6.3.1 Correctness

The NFR and MRR for each algorithm are presented in Tables 4 and 5 for the Forum and Interview queries, respectively. As the results suggest, PatternSearch and SimSearch have high NFR and also high MRRs. PatternSearch has a higher MRR because irrelevant expressions are less likely to match a carefully formed pattern. On the other hand, SimSearch has a slightly higher NFR because in some cases even an experienced user may not be able to guess the pattern that will yield a correct answer. Furthermore, the next section shows that a template pattern may need to be modified several times to capture a relevant result.

Algorithm	NFR	MRR	p-value
SimSearch	100%	0.74	-
PatternSearch	90%	0.86	0.171
MiaS	94%	0.46	*0.002*
TextSearch	100%	0.19	*0.000*
ExactMatch	13%	1	0.186
NormalizedExactMatch	34%	0.46	*0.007*
SubexprExactMatch	18%	0.94	0.173
NormalizedSubExactMatch	41%	0.43	*0.004*

Table 4: Algorithms' performance for Forum Queries.

Because MRR is only calculated when the search does not fail, ExactMatch has a high (in fact, perfect) MRR. However, in most cases, there is no expression that exactly matches the query, and hence no result is produced and this algorithm fails. SubexprExactMatch has a slightly better NFR, but it is still too low to satisfy many users' needs. This implies that often there are no relevant expressions or subexpressions that exactly match the query. However, in instances where a matching expression or subexpression exists, it is ranked highly by ExactMatch and SubexprExactMatch. Normalizing expressions, as done in NormalizedExactMatch and NormalizedSubExactMatch, further increases the NFR, but it also increases the chances that irrelevant expressions are matched. Because such algorithms do not offer an effective ranking algorithm, in many cases the most relevant results are not among the top 10 results.

Note that although the MRR of SimSearch is lower than ExactMatch and SubexprExactMatch, it has a much higher NFR as it produces some results for all queries. If we only consider queries for which there is an exact match (so ExactMatch produces at least one answer), SimSearch has an MRR that is not significantly different from that of ExactMatch. The reason is that in such cases, the structural similarity for documents that contain exact matches is 1, and such documents are ranked at the top of the results.

TextSearch has a very low MRR. Because this algorithm ignores the structure, it often does not rank a correct answer highly enough against many irrelevant expressions with similar MathML tags and symbols but different structures.

Note that, although we reformulate queries only for pattern search, the structural similarity search produces results that are comparable with the results of well-formulated pattern queries. ExactMatch or NormalizedExactMatch are essentially pattern search with poorly formed queries. As shown, such algorithms produce poor results.

To show the statistical significance of the results, we use a Student's t-test on the reciprocal ranks of the queries. For each algorithm, we test whether there is a statistical difference between the reciprocal ranks of its produced results and that of SimSearch. We consider a one-tailed t-test for paired samples (i.e. only non-failed searches are considered). As the data in Tables 4 and 5 suggest, there are significant differences (at the 0.05 significance level) between the results of SimSearch and all algorithms except PatternSearch, ExactMatch, and SubexprExactMatch (for Forum queries). The reason is that in cases that such algorithms do not fail, SimSearch ranks relevant results equally well.

6.3.2 Query Rewriting

To compare how much effort is required from the user to perform a search, we look at PatternSearch in terms of its rewrite rates. Assume a user is looking for $\sum_{i=1}^{10} 2^i$, and sup-

Algorithm	NFR	MRR	p-value
SimSearch	100%	0.78	-
PatternSearch	76%	0.96	0.084
MiaS	90%	0.63	*0.014*
TextSearch	100%	0.23	*0.000*
ExactMatch	15%	1	0.211
NormalizedExactMatch	50%	0.58	*0.005*
SubexprExactMatch	30%	0.62	*0.044*
NormalizedSubExactMatch	60%	0.44	*0.009*

Table 5: Algorithms' performance for Interview Queries.

pose the only relevant match to this query is $\sum_{i=1}^{n} a^i$. The edit distance between the corresponding DOM trees is relatively low, and thus this answer is ranked highly by the SimSearch algorithm. For PatternSearch, however, if the user forms a query with no wild cards, it performs similarly to ExactMatch, and the correct answer is not found. The following is a plausible sequence of query refinements before an answer is found:

$$\sum_{i=1}^{10} 2^i \rightarrow \sum_{[V1]=1}^{10} 2^{[V1]} \rightarrow \sum_{[V1]=1}^{[N1]} [N2]^{[V1]} \rightarrow \sum_{[V1]=1}^{[N1]} [V2]^{[V1]}$$

While the rewrite rate of SimSearch is always 1 (as no query rewriting is required), PatternSearch has an average rewrite rate of 2.2 and 1.45 for the Forum and Interview queries respectively. As the results suggest, when using PatternSearch, each query may well be refined to obtain relevant results, and hence the user must invest more effort to find relevant documents.

6.3.3 Summary

In summary, simply viewing mathematics expressions as if they were conventional document fragments, as represented by TextSearch, or not allowing variations in matched expressions or subexpressions, as represented by ExactMatch and SubexprExactMatch, leads to extremely poor search results. On the other hand, SimSearch and PatternSearch perform very well: much better than the other algorithms that ignore the structure or perform exact matching only. PatternSearch may perform slightly better than SimSearch, but the user will likely need to spend more time to tune a query pattern when using this algorithm. Reassuringly, these results are consistent across the two sources of queries.

7. CONCLUSIONS AND FUTURE WORK

Given a mathematics expression, finding pages with relevant mathematical content is an important problem that is the basis of many mathematics retrieval systems. Correctly predicting the relevance of mathematical expressions is a core problem that should be addressed in order to develop useful retrieval systems.

We characterized several possible approaches to this problem, and we elaborated two working systems that exploit the structure of mathematical expressions for approximate match: structural similarity search and pattern matching. We empirically showed that these two search paradigms outperform other search techniques, including the ones that perform exact matching of (normalized) expressions or subexpressions and the one that performs keyword search. We also showed that it takes more effort from the user to form queries when doing pattern search as compared to similarity search, but when relevant matches are found they are ranked somewhat higher. So in conclusion, structural similarity search seems to be the best way for general users to search for mathematical expressions, but we hypothesize that pattern search may be the preferred approach for experienced users in specific domains.

In this paper we focussed on the usability of answers and how well a search system can find relevant documents for a given query. Others may wish to re-evaluate these results using more controlled methods for assessing relevance. The study should next be extended in an ongoing effort to include new approaches as they are developed. Optimizing the proposed search techniques in terms of query processing time and index size is a separate direction [14]. Based on the results of this paper, more complex query languages can also be developed to accommodate queries that consist of multiple mathematical expressions supplemented by textual keywords that might match other parts of relevant documents, or pattern queries with one or more similarity constraints.

NTCIR is an international initiative to create a public and shared infrastructure to facilitate research in Math IR. It aims to provide a test collection and a set of math tasks. As a part of our future research, we plan to use this data (which is not yet available) to further evaluate the discussed algorithms.

8. ACKNOWLEDGEMENTS

We acknowledge financial support from NSERC, Mprime, and the University of Waterloo, and we thank the referees for their suggestions.

9. REFERENCES

[1] www.wolframalpha.com.

[2] S. Amer-Yahia, C. Botev, and J. Shanmugasundaram. TeXQuery: a full-text search extension to XQuery. In *WWW*, pages 583–594, 2004.

[3] G. Bancerek. Information retrieval and rendering with MML Query. In *MKM*, pages 266–279, 2006.

[4] P. Bille. A survey on tree edit distance and related problems. *Theor. Comput. Sci.*, 337(1-3):217–239, 2005.

[5] S. Buswell, O. Caprotti, D. P. Carlisle, M. C. Dewar, M. Gaëtano, and M. Kohlhase, editors. *The OpenMath Standard, Version 2.0.* The OpenMath Esprit Consortium, 2004.

[6] D. Carlisle, P. Ion, and R. Miner. *Mathematical Markup Language (MathML) Version 3.0.* W3C Recommendation, 2010.

[7] T. H. Einwohner and R. J. Fateman. Searching techniques for integral tables. In *ISSAC*, pages 133–139, 1995.

[8] J. Grimm. *Tralics, A LaTeX to XML Translator.* INRIA, 2008.

[9] F. Guidi and I. Schena. A query language for a metadata framework about mathematical resources. In *MKM*, pages 105–118, 2003.

[10] V. Hristidis, L. Gravano, and Y. Papakonstantinou. Efficient IR-style keyword search over relational databases. In *VLDB*, pages 850–861, 2003.

[11] S. Kamali, J. Apacible, and Y. Hosseinkashi. Answering math queries with search engines. In *WWW*, pages 43–52, 2012.

[12] S. Kamali and F. W. Tompa. Improving mathematics retrieval. In *DML*, pages 37–48, 2009.

[13] S. Kamali and F. W. Tompa. A new mathematics retrieval system. In *CIKM*, pages 1413–1416, 2010.

[14] S. Kamali and F. W. Tompa. Structural similarity search for mathematics retrieval. In *CICM*, 2013.

[15] M. Kohlhase and I. A. Sucan. A search engine for mathematical formulae. In *AISC*, pages 241–253. Springer, 2006.

[16] C. Laitang, M. Boughanem, and K. Pinel-Sauvagnat. XML information retrieval through tree edit distance and structural summaries. In *AIRS*, pages 73–83, 2011.

[17] S. Maclean and G. Labahn. A new approach for recognizing handwritten mathematics using relational grammars and fuzzy sets. *IJDAR*, pages 1–25, 2012.

[18] B. Miller. 3 years of DLMF: Web, math & search. In *CICM*, 2013.

[19] J. Misutka and L. Galambos. System description: Egomath2 as a tool for mathematical searching on wikipedia.org. In *Calculemus/MKM*, pages 307–309, 2011.

[20] R. Munavalli and R. Miner. Mathfind: a math-aware search engine. In *SIGIR*, pages 735–735, 2006.

[21] T. T. Nguyen, K. Chang, and S. C. Hui. A math-aware search engine for math question answering system. In *CIKM*, pages 724–733, 2012.

[22] M. Pawlik and N. Augsten. RTED: A robust algorithm for the tree edit distance. *PVLDB*, 5(4):334–345, 2011.

[23] A. Pillay and R. Zanibbi. Intelligent combination of structural analysis algorithms: Application to mathematical expression recognition. In *PenMath*, 2009.

[24] T. Schellenberg, B. Yuan, and R. Zanibbi. Layout-based substitution tree indexing and retrieval for mathematical expressions. In *DRR*, 2012.

[25] E. S. Smirnova and S. M. Watt. Communicating mathematics via pen-based interfaces. In *SYNASC*, pages 9–18, 2008.

[26] P. Sojka and M. Líska. The art of mathematics retrieval. In *ACM Symposium on Document Engineering*, pages 57–60, 2011.

[27] A. Youssef. Search of mathematical contents: Issues and methods. In *IASSE*, pages 100–105, 2005.

[28] A. Youssef. Methods of relevance ranking and hit-content generation in math search. In *Calculemus/MKM*, pages 393–406, 2007.

[29] R. Zanibbi and D. Blostein. Recognition and retrieval of mathematical expressions. *IJDAR*, 15(4):331–357, 2012.

[30] R. Zanibbi and L. Yu. Math spotting: Retrieving math in technical documents using handwritten query images. In *ICDAR*, pages 446–451, 2011.

[31] R. Zanibbi and B. Yuan. Keyword and image-based retrieval of mathematical expressions. In *DRR*, pages 1–10, 2011.

[32] J. Zhao, M.-Y. Kan, and Y. L. Theng. Math information retrieval: user requirements and prototype implementation. pages 187–196, 2008.

[33] M. M. Zloof. Query-by-Example: the invocation and definition of tables and forms. In *VLDB*, pages 1–24, 1975.