8-1-2013

Math expression retrieval using symbol pairs in layout trees

David Stalnaker

Math Expression Retrieval Using Symbol Pairs in Layout Trees

by

David Stalnaker, B.S.

THESIS

Presented to the Department of Computer Science

Golisano College of Computer and Information Sciences

in Partial Fulfillment

of the Requirements

for the Degree of

Master of Science

Rochester Institute of Technology

August 2013

Math Expression Retrieval Using Symbol Pairs in Layout Trees

APPROVED BY

SUPERVISING COMMITTEE:

Dr. Richard Zanibbi, Chair

Dr. Bo Yuan, Reader

Dr. Zack Butler, Observer

Math Expression Retrieval Using Symbol Pairs in Layout Trees

David Stalnaker, M.S.
Rochester Institute of Technology, 2013

Advisor: Dr. Richard Zanibbi

We have developed a layout-based math retrieval system by indexing on pairs of symbols in mathematical expressions. Existing approaches to layout-based retrieval include tree edit distance-based matching on MathML trees (Kamali and Tompa, 2013) and longest common subsequence matching in LaTeX strings (Kumar et al., 2012). In our work, we compare our new layout-based retrieval method with a math retrieval system built using the conventional text-based retrieval system Lucene (Zanibbi and Yuan, 2011), as such systems are commonly used for math search. We show that the search results returned by our system are scored by participants in a study as significantly more similar than those of the comparison system and that our system is fast enough to be used in real time.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

While search for text is a well-studied problem, math search is less mature. Search for mathematical expressions is difficult for a number of reasons. There are numerous representations for expressions, including: LaTeX, (content and presentation) MathML, Mathematica, and various forms of rendered output (PDF, images). Mathematical expressions are generally expressed as a tree structure (whereas text is linear), which necessitates more complex algorithms for matching. Partially matched expressions are difficult to rank because small syntactic changes can have large semantic meaning and vice versa.

The main motivation for a math search engine is to facilitate learning. Upon coming upon an unfamiliar mathematical expression, a student could search for it and learn what it means. Similarly, a researcher could find papers by searching the math contained within them [17]. Math search could also be used when mathematical queries are detected in an existing search engine [2]. One problem with a math search engine is that the representations of mathematics are complex and students are unlikely to be familiar with them. However, our group has developed a separate tool, m_{in} [11], which recognizes hand-written mathematics and can be used as an input for our search engine. Figure 1.1 shows the interaction between these two systems.

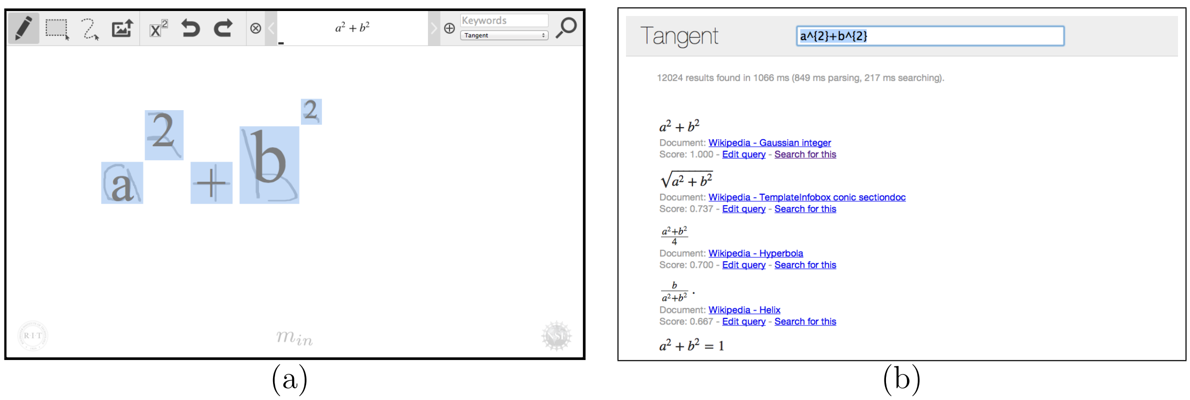


Figure 1.1: Screenshots of the m_{in} handwritten math recognition tool (a), and the results page of our Tangent search engine (b).

An inverted index is the tool used most commonly in text search. It is a lookup table from words to the documents that contain them. This idea has been modified to allow for word locality, phrase searching, index compression, and distributed indexing. With these and other improvements, systems can be built to quickly search massive corpora (notably the Internet) [20].

In our system, Tangent, we apply the concept of the inverted index to math search, indexing directly on the structure of the equation instead of a linear string representation thereof. The idea is that encoding the structure of the expression directly in the index will yield more partial matches, providing more relevant results. By using similar data structures and algorithms as in text search, overall system speed should be comparable to those systems. This leads us to this hypothesis:

> **Implementing a math expression search system using an inverted index on symbol pairs from layout trees will: 1) yield more relevant results than text-based retrieval, and 2) be fast enough for real time use.**

We conducted two experiments to test our hypothesis: a human study wherein we asked participants to score results from our system and a text search-based comparison system, and a performance analysis of our system. The results of these experiments support both parts of the hypothesis. The search results for our system scored significantly higher than those of the comparison system and a performance analysis showed the system to be usable with a large corpus.

Chapter 2

Background

Information retrieval is a well-established field in computer science, but the majority of the research has been focused on text retrieval with significantly less work in math retrieval. In this chapter, we will describe the basic concepts used in text retrieval, and methods to compress the indexes used therein. We will then describe several existing math search systems and discuss the general goals of math retrieval. Finally, we will describe MathML, a common representation for mathematical expressions.

2.1 Information Retrieval

As computers appeared, one of the important tasks that emerged was storing and searching large sets of documents. The inverted index was an early and natural development in this field. It can be thought of like an index in a book: it is a structure that maps each term to a list of documents that contains it. In its most simple form, an inverted index only allows us to find the documents that contain a given term [7, 10].

The inverted index is still the basis of almost every document retrieval system. Countless methods have been developed to make the structure more powerful. By looking up multiple terms and taking the union or intersection of the returned set of documents, we can handle multi-term queries. By storing the frequencies of terms in each document, we can order the documents by this frequency. By storing the location(s) of each term in each document, we can search for exact and inexact phrases.

There are numerous ways in which to rank the relevance of matched documents. The simplest is the frequency of a matched term in the document, or term frequency. A common metric, TF-IDF, makes use of this and the Inverse Document Frequency. The IDF is the inverse of the number of documents that contain the term. Including this in a ranking metric allows us to bias results towards infrequently used words, which works well because common words are usually less discriminative. Other measures that might be used as part of a ranking metric are the number of documents and terms in the index, and the term frequency.

2.2 Index Compression

While storing document references and counts in the index as 32- or 64-bit integers is convenient, it is wasteful of space. There are numerous ways to compress the inverted lists. In addition to the decreased storage need, this also usually brings a performance improvement, as the added cost of decompression is negligible compared to the decreased cost of disk reads. Most of the compression methods discussed will remain applicable with the modifications to the inverted file that we make for math search. The following is a summary of Zobel and Moffat's discussion of the same topic in [20].

One way to compress an index is to change the representation of an integer. The appropriate representation depends on the probability distribution of the values. If the values are very low, a unary representation is viable, wherein the number n is is represented by $n-1$ "1" bits followed by a "0" bit. This has the property that "1" is encoded with just 1 bit, but quickly becomes inefficient for larger values. Unary encoding is most efficient when $P[n] \approx \frac{1}{2^n}$, where n is the number being encoded and P is the probability distribution of the numbers to be encoded. In practice this means n must be very small.

Elias' gamma code allows for more efficient encoding of larger numbers. The value n is factored into $2^e + d$, where $e = \lfloor \log_2 n \rfloor$. The encoding is then $e+1$ encoded in unary followed by d encoded in binary (using e bits). It is most efficient when $P[n] \approx \frac{1}{2n^2}$.

While bitwise encoding schemes can lead to a great reduction in index size, they ignore the fact that computers can generally only address memory at the byte level. It is thus often a better idea to encode our index in whole bytes, for reason of both performance and ease of programming. An effective method is to break the value into 7-bit chunks. If the value fits in 7 bits, encode it in one byte with the high bit set to "0". Otherwise, encode the first 7 bits with a leading "1", and iterate with the next byte. This is more wasteful than the bitwise methods for small values, but in practice provides a good balance between compression rate and efficiency.

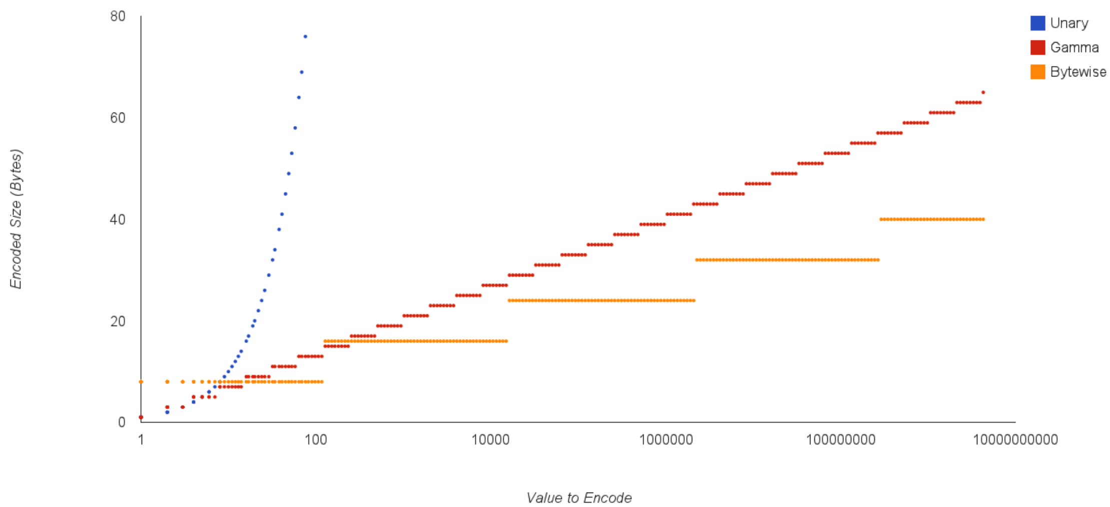


Figure 2.1: Size of encoded values. Input values are on a log scale, and the unary encoding is far off the scale for large values.

These encoding schemes work well for frequency counts because they are generally very small. Document IDs, however, are distributed uniformly and thus do not compress well. An effective way to solve this is to sort the document identifiers in each inverted list and store instead the differences between each element and its predecessor. In a list for a common word (which will make up a sizable portion of the index), these differences will be very small and compress easily.

2.3 Math Retrieval

There are several systems for math search that encode expressions as text and use existing text search systems for indexing and searching [18]. The key effort here is the process of "linearizing" the math expression for input into the text search system, both for documents to index and queries to the system. Zanibbi and Yuan's system [19] indexes individual LaTeX expressions by tokenizing the expression and mapping each symbol to a text representation. In this example from Miller [8], the LaTeX expression `x^{t-2} = 1` is represented as `x BeginExponent t minus 2 EndExponent Equal 1`. Expressions are then inserted into the Lucene search engine and queried with LaTeX expressions processed in the same manner.

Sojka and Líška's MIaS system [14, 15] operates on similar principles but differs in several areas. It is a full-text search system that indexes both math expressions and the surrounding text. Instead of single LaTeX expressions, MIaS indexes XHTML documents containing MathML expressions. The overall architecture of the system is similar, with linearized expressions being inserted into a text search system (it too uses Lucene). Munavalli and Miner's MathFind [9] is another text-search based system, indexing text and MathML expressions together by converting the expressions to text strings.

The key problem with this approach is that the text search has limited information on the structure of the expression. It is thus difficult to align the qualities of a good textual match with the qualities of a good match of the expression. The advantages are that it works reasonably well in practice, is easy to implement, and benefits directly from the decades of research in document retrieval. Our approach will use the same concept of the inverted index, but will index on pairs of symbols to better encode the structure of expressions.

There have been several approaches which look at the problem more directly. Kamali and Tompa describe a system based on efficient exact matching of MathML trees [3]. They use an index in which identical subtrees are shared between all expressions, which allows for much better performance. Inexact matching is enabled through the use of wildcards, where each wildcard can be a number, variable, operator, or expression (which can itself contain wildcards). More recently, they describe a system that uses a similar index but uses a tree-edit distance for searching and ranking, which allows for inexact matching without user-specified wildcards [4]. Calculating the tree-edit distance for each expression in the index would be too expensive, but through a combination of early termination for poor matches and caching for subexpressions, it achieves competitive performance (~800ms query time on a large corpus compared to ~300ms for MIaS).

Kumar et al. [6] use a Least Common Subsequence (LCS) algorithm for matching LaTeX strings. The input LaTeX strings are preprocessed as a canonicalization step where each function, variable, and number is mapped to an atomic term. Variables and numbers are generalized. A dynamic-programming LCS algorithm on these terms is then used to find the matches. This approach is fairly robust against small changes in the structure (i.e. it allows for inexact matching) as missing or modified parts of the expression will be gaps in the LCS. However the LCS algorithm is $O(n^2)$ in the expression size and requires a comparison with every expression in the index, which makes it unsuitable for large indexes.

Another approach to the problem uses substitution trees. Substitution trees come from the field of automatic theorem proving. The leaf nodes contain the expressions that have been inserted into the tree. The internal nodes contain expressions with at least one generic term. Each child represents a substitution of one or more of the generic terms of its parent with more specific terms (which can introduce new generic terms).

Kohlhase and Sucan's Math WebSearch [5] uses substitution trees to match expressions on their semantic meaning, rather than layout. This approach can match expressions that match the query term exactly up to α-equivalence, which means the structure must match exactly but individual terms may be substituted (to terms of the same type). In addition, subexpression matching was enabled by inserting all subexpressions of an expression when inserting it into the index. Input to Math WebSearch is Content MathML (see Section 2.5).

Schellenberg's work [12, 13] instead uses substitution trees to match expressions by their layout. Rather than performing an exact expression match (as Math WebSearch does), this system exhaustively searches the tree for matches. This has the benefit of allowing for partial matches that have structural similarity, but the drawback of greatly increased query times. The system introduces a ranking method for partial matches which will be discussed in the next section.

2.4 Ranking Math Query Results

While very effective metrics have been developed for ranking text queries, ranking math expressions remains an open question. There are some analogs between the two problems (for example, the concept of term frequency can be roughly applied in the same way to symbols in an expression), but a new strategy is needed.

Mathematical expressions can generally be considered to have a tree structure, unlike text which is linear. As such, the structure of the expression is important to capture in the ranking metric. Symbols in an expression can easily be broken into categories: operators, variables, constants, and functions. These categories should have different weights in the ranking function (e.g. operators have more meaning than variables, which can be renamed without much change to the expression) [16]. Relatedly, a useful quality for a ranking function to have would be to match expressions with the identical structure but variables renamed, with a penalty.

Another positive quality for a ranking function would be to favor results in which the matched symbols are more connected to each other than not. A common type of query would be one that matches a small part of a larger expression. For example, take the query $x+y$. The result $(x+y)*z$ would be a better match than $(x+z)*y$, because the query term is an exact subexpression. According to this quality, the more relevant result would thus be the one that contains symbols that are closest to an exact subexpression.

While most systems rely on their underlying text search systems for ranking partial matches, there have been some attempts made at ranking math expressions directly. Schellenberg [12, 13] defines a ranking function that combines two metrics: a bag-of-words comparison of the individual symbols in the expressions and a bipartite comparison that looks at pairs of symbols (including the structure between them: the relationship between the symbols and position along the baseline). For each of these metrics, the number of matching symbols / pairs is counted, and the average of these two scores is taken. Additionally, symbols that match the correct type but not the exact symbol are counted at 25% of a full match. While our index will be built using a different method than that of this system, our ranking function will build directly on its ideas.

2.5 MathML

MathML is an XML-based language for describing mathematical expressions. There are two forms: Content MathML describes semantic meaning and Presentation MathML describes layout. While some systems (Math WebSearch [5]) build indexes on semantics, we will be focusing on layout and thus Presentation MathML.

A MathML expression is contained in a root `<math>` tag. There are four token elements (elements that do not contain other elements): identifiers (`<mi>`), operators (`<mo>`), numbers (`<mn>`), and text (`<mtext>`).

The remaining tags are layout tags. The `<mrow>` tag describes a row of adjacent elements. Each of the child elements are placed in a row. The `<mfrac>` tag describes a fraction. The first child is the numerator and the second is the denominator. The `<msup>` tag describes a superscript, with the first child being the base and the second being the exponent. `<msub>` and `<msubsup>` tags describe subscripts and combination subscript-superscripts, respectively. Similarly, `<mover>` , `<munder>` , and `<munderover>` describe elements that are directly under and/or over an element (e.g. an integral sign). The `<msqrt>` tag describes the square root of its only child and the `<mroot>` tag describes the n-th root of its first child, where n is its second child. Tables are contained in `<mtable>` elements and contain rows (`<mtr>`) which contain cells (`<mtd>`). The `<mfenced>` tag describes a group where its children are surrounded by some fence symbol (e.g. parentheses).

As Presentation MathML can represent the same expression in multiple ways, a canonicalization procedure is needed to prevent a valid match from being ignored because it was represented differently. Archambault and Moço define Canonical MathML [1], an extension of Presentation MathML that does exactly this. Canonicalization is important because we are encoding the structure obtained from the MathML in our index. If an expression can take multiple forms in our index, a matching query and result could appear to not match if they are expressed differently in MathML, even when they are describing the same expression.

In Canonical MathML, all tags like `<msub>` , `<mfrac>` , and `<mroot>` must have exactly the right number of children. Most other rules involve enforcing the use of the `<mrow>` tag where it is not strictly necessary in standard MathML. Summation, product, and integrals must be contained in a two-element row. The first element is the symbol, which might also be wrapped in a `<msub>` or `<msubsup>` tag. The second element is the subexpression being (for example) summed, which must be wrapped in a second `<mrow>` if it contains multiple symbols. Similarly, a parenthesis group is contained in a `<mrow>`, with all fence symbols and commas as `<mo>` elements in it.

A common and more human-readable representation of mathematical expressions than MathML is LaTeX. LaTeX expressions are presentational and can be converted to Presentation MathML (see Section 3.1).

2.6 Summary

Existing methods of math information retrieval can be generally broken up into two groups: adaptations of existing text search tools or some form of tree-matching. The approach we describe in the next section takes a different tack: we apply the idea of an inverted index more directly on the structure of mathematical expressions.

Chapter 3

Methodology

Tangent uses an inverted index of pairs of symbols from expressions. In this chapter, we will describe the corpora and formats used for math representation. We will define the Symbol Layout Trees and the Symbol Pairs used in the index, and then the index itself. Then, we will define 5 different ranking functions for ordering matches. Finally, we will describe the implementation of our system.

3.1 Corpora

3.1.1 MREC

MREC (Math REtrieval Collection) is a collection of approximately 324,000 academic publications. The arXiv[1] contains preprint papers in science and mathematics. These documents have been converted to XHTML and MathML by the LaTeXML[2] project. MREC takes the documents that have been converted successfully and modifies them, most importantly by converting the MathML expressions to Canonical MathML.

[1]http://arxiv.org
[2]http://dlmf.nist.gov/LaTeXML

3.1.2 Wikipedia

The Wikipedia corpus contains every mathematical expression from the English-language Wikipedia project. It contains mathematical expressions from virtually every field and more importantly, definitions and explanations thereof. It is a desirable corpus because the information it contains is helpful in learning mathematics, which is one of the major use cases of a math search system. Additionally, we feel it is ideal for a comparison between search engines because the breadth rather than depth of the mathematics represented. Each concept might only have a few expressions, but with so many articles covering so much material, it is likely that a given query will have a wide variety of relevant results.

The corpus was assembled from a full XML archive of English Wikipedia created on May 4, 2013. The LaTeX expressions were extracted from the document along with the associated articles. The extracted LaTeX expressions were then converted into MathML documents using the same LaTeXML tool as used by MREC. There are 482,364 expressions contained in 32,780 articles.

3.2 Input Formats

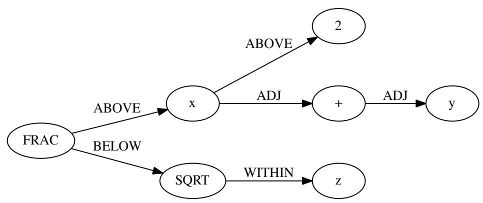
Our system can parse both LaTeX expressions and the XHTML + Canonical MathML files from the MREC and Wikipedia datasets. For simplicity, we use LaTeXML (the same tool used to create the datasets) to convert any LaTeX expressions to MathML, which we then parse into Symbol Layout Trees. While Wikipedia (MathML) will be used as the search corpus, LaTeX expressions are also necessary as MathML is too unwieldy for user input (for the query expression). In practice, the index will likely be built from MathML and search queries will be input with LaTeX (either written by the user or generated by our handwritten math recognition tool, m_{in} [11]). Every commonly-used MathML tag (all of those found in both corpora) is supported with the exception of those related to tables. The Symbol Pair representation used for the index is not well-defined for tables, so MathML expressions containing tables are ignored.

3.3 Symbol Layout Trees

Internally, Tangent uses a Symbol Layout Tree (SLT) to represent expressions. The vertices in this tree are the symbols in the expression and the edges are the spacial relationships between them. The tree is rooted at the leftmost symbol on the main baseline. Each symbol can have a relationship to those ABOVE, BELOW, ADJACENT to, and WITHIN it. Examples of ABOVE relationships include superscripts and upper limits on integrals (in some SLT representations these would be separate relationships, but we have combined them). Similarly, BELOW relationships include subscripts and lower limits. Fractions are encoded as a FRAC symbol with the numerator ABOVE and the denominator BELOW. A square root can have an expression WITHIN it, and most other symbols will be ADJACENT (always to the right). Figure 3.1 shows an example expression, its Symbol Layout Tree, and its Symbol Pairs (discussed in the next section).

$$\frac{x^2+y}{\sqrt{z}}$$

(a)



(b)

$(\mathrm{FRAC}, x, 1, 1)$ $(\mathrm{FRAC}, 2, 2, 2)$ $(\mathrm{FRAC}, +, 2, 1)$ $(\mathrm{FRAC}, y, 3, 1)$ $(\mathrm{FRAC}, \mathrm{SQRT}, 1, -1)$ $(\mathrm{FRAC}, z, 2, -1)$

$(x, 2, 1, 1)$ $(x, +, 1, 0)$ $(x, y, 2, 0)$ $(+, y, 1, 0)$ $(\mathrm{SQRT}, z, 1, 0)$

(c)

Figure 3.1: Example expression (a), Symbol Layout Tree (b), and Symbol Pairs(c)

3.4 Symbol Pairs

In text information retrieval, the documents are split into words to be inserted into an index. To (naively) apply this idea directly to mathematics, we could insert each symbol from an expression into the index, but this would lose all information about the structure of the expression.

Our approach is to instead add *pairs* of symbols to the index. The pairs are encoded with a representation of the structure between them. This Symbol Pair representation is a tuple $(s1, s2, d_h, d_v)$ where s_1 and s_2 are the two symbols in the pair. The horizontal distance, d_h, is the length of the path between s_1 and s_2 when traversing the SLT. It can also be thought of the number of symbols between s_1 and s_2 when moving rightward along the baseline (though not strictly true as, for example, the ABOVE relationship between a FRAC and a numerator counts towards d_h but does not move rightward). The vertical distance, d_v, is the height of s_2 above s_1's baseline, which increases for ABOVE relationships, decreases for BELOW relationships, and is not affected by ADJACENT or WITHIN relationships. For example, in the expression $x^y + z$, we would add $(x, y, 1, 1)$, $(x, +, 1, 0)$, $(x, z, 2, 0)$, and $(+, z, 1, 0)$ to the index. This representation is chosen to maintain the relative relationship between the pair of symbols but be general enough to be used for partial matches.

The Symbol Pairs are generated between every symbol (s_1) in the SLT and every symbol in the subtrees (children) of s_1. This means that s_1 is always a parent element of s_2 in the SLT. Not all pairs of symbols from the SLT will be included; specifically, two symbols that in separate child subtrees of a parent will not be included when generating the Symbol Pairs for an expression. There are a worst-case $\binom{n}{2}$ pairs inserted for an expression with n symbols, which would occur for any linear expression (i.e. when there are no branches in the SLT).

3.5 Symbol Pair Index

The index we have developed is similar to that of a text search engine, with Symbol Pairs instead of words. The index is a hash table mapping Symbol Pairs to the list of expressions that contain them. To create the index, we look up each symbol pair in the index (creating an empty list if none was found) and append the expression to the list that was found.

Querying the index is more complex. We construct a hash table, result_pairs, that maps each matching expression to a list of the Symbol Pairs it has in common with the query. To do this, we first look up each pair from the query expression in the index, getting a list of expressions that contain that pair. Then, for each expression in that list, we add the pair to the result_pairs entry for that expression. The full list of result expressions are then ordered by a ranking function, which is given the search expression, the result expressions, and the list of matching Symbol Pairs for each result expression.

It is possible for an expression to contain the same Symbol Pair more than once. When this happens, we count the pair n times, where n is the minimum of number of times the pair occurs in the query and the result. For example, if a Symbol Pair occurs 5 times in a result but only twice in the query, we will count it twice.

```
Index(expression, index):
    for pair in symbol pairs of expression:
        append expression to index[pair]

Search(query, index, k):
    for pair in symbol pairs of query:
        for expression in index[pair]:
            append pair to result_pairs[expression]
    sort expressions by the ranking function (using result_pairs)
    return the top k expressions
```

Figure 3.2: Pseudocode for indexing and searching.

In text search, the index often includes in the lists the position in the document where each word was found. Analogously, we include an identifier that defines the absolute position of s_2. This identifier is a d-element list, where d is the depth of the symbol in the tree. The ith element in this list is a 0, 1, 2, or 3 depending on whether the ith relationship in the path to s_2 is BELOW, ABOVE, ADJACENT, or WITHIN, respectively. Because d_h tells us the path length between s_1 and s_2, we can later calculate the identifier for s_1 by removing the last d_h elements of s_2's identifier. As only one of the ranking functions uses this information, we have not included it in the pseudocode for clarity.

3.6 Ranking Functions

It is necessary to apply a ranking function to rank the matched expressions by some metric of relevance. We have developed several such functions: F-Measure, Recall-Biased, Distance, Prefix, and Inverse Expression Frequency. F-Measure is the baseline, which balances between recall and precision of the match. The rest of the the ranking functions are each a modification to the F-Measure, usually by weighting each pair somehow. Recall-Biased uses a weighted F-Measure to prefer matches that contain most of the Symbol Pairs in the query. Distance gives higher weight to the Symbol Pairs that are closer. Inverse Expression Frequency gives higher weight to the Symbol Pairs that are less common in the index. Prefix removes Symbol Pairs that are from different parts of the query.

F-Measure Ranker. Our baseline metric is the F-Measure ranker. The recall-of-match is the percentage of Symbol Pairs in the query that are in the match, and the precision-of-match is likewise the percentage matched in the result. The F-Measure is the harmonic mean of the recall and precision of

F-Measure	$\dfrac{2\lvert M\rvert}{\lvert Q\rvert + \lvert R\rvert}$
Recall-biased	$\dfrac{(1+1.5^2)\lvert M\rvert}{1.5^2\lvert Q\rvert + \lvert R\rvert}$
Distance	$\dfrac{2\sum_{p\in M}\frac{1}{p.d_h}}{\sum_{p\in Q}\frac{1}{p.d_h} + \sum_{p\in R}\frac{1}{p.d_h}}$
Inverse Expression Frequency	$\dfrac{2\sum_{p\in M}\mathrm{ief}(p)}{\sum_{p\in Q}\mathrm{ief}(p) + \sum_{p\in R}\mathrm{ief}(p)}$
Prefix	$\dfrac{2\lvert\,\mathrm{prefix}(M)\rvert}{\lvert Q\rvert + \lvert R\rvert}$

Table 3.1: Definitions for the ranking functions. Q, R, and M are the sets of Symbol Pairs in the query, result candidate, and match (intersection) of the query and result, respectively. Note that each is an F-Measure that has been modified in some way.

the match: $\frac{2|M|}{|Q|+|R|}$, where Q, R, and M are the sets of pairs in the query, result candidate, and match (between the query and result) respectively. Ranking just by recall would bias results towards those that have the most pairs in common with the query, but it is not effective. It would not consider the length of the result and thus can rank highly results that are very long and thus have a low percentage (but high number) of matched pairs. In a similar way, ranking just by precision allows matches that are very short, which is also undesirable. Using the harmonic mean of these values allows us to strike an appropriate balance between these two extremes.

Recall-Biased Ranker. The Recall-Biased ranker is a slight modification to the F-Measure ranker which biases the results slightly towards those with high recall rather than precision. The reasoning behind this is that that a user might prefer partial matches that contain the query to those that are contained in the query. This would bias the system to including results that contain the query as a subexpression than vice versa. It is more likely that the user neglected to input part of the expression than he or she input too much. The formula for the Recall-Biased ranker is the $F_{1.5}$ score, where the general F_β score is $(1+\beta^2)\cdot\frac{\text{precision}\cdot\text{recall}}{(\beta^2\cdot\text{precision})+\text{recall}}$.

Distance Ranker. The Distance ranker gives higher weights to the pairs whose symbols are closer. Each pair is weighted by $1/d_h$, which has the effect of giving more importance to the pairs that are closer together. The ranker modifies the F-Measure thusly: instead of counting the set of pairs in each of M, Q, and R, we sum this $1/d_h$ weight for each Symbol Pair in the set. The formula for the ranker is then $\dfrac{2\displaystyle\sum_{p\in M}\frac{1}{p.d_h}}{\displaystyle\sum_{p\in Q}\frac{1}{p.d_h}+\sum_{p\in R}\frac{1}{p.d_h}}$, where $p.d_h$ is the d_h value for p.

Inverse Expression Frequency Ranker. The Inverse Expression Frequency (IEF) ranker adapts the common text search metric of TF-IDF (term frequency - inverse document frequency) to the math domain. The formula ends up being quite different (to that of text search), but the basic idea is the same: apply more weight to uncommon pairs (terms). We use the same f-measure metric, but instead of counting the number of pairs that match, we sum the inverse expression frequency (IEF) of each matched pair. The final score is $\dfrac{2\sum\limits_{p\in M}\text{ief}(p)}{\sum\limits_{p\in Q}\text{ief}(p)+\sum\limits_{p\in R}\text{ief}(p)}$.

Prefix Ranker. The Prefix ranker looks at a way to find an alignment between the query and result expressions and only include pairs in the match if they occur on that alignment. To provide this alignment we use the additional path information associated with each pair in the index (See Section 3.5). Specifically, for each pair in the match, we get the path to s_1 in both the query and the result. These paths are then used to find which pairs share a common prefix in the match using the algorithm described in Figure 3.4. The idea is to find the first place (from the symbol moving up the SLT) where the paths from the query and result diverge. We can then say that the Symbol Pair is rooted at this this prefix of the two paths. The Symbol Pairs for a large subexpression match will all be rooted at the same prefix, but Symbol Pairs that are in different places between the query and result will not - the idea is to prefer the more connected match.

Consider the example query and result in Figure 3.3. Between the two expressions, there are four matching Symbol Pairs: $(x, +, 1, 0)$, $(x, 2, 2, 0)$, $(+, 2, 1, 0)$, and $(y, 2, 1, 1)$. The path to x is $\emptyset$ in the query (it is the first symbol on the main baseline) and [A] in the result (above the FRAC symbol). The path to + is [N] in the query (one symbol along the baseline) and [AN] in the result. The path to y is [NNNN] in the query and [ANNN] in the result. For the first two Symbol Pairs, the prefix is $(\emptyset, [\text{A}])$, because the rightmost

$$x + 2 + y^2$$

(a)

$$\frac{x + 2y^2}{z}$$

(b)

Symbol Pair	Path in Query	Path in Result	Prefix
$(x, +, 1, 0)$	$\emptyset$	[A]	$(\emptyset, [\mathrm{A}])$
$(x, 2, 2, 0)$	$\emptyset$	[A]	$(\emptyset, [\mathrm{A}])$
$(+, 2, 1, 0)$	[N]	[AN]	$(\emptyset, [\mathrm{A}])$
$(y, 2, 1, 1)$	[NNNN]	[ANNN]	$([\mathrm{N}], [\mathrm{A}])$

(c)

*Figure 3.3: Example query (a), result (b), and matched Symbol Pairs and the paths to s_1 (c). "A" stands for ABOVE and "N" stands for ADJACENT (**n**ext to).*

element of the paths differs. For $(+, 2, 1, 0)$, we remove the common N from the end of the two paths, and get the same prefix of $(\emptyset, [\mathrm{A}])$. For $(y, 2, 1, 1)$, we can remove three N relationships, but this leaves us with a different prefix $([\mathrm{N}], [\mathrm{A}])$. As 3 matched pairs have the same prefix, this gives us a score of $\frac{2*3}{15+16} \approx 0.194$, whereas the F-Measure would have been $\frac{2*4}{15+16} \approx 0.258$.

The Prefix ranker handles the case of the same pair matching multiple times differently than the other rankers. We calculate the prefix for all combinations of the matches. For example, if a pair was in the query twice and a result candidate three times, we would calculate the prefix for all 6 combinations. Only two (in general, the minimum of the number in the query and

```
PrefixRank(pairs, query_size, result_size):
prefixes := dictionary()
for each pair p, with path l_q in the query and l_r in the result:
    n := max(len(l_q), len(l_r))
    pad l_q and l_r with null elements at beginning to length n
    while last(l_q) == last(l_r):
        popright(l_q)
        popright(l_r)
    prefixes[(l_q, l_r)] += 1
largest := max(prefixes)
return f-measure(largest, query_size, result_size)
```

Figure 3.4: Pseudocode for the Prefix ranking function.

result) of the prefixes can be the same, so this property is maintained from the other rankers while allowing the prefixes that are in the largest group to contribute to the result.

3.7 Implementation

The implementation of our system is designed around the Redis[3] key-value store. Redis is a robust, scalable, in-memory database that maps string keys to strings, lists, sets, sorted sets, and hash tables (all of strings). It was chosen to ease development (as the Tangent server can be restarted independently of Redis) and for its high performance. The index and all supporting information is stored in Redis. As Redis is a simple key-value store, each type of information is given a key; Table 3.2 lists the most important keys and the

[3]http://redis.io

values they store.

Key	Value
expr_count	total # of expressions; next expression id
pair:[sp]:exprs	list of expressions containing symbol pair [sp]
expr:[e]:docs	set of documents containing expression [e]
expr:[e]:mathml	MathML text for expression [e] (for display)
expr:[e]:[r]score	maximum score for expression [e] and ranker[r]

Table 3.2: Important Redis keys and their values.

The Tangent system itself is implemented in Python. There is a core library (approximately 1000 lines of code), which defines the data structures (SLT and Symbol Pairs), a MathML parser, and a RedisIndex class with functionality to add and search for expressions. Each Ranker is defined as a static class which contains a ranking function and various properties that tell the index what information to fetch from Redis. This allows the ranking function to be chosen on the fly, with all of them able to use the same index.

Building on this, there is a command-line tool for both indexing and searching. The primary interface for searching is however through a web interface built using the Flask[4] web framework. This simple web server receives a request containing the query expression and uses the RedisIndex from the core library to perform a search using the algorithm described in Section 3.5. The index then returns 10 result expressions and a link to the article for each,

[4]http://flask.pocoo.org

which are then displayed to the user. A screenshot of this interface can be seen in Figure 1.1b.

3.8 Summary

We have described our Symbol Layout Tree and Symbol Pair data structures and the index we have created using them. We also defined 5 different ranking functions for that index that each have a different goal for improving the quality of the results. In the next section, we will describe an experiment to compare our search results with those of an existing system.

Chapter 4

Results and Discussion

To test our hypothesis, we needed to run two experiments. The first was to compare the search results between our system and another and the second was to evaluate the performance characteristics of our system. In this section, we will first describe the design of our experiments. We will then display the results from the various parts of the experiments. Following that will be a discussion of the results.

4.1 Experimental Design

4.1.1 Query Result Relevance

It is difficult to evaluate relevance for query results automatically. Not only is the relevance of a result a subjective judgment of the user, relevance can vary widely between users and depending on the task of the user. For this reason, we conducted a human study to compare our search results with those of an existing, text-search-based system (Yuan and Zanibbi [19]). The experiment asked participants to score the top 10 unique results from each system for 10 queries. The participants were shown one query and result and asked "How similar is the result to the query?" They were instructed to answer on a 5-point Likert scale (See Figure 4.1). We asked the participants to score by similarity rather than relevance because relevance is dependent on the search task, which is beyond the scope of this experiment.

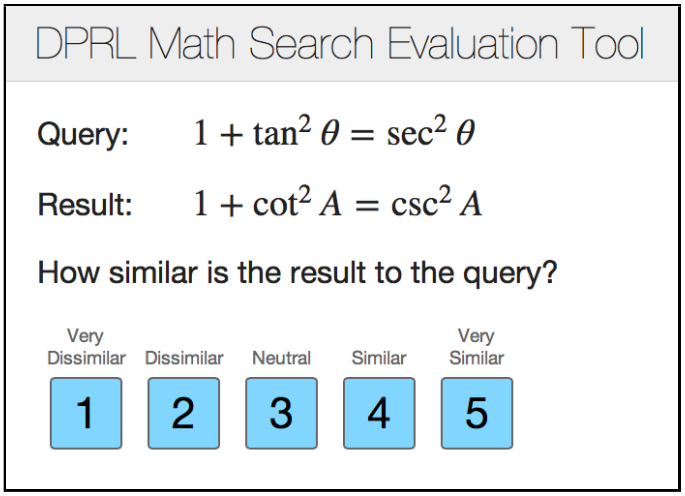


Figure 4.1: Screenshot of the evaluation tool used in the experiment.

The corpus used for the experiment was the Wikipedia corpus. The queries were chosen by randomly sampling a larger set of queries from the corpus, and picking from this 10 that represented diversity in size, type of structure, and field (see Table 4.1). Due to time constraints in the experiment, we were unable to compare all of the ranking functions we developed. We conducted an informal experiment beforehand and determined that the best-performing ranking functions were F-Measure, Distance, and Prefix. The experiment was thus a comparison between Tangent with these three ranking functions and Yuan and Zanibbi's Lucene-based system (henceforth referred to as Lucene).

#	Query
1	$\widetilde{\rho}$
2	$\bar{u} = (x, y, z)$
3	$1 + \tan^2\theta = \sec^2\theta$
4	$cos(\theta_E) = e^{-TR/T_1}$
5	$a = g\frac{m_1 - m_2}{m_1 + m_2}$
6	$\int_a^b f(x)\,dx = F(b) - F(a).$
7	$\Big(1/6,\ \sqrt{1/28},\ -\sqrt{12/7},\ 0,\ 0,\ 0,\ 0,\ 0\Big)$
8	$\sum_{i=m}^{n} a_i = a_m + a_{m+1} + a_{m+2} + \cdots + a_{n-1} + a_n.$
9	$f(x;\mu,c) = \sqrt{\frac{c}{2\pi}}\ \frac{e^{-\frac{c}{2(x-\mu)}}}{(x-\mu)^{3/2}}$
10	$D4\sigma = 4\sigma = 4\sqrt{\frac{\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} I(x,y)(x-\bar{x})^2\,dx\,dy}{\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} I(x,y)\,dx\,dy}}$

Table 4.1: List of queries used in the experiment.

The experiment was run in person, one participant at a time, through a web-based evaluation tool (see Figure 4.1). Before the experiment, participants were asked to fill out a short demographics survey. Then, they completed a short familiarization exercise, wherein they scored 5 results from each of 2 queries. These responses were excluded from analysis. After this task was complete and any questions were answered, they began the main experiment. In addition to the scores, we also recorded the time it took the participant for each query. Finally, they were asked to rate the difficulty of the task and to describe their process of scoring. The task took approximately 30 minutes and participants were paid $10 for their time.

To save time, any results found by both systems were only scored once. This allowed us to include results from three of our ranking functions without lengthening the experiment, as many of the results are shared between the different rankers. After deduplication, the participants were asked to score 214 results, 10 of which were for the familiarization exercise. To ensure a fair comparison, both the order of the queries and the order of the results were randomized for each participant.

4.1.2 System Performance

Indexing and retrieval speed is simpler to evaluate, as it can be directly measured. Indexing time was tested using the full Wikipedia corpus (approximately 482,000 expressions). Retrieval time was tested using the 10 queries from the experiment and the same corpus. We analyzed system clock time as opposed to operation counts because we are just interested in if the system is fast enough, and for that we need actual clock times. Operation counts would be more appropriate for a formal performance comparison. Using clock time has the drawback of introducing noise into our timings and being dependent on specific hardware. The former can be mitigated with multiple trials and the latter is reasonable considering our goal.

As indexing speed is only important for the initialization of the system, it is the less important of the two quantities being measured. It is still an interesting and important quantity. It would be difficult to experiment with the system if it takes days to index the corpus. Retrieval speed is much more important to the success of the system. It is mostly binary: either the system is fast enough to be used in real time (queries running in under approximately 3 seconds) or it is not.

4.2 Results

4.2.1 Demographics and Survey Responses

20 students and professors participated in the experiment. 15 (75%) of the participants were male. 15 (75%) were between the age of 18 and 24, three (15%) were 25-34, and one (5%) was 35-44. All participants were from fields in science and technology, with 13 (65%) in computing, 5 (25%)in science, and 2 (10%) in mathematics. Students and faculty in these fields were targeted because they represent a group that might find a math search engine useful in their work. 1 participant (5%) found the task very difficult, 11 (55%) found it difficult, 7 (35%) were neutral, and 1 (5%) found it easy. When asked to describe how they evaluated the results, 17 (85%) mentioned using visual similarity and 10 (50%) mentioned semantic meaning. The additional comments mostly described how difficult the users found the task, which aligned with their ratings.

4.2.2 Search Result Scores

Our result scores are from a Likert scale, which is ordinal data with no inherent numeric value. Whether it is valid to use Likert data as numeric values for statistical analysis remains an open question. We will avoid doing this, as we can test our hypothesis with the more conservative approach of treating it only as ordinal data. We can thus not directly apply an analysis of variance (ANOVA) test to our results. In order to get around this, we binarize the scores to values of either relevant (scored similar or very similar) or not

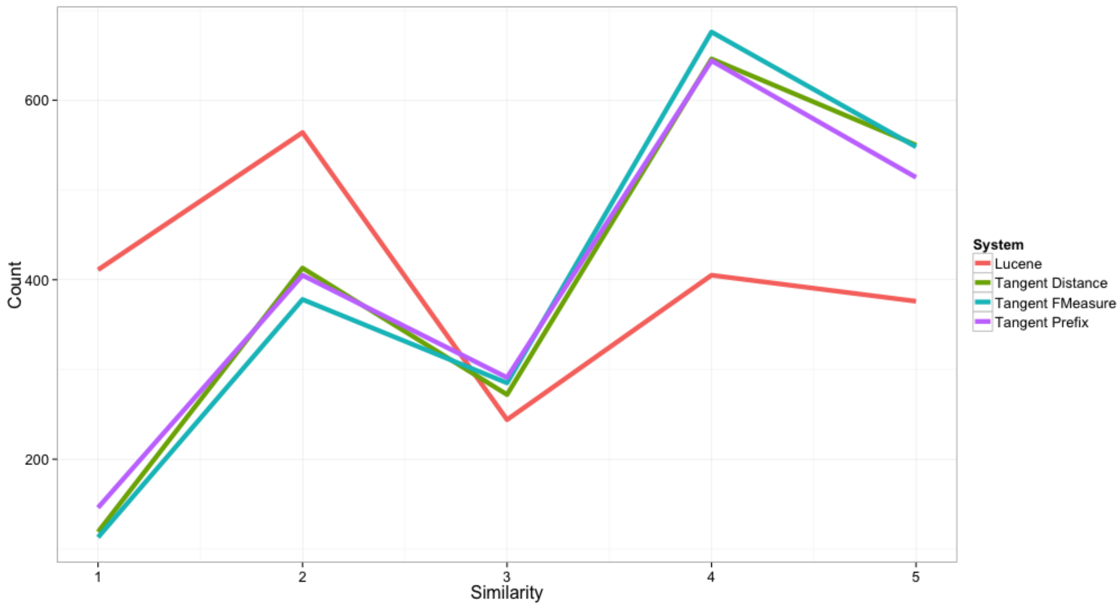


Figure 4.2: Score counts by system for 10 results each for 10 queries.

relevant (scored neutral or worse). With this, we can then calculate the top-10, top-5, and top-1 precision for each query. Table 4.2 shows these values for each system.

A two-way ANOVA test looking at the effects of the system and query on average top-10 precision shows a very strong effect of the system ($p < 2.2 * 10^{-16}$) as well as of the query ($p < 8.9 * 10^{-16}$). There was no interaction effect found between the system and query ($p < 0.205$). Running pairwise t-tests (using the Bonferroni correction for multiple comparisons), we see that there is a significant difference is between Lucene and each of the three Tangent variants (all with $p < 10^{-13}$). The t-tests did not show a significant difference between any of the different ranking functions for Tangent.

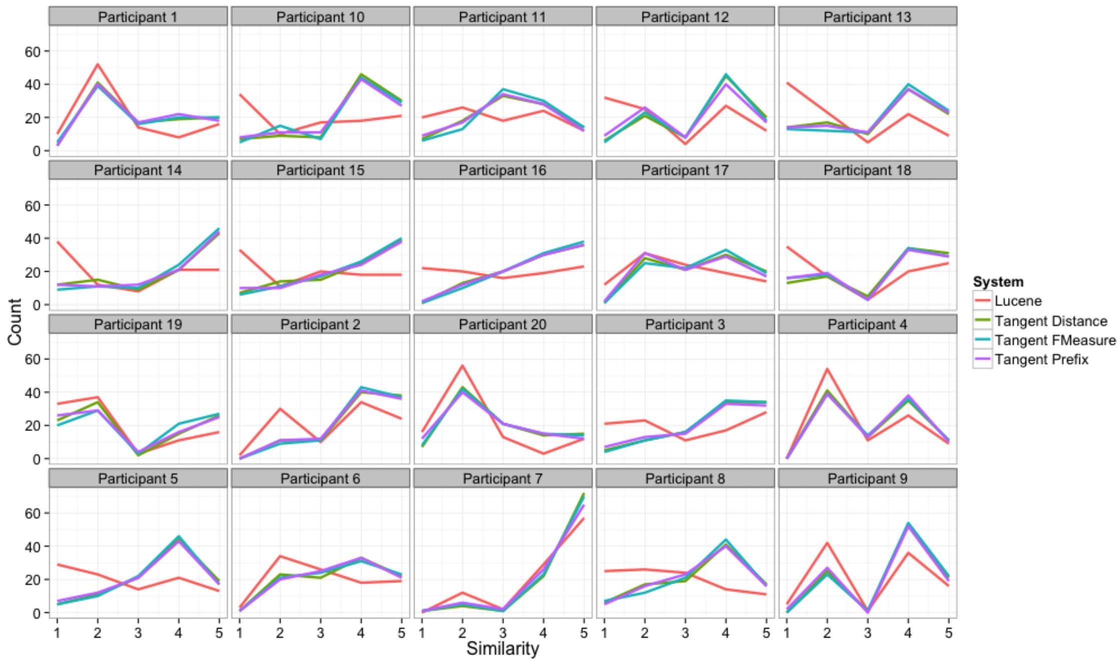


Figure 4.3: Score counts by participant and system.

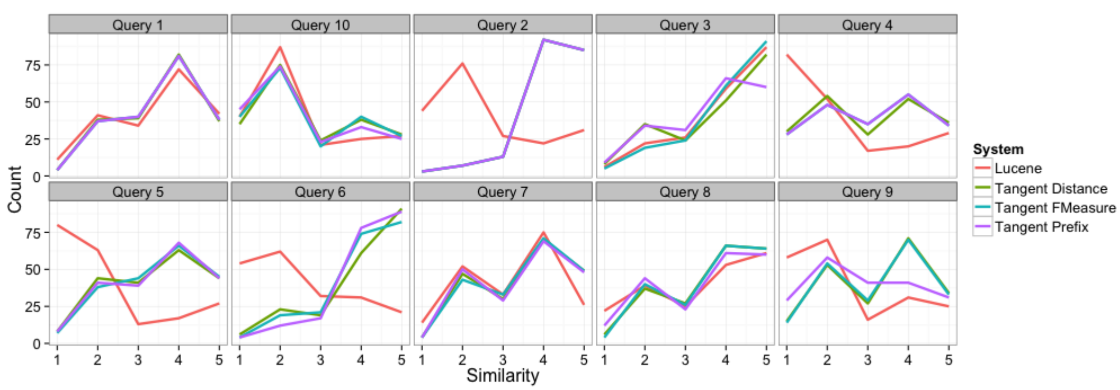


Figure 4.4: Score counts by query and system.

The ANOVA test is unnecessarily conservative for two reasons: firstly because it doesn't account for the variability between participants, and sec-

Query	Lucene			Tangent Distance			Tangent F-Measure			Tangent Prefix		
	top10	top5	top1	top10	top5	top1	top10	top5	top1	top10	top5	top1
1	0.57	0.51	0.40	0.60	0.71	1.00	0.60	0.71	1.00	0.60	0.71	1.00
2	0.27	0.42	0.90	0.89	0.96	1.00	0.89	0.96	1.00	0.89	0.96	1.00
3	0.73	0.90	1.00	0.67	0.85	1.00	0.76	0.85	1.00	0.63	0.74	1.00
4	0.25	0.42	0.95	0.44	0.62	0.95	0.45	0.68	0.95	0.45	0.68	0.95
5	0.22	0.30	0.05	0.54	0.62	1.00	0.56	0.63	1.00	0.56	0.63	1.00
6	0.26	0.36	0.15	0.76	0.97	1.00	0.78	0.97	1.00	0.84	0.90	1.00
7	0.51	0.41	0.75	0.60	0.78	1.00	0.60	0.72	1.00	0.59	0.78	1.00
8	0.57	0.63	1.00	0.65	0.84	1.00	0.65	0.81	1.00	0.61	0.81	1.00
9	0.28	0.34	0.10	0.53	0.64	1.00	0.52	0.58	1.00	0.36	0.57	1.00
10	0.26	0.41	0.70	0.33	0.46	1.00	0.34	0.49	1.00	0.29	0.45	1.00
μ	0.39	0.47	0.60	0.60	0.75	1.00	0.61	0.74	1.00	0.58	0.72	1.00
σ	0.18	0.18	0.39	0.16	0.16	0.02	0.16	0.16	0.02	0.19	0.15	0.02

Table 4.2: Top 10, Top 5, and Top 1 precision by query and system.

ondly because the binarization process loses information. Because the difference between Lucene and Tangent is so great, the test is perfectly valid and able to detect the difference. A Friedman test addresses both of these concerns. As a non-parametric test based on ranking the given scores, it is able to operate on the ordinal Likert scores directly. Additionally, the test allows blocks, where there is a source of variability between the blocks that is not important in the experiment. The blocks are ranked individually and thus the unwanted variability does not affect the result of the test. In our case, the blocking factors will be the query and the participant, the two sources of variability that we identified earlier. The p-value for the Friedman test is even stronger than the ANOVA result - it is a smaller value than our statistical package can represent.

4.2.3 Response Times

In addition to the scores directly given by the participants, we also collected the time it took each participant to score each result. As some results were shared between systems and only displayed to each user once, the timings were likewise counted for each system with that result. An ANOVA test on the timings by system showed that with high confidence ($p < 0.00007$) there is a significant difference between the systems. Specifically, from a t-test (Bonferroni correction) post-hoc, there is a difference between Lucene and each of the three Tangent systems, with p-values of 0.018, 0.0037, .000053 between Lucene and Distance, F-Measure, and Prefix respectively.

	mean	standard deviation
Lucene	5.84	5.81
Tangent Distance	5.36	5.52
Tangent FMeasure	5.29	4.68
Tangent Prefix	5.12	4.46

Table 4.3: Response times by system (in seconds).

There is also a strong correlation between the scores given to and time taken to score each result. As can be seen in Figure 4.5, participants took longest to score the expressions that were not obviously similar or dissimilar. This intuitively makes sense, as less thought is required if the expressions are identical or vastly dissimilar.

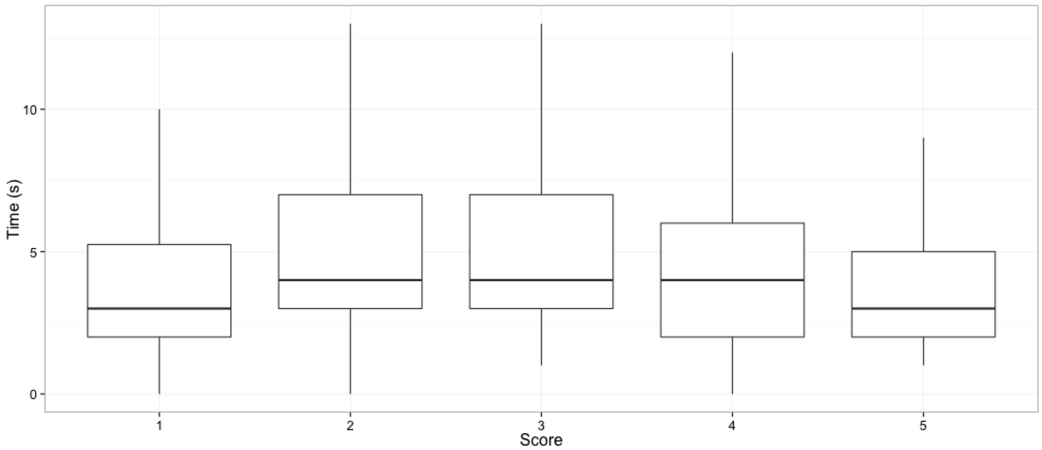


Figure 4.5: Response times by score.

4.3 Performance Metrics

As Lucene is a robust, mature, and heavily-optimized search engine, the Lucene-based system is much more performant than Tangent. It is faster for both indexing and searching, and the index is far smaller. But while Tangent is clearly slower than Lucene, it is not unusably slow. Indexing the Wikipedia dataset took 53 minutes - much slower than Lucene's 7 minutes 44 seconds, but perfectly workable for a one-time task. The average query time for the 10 queries used in the experiment is ~1.5 s with a standard deviation of ~1 s. This is well within our stated goal of being fast enough to be used in real time. Query time is dependent on the size of the query and number of matches, but the even the largest (slowest) queries we tested took under 3 seconds. Tangent's index for the Wikipedia dataset used 6.19 GB of memory while Lucene's on-disk index was a much smaller 107 MB (Tangent does not use compression while Lucene does).

4.4 Discussion

The results confirm both parts of our hypothesis. The ANOVA test (on computed precision) and Friedman test (on raw scores) both reject the null hypothesis and the associated post-hoc tests show that all three Tangent variations produce better precision-at-k values than the Lucene-based system. Our performance comparison shows that while slower, performance in all categories is sufficient for real-world use.

Both by looking at the graphs and the statistical tests, we can see that our result is clear: Tangent's results are better than Lucene's (Appendix A contains results for each query). While our scores are for the similarity between the query and result, there is intuitively a very strong correlation between similarity and relevance. If a result *looks* similar the query, we propose that it is more likely to *be* similar to the query.

The comparison between the various ranking functions was surprising and less conclusive. We only tested three (F-Measure, Distance, and Prefix), but IEF and Recall both performed worse than the others in a prior, informal comparison. None of the three we did test were shown to be significantly different. Considering this, we recommend the F-Measure ranker as it is the simplest and (one of) the best performing.

Several trends are apparent in the scores obtained from the experiment. By looking at the scores by participant (Figure 4.3) and by query (Figure 4.4), we can see that there is a lot of variability within both variables. Some participants scored expressions much higher on average than others, and several avoided the “neutral” score. The scores vary between queries even more: in 5 of the queries (queries 2, 4, 5, 6, and 9), Tangent performs much better than Lucene, but the scores in the other 5 queries are much closer. Additionally, both systems perform poorly in query 10.

But while there was a lot of variation between participant and between queries, Tangent’s scores were almost always higher. There are no individual participants or queries for which you could argue that Lucene performs better. Looking at the scores by query, we can put the queries in two categories: 5 where Tangent and Lucene score roughly as well as each other, and 5 where Tangent does much better. What we can draw from this is that while Lucene sometimes can find good matches, Tangent’s results are much more consistent. This is important for a search engine, as a system that only returns good results for half of the queries would be frustrating to use.

The performance analysis shows that while slower, Tangent is not untenable for real-world use. The high memory usage necessitates a modern computer that can address more than 4 GB of memory, but most current computers are capable of running Tangent well. Both indexing and query speed are good enough for the system to be used as an actual search engine. Additionally, there is a few decades’ research in index compression and optimization for text search that can largely applied to Tangent (see future work). It is likely that a much more optimized system could be comparable to Lucene in memory usage, indexing speed, and query speed. It’s possible to informally compare our query speed with Kamali and Tompa's results [4]. In an index that is roughly twice the size as ours, their system is roughly twice as fast as Tangent (~800ms vs ~1500ms per query). In the same comparison, Sojka and Líška's MIaS [14] is even faster, at ~300ms, and other various text-search and exact match systems take between 200 and 400ms. This speed difference could be a problem for larger indexes, but we can probably narrow the gap significantly with some optimization.

4.5 Summary

The experiments described in this section confirmed both parts of our hypothesis. The system provides high-quality search results and does so relatively efficiently. There is, however, much room for improvement in both areas that could be addressed in future work.

Chapter 5

Future Work and Conclusion

5.1 Future Work

5.1.1 Performance Optimizations

While the performance of Tangent is acceptable, there are many opportunities for improvement. The most obvious is to reimplement the system in a higher-performance language than Python. This is actually not as important as it might appear, because most of the query time is spent reading data from Redis (which calls C libraries). Another possibility would be storing the index directly in memory, rather than indirectly through Redis. We briefly experimented with both approaches (using the Go language), but very surprisingly, query times were similar between the current Python-Redis and the in-memory Go implementations. This was however not an exhaustive test, and implementation-specific speedups could be possible.

More interestingly, however, are the decades of research in optimizations for (text) information retrieval. Index compression in particular could be a great boon to performance because the majority of the query time is spent transferring the inverted lists from the Redis server. Storing the document references in a compressed form (either Elias' gamma code or a bytewise representation, described in Chapter 2) could a great reduction in the size of these lists, as would storing the differences between document identifiers rather than the identifiers themselves. Additionally, we could define a more compact representation for the Symbol Pairs, but most of the time is spent moving the document identifiers.

Another technique that could be applied from text information retrieval is parallelization. For an index the size of Wikipedia, a load-balanced set of servers running the entire index would allow the system to scale to more users, albeit without improving query time. For larger indexes, performance (query time) starts to become unacceptable. For these, more work would be needed to allow servers to only contain parts of the full index and communicate with each other to present a final result.

5.1.2 Retrieval Improvements

There are several ways worth exploring to improve the quality of the search results. A major limitation of Tangent is that it does not allow for substitution of symbols. For example, a^2 should be a partial match for b^3, as they are both a variable raised to an integer power. While we showed that the structure surrounding the symbols needing substitution is generally enough for Tangent to return good matches (see Query 2 in Appendix A), substitution would likely improve the result quality further. This could be accomplished by creating a separate substitution index where one or both of the symbols is replaced by a more general symbol (e.g. VARIABLE, OPERATOR, CONSTANT). However, the lists in such an index would likely be very long as more expressions would share the general symbols than the specific ones. This could make the performance of the system much worse without a clever approach to avoid this repetition.

Using our definition of d_h, Symbol Pairs must match this distance exactly. This is not as reasonable for large values of d_h (there is not much difference between a pair being 8 or 9 symbols apart. It could be possible to find more general matches if we binned the values for d_h. Another approach would be to search for pairs not only with exactly d_h but also $d_h + 1$, $d_h - 1$, and so on.

Our definition for d_v in the Symbol Pair is lossy. If, for example, an ABOVE relationship is followed by a BELOW relationship, d_v will be 0, the same as if there were two ADJACENT relationships. This would occur with the (x, z) pair in the expression x^{y_z}. A possible solution to this problem would be to define d_v as a list of the baseline changes - for this example, d_v would be [ABOVE, BELOW]. It is not clear if this would greatly affect the results, or even if the effect would be positive.

There are two possible problems with the way we generate the Symbol Pairs. The first is that we do not generate Symbol Pairs between pairs that are on separate branches of the SLT. For example, there is no pair between the 2 and y in $x^2 + y$. Fixing this would require major changes to the Symbol Pair representation, and as these pairs tend to represent weak semantic relationships, might not be helpful. The second issue is that the number of pairs increases quadratically with the size of a (linear) subexpression. While the unintentional effect of preferring well-connected matches seems to be positive, it is not apparent why this particular weighting is ideal. It would be possible to counter this effect (possibly to introduce another) by weighting each Symbol Pair inversely proportional to the number of other Symbol Pairs that use either of the symbols in it.

Currently, Tangent does not support tables or matrices because they cannot be represented well using a Symbol Layout Tree. One simple solution would be to simply index each table cell separately, as its own expression. Another approach would be to modify the SLT and Symbol Pair definitions to allow for this structure.

Finally, it might be interesting to use geometric distances for d_h and d_v instead of distances within the layout tree. We could render the expression to find these distances. Doing this would change the general type of matching that is being done, moving towards a directly visual match from a (somewhat) semantic one. It is unknown if this would produce better or worse results, but it would introduce to major problems. The first is that the distances would be dependent on the font and font size. The second is that these distances would then be real-valued and thus not able to be inserted into a table. Without a solution to these problems, indexing would likely be unfeasible.

5.1.3 Additional Comparisons

Our system performed well in the comparison with the Lucene-based system, but there are numerous systems that approach the problem differently.

It would be interesting to compare our search results with those of some of the other systems mentioned in Chapter 2; Kamali and Tompa's SimSearch [4] and Sojka and Líška's MIaS [14] would be good candidates.

5.2 Conclusion

Hypothesis: Implementing a math expression search system using an inverted index on symbol pairs from layout trees will: 1) yield more relevant results than text-based retrieval, and 2) be fast enough for real time use.

We have shown evidence to strongly support both claims in the hypothesis. Our novel approach to math expression retrieval yields high quality search results and does so efficiently. Not only have we created a search system that performs competitively today, we have introduced several new opportunities for future work (discussed below). Additionally, we introduced a rigorous method for evaluating results in math expression retrieval using human evaluation.

Appendices

Query 1: $\tilde{\rho}$

Rank	Lucene	Tangent F-Measure	Tangent Distance	Tangent Prefix
1	$\rho \to \widetilde{\tilde{\rho}}$	$\tilde{\rho}$	$\tilde{\rho}$	$\tilde{\rho}$
2	$\tilde{\rho}$	$\rho\tilde{\rho}$	$\rho\tilde{\rho}$	$\rho\tilde{\rho}$
3	$(\widetilde{\rho}, \widetilde{V})$	$\rho \to \widetilde{\tilde{\rho}}$	$\rho \to \widetilde{\tilde{\rho}}$	$\rho \to \widetilde{\tilde{\rho}}$
4	$\overline{\rho}\widetilde{\phi\psi}$	$\widetilde{\rho} = \rho^*$	$\widetilde{\rho} = \rho^*$	$\widetilde{\rho} = \rho^*$
5	$\widetilde{X}$	$\tilde{\rho}(\mathbf{k})$	$\tilde{\rho}(\mathbf{k})$	$\tilde{\rho}(\mathbf{k})$
6	$\widetilde{B}$	$\tilde{\rho}(r)$	$\tilde{\rho}(r)$	$\tilde{\rho}(r)$
7	$\widetilde{M}$	$\tilde{\rho}_{uc}(\mathbf{k})$	$\tilde{\rho}_{uc}(\mathbf{k})$	$\tilde{\rho}_{uc}(\mathbf{k})$
8	$\widetilde{t}$	$(\widetilde{\rho}, \widetilde{V})$	$(\widetilde{\rho}, \widetilde{V})$	$(\widetilde{\rho}, \widetilde{V})$
9	$(\rho, V) \mapsto (\widetilde{\rho}, \widetilde{V})$	$d\tilde{\rho}(\rho) = 0$	$d\tilde{\rho}(\rho) = 0$	$d\tilde{\rho}(\rho) = 0$
10	$\widetilde{\rho} = \rho^*$	$\mathrm{tr}_R\,\tilde{\chi} = \tilde{\rho}$	$\tilde{\rho}(1) \cdot v = v$	$\mathrm{tr}_R\,\tilde{\chi} = \tilde{\rho}$

Query 2: $\bar{u} = (x, y, z)$

Rank	Lucene	Tangent F-Measure	Tangent Distance	Tangent Prefix
1	$f(\bar{u}) = f(x, y, z)$	$\bar{u} = (x, y, z)$	$\bar{u} = (x, y, z)$	$\bar{u} = (x, y, z)$
2	$= \mathrm{R}(z, dt)\|x, y, z\rangle$	$u = (x, y, z)$	$u = (x, y, z)$	$u = (x, y, z)$
3	$(x \vee y)(\bar{x} \vee z)(y \vee z) = (x \vee y)(\bar{x} \vee z)$	$\mathbf{v} = (x, y, z)$	$\mathbf{v} = (x, y, z)$	$\mathbf{v} = (x, y, z)$
4	$\bar{u} = (x, y, z)$	$\mathbf{r} = (x, y, z)$	$\mathbf{r} = (x, y, z)$	$\mathbf{r} = (x, y, z)$
5	$z(x) = \frac{d}{dx} y(x)$	$\mathbf{x} = (x, y, z)$	$\mathbf{x} = (x, y, z)$	$\mathbf{x} = (x, y, z)$
6	$f(t, \bar{u}) = f(t, x, y, z)$	$F = (x, y, z)$	$F = (x, y, z)$	$F = (x, y, z)$
7	$\mathrm{P}(X = x\|Y = y, Z = z) = \mathrm{P}(X = x\|Z = z)$	$\mathbf{r_0} = (x, y, z)$	$\mathbf{r_0} = (x, y, z)$	$\mathbf{r_0} = (x, y, z)$
8	$= H(p, q) \cdot G(p, q)\|_{p=\frac{x}{\lambda z}, q=\frac{y}{\lambda z}}$	$\vec{x} = (x, y, z)$	$\vec{x} = (x, y, z)$	$\vec{x} = (x, y, z)$
9	$P = \{(x, y, z)\|3x + y - 2z = 10\}$	$\mathbf{x} = (x, y, z)^T$	(x, y, z)	$\mathbf{x} = (x, y, z)^T$
10	$z(x) = Q(y(x), \frac{d}{dx} y(x))$	(x, y, z)	$\mathbf{x} = (x, y, z)^T$	(x, y, z)

Query 3: $1 + \tan^2\theta = \sec^2\theta$

Rank	Lucene	Tangent F-Measure	Tangent Distance	Tangent Prefix
1	$1 + \tan^2\theta = \sec^2\theta$	$1 + \tan^2\theta = \sec^2\theta$	$1 + \tan^2\theta = \sec^2\theta$	$1 + \tan^2\theta = \sec^2\theta$
2	$\tan^2\theta + 1 = \sec^2\theta$	$1 + \tan^2 y = \sec^2 y$	$1 + \tan^2 y = \sec^2 y$	$1 + \tan^2 y = \sec^2 y$
3	$\sec^2\theta = 1 + \tan^2\theta$	$\frac{d}{d\theta}\tan\theta = \sec^2\theta$	$\frac{d}{d\theta}\tan\theta = \sec^2\theta$	$\frac{d}{d\theta}\tan\theta = \sec^2\theta$
4	$1 + \tan^2\theta = \sec^2\theta \quad \text{and} \quad 1 + \cot^2\theta = \csc^2\theta.$	$1 + \cot^2\theta = \csc^2\theta$	$\sec^2\theta = 1 + \tan^2\theta$	$1 + \cot^2\theta = \csc^2\theta$
5	$\cos^2\theta + \sin^2\theta = 1\ ,$	$\sec^2\theta = 1 + \tan^2\theta$	$1 + \cot^2\theta = \csc^2\theta$	$\sqrt{1 + \tan^2\theta_o}$
6	$\sin^2\theta + \cos^2\theta = 1$	$\sqrt{1 + \tan^2\theta_o}$	$\tan^2\theta + 1 = \sec^2\theta$	$\pm\sqrt{1 + \tan^2\theta}$
7	$\cos^2\theta + \sin^2\theta = 1$	$\pm\sqrt{1 + \tan^2\theta}$	$\sqrt{1 + \tan^2\theta_o}$	$1 + \cot^2 A = \csc^2 A$
8	$1 + \cot^2\theta = \csc^2\theta$	$1 + \cot^2 A = \csc^2 A$	$\pm\sqrt{1 + \tan^2\theta}$	$1 + \cot^2 y = \csc^2 y$
9	$\cot^2\theta +\ 1 = \csc^2\theta$	$1 + \cot^2 y = \csc^2 y$	$d\tan\theta = \sec^2\theta\, d\theta = \frac{dx}{3}.$	$\pm\frac{1}{\sqrt{1+\tan^2\theta}}$
10	$x = r\cos\theta = 2a\sin^2\theta = \frac{2a\tan^2\theta}{\sec^2\theta} = \frac{2at^2}{1+t^2}$	$\tan^2\theta + 1 = \sec^2\theta$	$\pm\frac{\tan\theta}{\sqrt{1+\tan^2\theta}}$	$d\tan\theta = \sec^2\theta\, d\theta = \frac{dx}{3}.$

Query 4: $cos(\theta_E) = e^{-TR/T_1}$

Rank	Lucene
1	$cos(\theta_E) = e^{-TR/T_1}$
2	$cos(\theta_E) = e^{-(d1+at)/T_1}$
3	$AF_{HAST} = AF_H * AF_T = e^{(Constant*(RH_s^n - RH_o^n)} * e^{(E_a/k)*(1/T_o - 1/T_s)}$
4	$AF_T = e^{(E_a/k)*(1/T_o - 1/T_s)}$
5	$P(E_i) = g(E_i)/(e^{(E-\mu)/kT} + 1)$
6	$\sin\theta_c = [-K_2 \pm (K_2^2 - 3K1K_3)^{1/2}]/3K_3{}^{1/2}$
7	$G_{T_2}^{\ominus} = G_{T_1}^{\ominus} + (C_p^{\ominus} - S_{T_1}^{\ominus})(T_2 - T_1) - T_2 \ln{(T_2/T_1)} C_p^{\ominus}$
8	$1 = \cos(3\pi/8) A_{T,L}^{x_c} + B_{T,L}^{x_c} + \cos(\pi/4) E_{T,L}^{x_c}$
9	$I_{\mathrm{C}} = I_{\mathrm{SO}}(e^{V_{\mathrm{BE}}/V_{\mathrm{T}}} - 1) \approx I_{\mathrm{SO}} e^{V_{\mathrm{BE}}/V_{\mathrm{T}}}$
10	$\mathbb{E}_\theta[(\theta_i - X_i)h(\mathbf{X}) \vert X_j = x_j (j \neq i)] = \int (\theta_i - x_i) h(\mathbf{x}) \left(\frac{1}{2\pi}\right)^{n/2} e^{-(1/2)(\mathbf{x}-\theta)^T(\mathbf{x}-\theta)} m(dx_i)$

Rank	Tangent F-Measure	Tangent Distance	Tangent Prefix
1	$cos(\theta_E) = e^{-TR/T_1}$	$cos(\theta_E) = e^{-TR/T_1}$	$cos(\theta_E) = e^{-TR/T_1}$
2	$cos(\theta) = 1$	$cos(\theta_E) = e^{-(d1+at)/T_1}$	$cos(\theta) = 1$
3	$cos(\theta_E) = e^{-(d1+at)/T_1}$	$cos(\theta) = 1$	$cos(\theta_E) = e^{-(d1+at)/T_1}$
4	$cos(\alpha) = 1$	$r \cdot cos(\theta)$	$cos(\alpha) = 1$
5	$r \cdot cos(\theta)$	$\mu = \vert cos(\theta) \vert$	$r \cdot cos(\theta)$
6	$x_1 = a_1 cos(\theta)$	$g =< cos(\theta) >$	$x_1 = a_1 cos(\theta)$
7	$x_2 = a_2 cos(\theta)$	$x_1 = a_1 cos(\theta)$	$x_2 = a_2 cos(\theta)$
8	$\sum_{r=1}^{3} cos(\frac{2r\pi}{7}) = \frac{-1}{2}$	$x_2 = a_2 cos(\theta)$	$\sum_{r=1}^{3} cos(\frac{2r\pi}{7}) = \frac{-1}{2}$
9	$cos(\alpha) = \frac{1}{k}$	$cos(\alpha) = 1$	$cos(\alpha) = \frac{1}{k}$
10	$T_L = \frac{T_N}{cos(\theta)}$	$T_L = \frac{T_N}{cos(\theta)}$	$T_L = \frac{T_N}{cos(\theta)}$

Query 5: $a = g\frac{m_1-m_2}{m_1+m_2}$

Rank	Lucene	Tangent F-Measure	Tangent Distance	Tangent Prefix
1	$\mathbf{A}_{\text{pent.}} = \frac{1}{2}\lvert x_1y_2 + x_2y_3 + x_3y_4 + x_4y_5 + x_5y_1 - x_2y_1 - x_3y_2 - x_4y_3 - x_5y_4 - x_1y_5\rvert$	$a = g\frac{m_1-m_2}{m_1+m_2}$	$a = g\frac{m_1-m_2}{m_1+m_2}$	$a = g\frac{m_1-m_2}{m_1+m_2}$
2	$V_V = V_T - V_a - V_b\frac{P_2}{P_2-P_1}$	$v = c\frac{m_1^2-m_2^2}{m_1^2+m_2^2}$	$\frac{m_1m_2}{m_1+m_2}$	$v = c\frac{m_1^2-m_2^2}{m_1^2+m_2^2}$
3	$V_G = \frac{R_x}{R_3+R_x}V_s - \frac{R_2}{R_1+R_2}V_s$	$\frac{m_1m_2}{m_1+m_2}$	$v = c\frac{m_1^2-m_2^2}{m_1^2+m_2^2}$	$\frac{m_1m_2}{m_1+m_2}$
4	$a = g\frac{m_1-m_2}{m_1+m_2}$	$\mu = \frac{m_1m_2}{m_1+m_2}$	$\mu = \frac{m_1m_2}{m_1+m_2}$	$\frac{m_1(u_2-u_1)}{m_1+m_2}$
5	$\mathbf{A}_{\text{tri.}} = \frac{1}{2}\lvert x_1y_2 + x_2y_3 + x_3y_1 - x_2y_1 - x_3y_2 - x_1y_3\rvert$	$\frac{m_1(u_2-u_1)}{m_1+m_2}$	$I = \frac{m_1m_2}{m_1+m_2}d^2$	$\mu = \frac{m_1m_2}{m_1+m_2}$
6	$D_2 = 1P_1 + 5P_2 - 3D_{\infty_1} - 3D_{\infty_2}$	$I = \frac{m_1m_2}{m_1+m_2}d^2$	$\frac{m_1u_1+m_2u_2}{m_1+m_2}$	$v = \frac{m_1}{m_1+m_2}u_1\,.$
7	$U_1 = R_1 - \frac{n_1(n_1+1)}{2}$	$\frac{m_1u_1+m_2u_2}{m_1+m_2}$	$\frac{m_1(u_2-u_1)}{m_1+m_2}$	$\frac{m_1u_1+m_2u_2}{m_1+m_2}$
8	$D_1 = 6P_1 + 4P_2 - 5D_{\infty_1} - 5D_{\infty_2}$	$\frac{m_2(u_2-u_1)}{m_1+m_2}$	$\frac{m_1(u_1-u_2)}{m_1+m_2}$	$\frac{m_2(u_2-u_1)}{m_1+m_2}$
9	$\frac{C_1}{C_2} = \frac{R_4}{R_3} - \frac{R_2}{R_1}$	$\frac{m_1(u_1-u_2)}{m_1+m_2}$	$v = \frac{m_1}{m_1+m_2}u_1\,.$	$\frac{m_1(u_1-u_2)}{m_1+m_2}$
10	$y = \frac{y_1-y_2}{x_1-x_2}x + \frac{x_1y_2-x_2y_1}{x_1-x_2}$	$T = \frac{2gm_1m_2}{m_1+m_2}$	$r = \frac{m_2d}{m_1+m_2}.$	$I = \frac{m_1m_2}{m_1+m_2}d^2$

Query 6: $\int_a^b f(x)\,dx = F(b) - F(a)$.

Rank	Lucene
1	$Y(t) = e^{(t-t_0)A}\ Y_0 + \int_{t_0}^{t} e^{(t-x)A}\ F(x)\ dx.$
2	$F(t) = \int_{x_0(t)}^{x_1(t)} f(x;t)\,dx.$
3	$\int_{-\infty}^{\infty} v_n^2(x)dF(x) = \int_0^1 u_n^2(t)\,dt.$
4	$\int_a^b f(x)\,dx = F(b) - F(a).$
5	$F(k) = \int_{-\infty}^{+\infty} f(x)e^{ik\cdot x}dx$
6	$\int_a^b f^{-1}(x)\,dx = g(b) - g(a)\ ,\ g(t) = tf^{-1}(t) - F(f^{-1}(t))$
7	$F(k) = \|F(k)\|e^{i\phi(k)} = \int_{-\infty}^{\infty} f(x)\ e^{-2\pi i u\cdot x}\ dx$
8	$\int_{-1}^{1} C_n^{(\alpha)}(x)C_m^{(\alpha)}(x)(1-x^2)^{\alpha-\frac{1}{2}}\ dx = 0.$
9	$\Omega(r) = \frac{\int_r^{\infty}(1-F(x))\,dx}{\int_{-\infty}^{r}F(x)dx}$
10	$(F,G)'_\sigma = \displaystyle\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} F(x)\overline{G(y)}\|x-y\|^{2\sigma-1}\,dx\,dy.$

Rank	Tangent F-Measure	Tangent Distance	Tangent Prefix
1	$\int_a^b f(x)\,\mathrm{d}x = F(b) - F(a).$	$\int_a^b f(x)\,\mathrm{d}x = F(b) - F(a).$	$\int_a^b f(x)\,\mathrm{d}x = F(b) - F(a).$
2	$\int_a^b f(t)\,dt = F(b) - F(a).$	$\int_a^b f(t)\,dt = F(b) - F(a).$	$\int_a^b f(t)\,dt = F(b) - F(a).$
3	$\int_a^b f'(x)\,dx = f(b) - f(a).$	$\int_a^b f(x)\,dx = G(b) = F(b) - F(a).$	$\int_a^b f'(x)\,dx = f(b) - f(a).$
4	$\int_\gamma f(z)\,dz = F(b) - F(a).$	$\int_a^b f'(x)\,dx = f(b) - f(a).$	$\int_\gamma f(z)\,dz = F(b) - F(a).$
5	$\int_a^b f(x)\,dx = G(b) = F(b) - F(a).$	$\int_\gamma f(z)\,dz = F(b) - F(a).$	$\int_\gamma g(\zeta)d\zeta = f(b) - f(a).$
6	$\int_\gamma g(\zeta)d\zeta = f(b) - f(a).$	$F(b) - F(a) = \int_a^b f(x)\,dx,$	$\int f(x)dx = F(x)$
7	$\int f(x)dx = F(x)$	$\int f(x)dx = F(x)$	$\int_a^b f(x)\,dx = f(c)(b-a)$
8	$\int_a^b f(x)\,dx = f(c)(b-a)$	$\frac{\int_a^b xg(x)dx}{F(b)-F(a)}$	$\int_a^b f(x)\,dx = G(b) = F(b) - F(a).$
9	$\int_0^\pi f(x)\sin(x)\,dx = F(\pi) + F(0).$	$\int f(x)\,dx = F(x) + C.$	$\int_0^t f(u)\,du = f(t) - f(0).$
10	$\int f(x)\,dx = F(x) + C.$	$\frac{g(x)}{F(b)-F(a)}$	$\int f(x)\,dx = F(x) + C.$

Query 7: $\left(1/6,\ \sqrt{1/28},\ -\sqrt{12/7},\ 0,\ 0,\ 0,\ 0,\ 0\right)$

Rank	Tangent Distance	Tangent Prefix
1	$\left(\sqrt{1/45},\ 1/6,\ \sqrt{1/28},\ -\sqrt{12/7},\ 0,\ 0,\ 0,\ 0,\ 0\right)$	$\left(1/6,\ \sqrt{1/28},\ -\sqrt{12/7},\ 0,\ 0,\ 0,\ 0,\ 0\right)$
2	$\left(\sqrt{1/45},\ 1/6,\ \sqrt{1/28},\ \sqrt{1/21},\ \sqrt{1/15},\ \sqrt{1/10},\ -\sqrt{3/2},\ 0,\ 0\right)$	$\left(\sqrt{1/45},\ 1/6,\ \sqrt{1/28},\ -\sqrt{12/7},\ 0,\ 0,\ 0,\ 0,\ 0\right)$
3	$\left(\sqrt{1/55},\ \sqrt{1/45},\ 1/6,\ \sqrt{1/28},\ \sqrt{1/21},\ \sqrt{1/15},\ -2\sqrt{2/5},\ 0,\ 0,\ 0\right)$	$\left(1/6,\ \sqrt{1/28},\ \sqrt{1/21},\ -\sqrt{5/3},\ 0,\ 0,\ 0,\ 0\right)$
4	$\left(1/6,\ \sqrt{1/28},\ \sqrt{1/21},\ \sqrt{1/15},\ \sqrt{1/10},\ \sqrt{1/6},\ -2\sqrt{1/3},\ 0\right)$	$\left(\sqrt{1/28},\ -\sqrt{12/7},\ 0,\ 0,\ 0,\ 0,\ 0\right)$
5	$\left(\sqrt{1/55},\ \sqrt{1/45},\ 1/6,\ \sqrt{1/28},\ \sqrt{1/21},\ \sqrt{1/15},\ \sqrt{1/10},\ \sqrt{1/6},\ -2\sqrt{1/3},\ 0\right)$	$\left(1/6,\ -\sqrt{7/4},\ 0,\ 0,\ 0,\ 0,\ 0,\ 0\right)$
6	$\left(\sqrt{1/28},\ -\sqrt{12/7},\ 0,\ 0,\ 0,\ 0,\ 0\right)$	$\left(\sqrt{1/55},\ \sqrt{1/45},\ 1/6,\ \sqrt{1/28},\ -\sqrt{12/7},\ 0,\ 0,\ 0,\ 0,\ 0\right)$
7	$\left(1/6,\ \sqrt{1/28},\ \sqrt{1/21},\ -\sqrt{5/3},\ 0,\ 0,\ 0,\ 0\right)$	$\left(\sqrt{1/45},\ 1/6,\ -\sqrt{7/4},\ 0,\ 0,\ 0,\ 0,\ 0,\ 0\right)$
8	$\left(\sqrt{1/55},\ \sqrt{1/45},\ 1/6,\ -\sqrt{7/4},\ 0,\ 0,\ 0,\ 0,\ 0,\ 0\right)$	$\left(\sqrt{1/28},\ \sqrt{1/21},\ -\sqrt{5/3},\ 0,\ 0,\ 0,\ 0\right)$
9	$\left(\sqrt{1/28},\ \sqrt{1/21},\ \sqrt{1/15},\ \sqrt{1/10},\ \sqrt{1/6},\ -2\sqrt{1/3},\ 0\right)$	$\left(-\sqrt{12/7},\ 0,\ 0,\ 0,\ 0,\ 0\right)$
10	$\left(\sqrt{1/55},\ \sqrt{1/45},\ 1/6,\ \sqrt{1/28},\ -\sqrt{12/7},\ 0,\ 0,\ 0,\ 0,\ 0\right)$	$\left(\sqrt{1/55},\ \sqrt{1/45},\ 1/6,\ -\sqrt{7/4},\ 0,\ 0,\ 0,\ 0,\ 0,\ 0\right)$

Rank	Tangent Distance	Tangent Prefix
1	$\left(1/6,\ \sqrt{1/28},\ -\sqrt{12/7},\ 0,\ 0,\ 0,\ 0,\ 0\right)$	$\left(1/6,\ \sqrt{1/28},\ -\sqrt{12/7},\ 0,\ 0,\ 0,\ 0,\ 0\right)$
2	$\left(\sqrt{1/45},\ 1/6,\ \sqrt{1/28},\ -\sqrt{12/7},\ 0,\ 0,\ 0,\ 0,\ 0\right)$	$\left(\sqrt{1/45},\ 1/6,\ \sqrt{1/28},\ -\sqrt{12/7},\ 0,\ 0,\ 0,\ 0,\ 0\right)$
3	$\left(\sqrt{1/28},\ -\sqrt{12/7},\ 0,\ 0,\ 0,\ 0,\ 0\right)$	$\left(\sqrt{1/28},\ -\sqrt{12/7},\ 0,\ 0,\ 0,\ 0,\ 0\right)$
4	$\left(\sqrt{1/55},\ \sqrt{1/45},\ 1/6,\ \sqrt{1/28},\ -\sqrt{12/7},\ 0,\ 0,\ 0,\ 0,\ 0\right)$	$\left(\sqrt{1/55},\ \sqrt{1/45},\ 1/6,\ \sqrt{1/28},\ -\sqrt{12/7},\ 0,\ 0,\ 0,\ 0,\ 0\right)$
5	$\left(1/6,\ \sqrt{1/28},\ \sqrt{1/21},\ -\sqrt{5/3},\ 0,\ 0,\ 0,\ 0\right)$	$\left(1/6,\ \sqrt{1/28},\ \sqrt{1/21},\ -\sqrt{5/3},\ 0,\ 0,\ 0,\ 0\right)$
6	$\left(1/6,\ -\sqrt{7/4},\ 0,\ 0,\ 0,\ 0,\ 0,\ 0\right)$	$\left(1/6,\ -\sqrt{7/4},\ 0,\ 0,\ 0,\ 0,\ 0,\ 0\right)$
7	$\left(-\sqrt{12/7},\ 0,\ 0,\ 0,\ 0,\ 0\right)$	$\left(-\sqrt{12/7},\ 0,\ 0,\ 0,\ 0,\ 0\right)$
8	$\left(\sqrt{1/28},\ \sqrt{1/21},\ -\sqrt{5/3},\ 0,\ 0,\ 0,\ 0\right)$	$\left(\sqrt{1/28},\ \sqrt{1/21},\ -\sqrt{5/3},\ 0,\ 0,\ 0,\ 0\right)$
9	$\left(\sqrt{1/45},\ 1/6,\ -\sqrt{7/4},\ 0,\ 0,\ 0,\ 0,\ 0,\ 0\right)$	$\left(\sqrt{1/21},\ -\sqrt{5/3},\ 0,\ 0,\ 0,\ 0\right)$
10	$\left(\sqrt{1/45},\ 1/6,\ \sqrt{1/28},\ \sqrt{1/21},\ -\sqrt{5/3},\ 0,\ 0,\ 0,\ 0\right)$	$\left(\sqrt{1/45},\ 1/6,\ -\sqrt{7/4},\ 0,\ 0,\ 0,\ 0,\ 0,\ 0\right)$

Query 8: $\sum_{i=m}^{n} a_i = a_m + a_{m+1} + a_{m+2} + \cdots + a_{n-1} + a_n.$

Rank	Tangent Distance
1	$\sum_{i=m}^{n} a_i = a_m + a_{m+1} + a_{m+2} + \cdots + a_{n-1} + a_n.$
2	$\sum_{i=m}^{n} x_i = x_m + x_{m+1} + x_{m+2} + \cdots + x_{n-1} + x_n.$
3	$\sum_{u=u_1}^{u_2} \sum_{v=v_1}^{v_2} b_{uv} = p_{u_2 v_2} + p_{u_1 v_1} - p_{u_1 v_2} - p_{u_2 v_1}$
4	$\mathbf{a} \cdot \mathbf{b} = \sum_{i=1}^{n} a_i b_i = a_1 b_1 + a_2 b_2 + \cdots + a_n b_n$
5	$\mathbf{x} \cdot \mathbf{y} = \sum_{i=1}^{n} x_i y_i = x_1 y_1 + x_2 y_2 + \cdots + x_n y_n$
6	$\mathbf{x} \cdot \mathbf{y} = \sum_{i=1}^{n} x_i y_i = x_1 y_1 + x_2 y_2 + \cdots + x_n y_n,$
7	$a_1 + a_2 + a_3 + \cdots + a_n = \sum_{i=1}^{n} a_i$
8	$P_1 D_{1j} + P_2 D_{2j} + \cdots + P_n D_{nj} = \sum_{i=1}^{n} P_i D_{ij}$
9	$\lvert n^{(4)}\rangle = \Bigg[\frac{V_{k_1 k_2} V_{k_2 k_3} V_{k_3 k_4} V_{k_4 k_2} + V_{k_3 k_2} V_{k_1 k_2} V_{k_4 k_3} V_{k_2 k_4}}{2 E_{k_1 n} E_{k_2 k_3}^2 E_{k_2 k_4}} - \frac{V_{k_2 k_3} V_{k_3 k_4} V_{k_4 n} V_{k_1 k_2}}{E_{k_1 n} E_{k_2 n} E_{n k_3} E_{n k_4}} + \frac{V_{k_1 k_2}}{E_{k_1 n}} \left(\frac{\lvert V_{k_2 k_3}\rvert^2 V_{k_2 k_2}}{E_{k_2 k_3}^3} - \frac{\lvert V_{n k_3}\rvert^2 V_{k_2 n}}{E_{k_3 n}^2 E_{k_2 n}} \right)$
10	$\sum_{n=0}^{\infty} a_n = a_0 + a_1 + a_2 + \cdots .$

Rank	Tangent F-Measure	Tangent Distance	Tangent Prefix
1	$\sum_{i=m}^{n} a_i = a_m + a_{m+1} + a_{m+2} + \cdots + a_{n-1} + a_n.$	$\sum_{i=m}^{n} a_i = a_m + a_{m+1} + a_{m+2} + \cdots + a_{n-1} + a_n.$	$\sum_{i=m}^{n} a_i = a_m + a_{m+1} + a_{m+2} + \cdots + a_{n-1} + a_n.$
2	$\sum_{i=m}^{n} x_i = x_m + x_{m+1} + x_{m+2} + \cdots + x_{n-1} + x_n.$	$\sum_{n=0}^{\infty} a_n = a_0 + a_1 + a_2 + \cdots .$	$\sum_{i=m}^{n} x_i = x_m + x_{m+1} + x_{m+2} + \cdots + x_{n-1} + x_n.$
3	$\sum_{n=0}^{\infty} a_n = a_0 + a_1 + a_2 + \cdots .$	$\sum_{i=m}^{n} x_i = x_m + x_{m+1} + x_{m+2} + \cdots + x_{n-1} + x_n.$	$\sum_{n=0}^{\infty} a_n = a_0 + a_1 + a_2 + \cdots$
4	$\sum_{n=0}^{\infty} a_n = a_0 + a_1 + a_2 + \cdots$	$\sum_{n=0}^{\infty} a_n = a_0 + a_1 + a_2 + \cdots$	$\sum_{n=0}^{\infty} a_n = a_0 + a_1 + a_2 + \cdots ;$
5	$\sum_{n=0}^{\infty} a_n = a_0 + a_1 + a_2 + \cdots ;$	$a_1 + a_2 + a_3 + \cdots + a_n = \sum_{i=1}^{n} a_i$	$\sum_{n=0}^{\infty} a_n = a_0 + a_1 + a_2 + \cdots .$
6	$a_1 + a_2 + a_3 + \cdots + a_n = \sum_{i=1}^{n} a_i$	$\sum_{n=0}^{\infty} a_n = a_0 + a_1 + a_2 + \cdots ;$	$a_1 + a_2 + a_3 + \cdots + a_n = \sum_{i=1}^{n} a_i$
7	$a_1 + \cdots + a_{n-1} \leq b_1 + \cdots + b_{n-1}$	$S_k = \sum_{n=0}^{k} a_n = a_0 + a_1 + \cdots + a_k.$	$N = (a_1 + a_2 + a_3 + \cdots + a_n)^2.$
8	$a_1 + \cdots + a_n = b_1 + \cdots + b_n.$	$a_n = a_{n-1} + a_{n-2} + a_{n-3}$	$\lvert a_{n+1} + a_{n+2} + \cdots + a_{n+p}\rvert < \varepsilon$
9	$\sum_{n=1}^{p} a_n^{+} + \sum_{n=1}^{q} a_n^{-} = a_{\sigma(1)} + \cdots + a_{\sigma(m_1)} + a_{\sigma(m_1+1)} + \cdots + a_{\sigma(n_1)},$	$\lvert a_{n+1} + a_{n+2} + \cdots + a_{n+p}\rvert < \varepsilon$	$a_1 + \cdots + a_{n-1} \leq b_1 + \cdots + b_{n-1}$
10	$S_k = \sum_{n=0}^{k} a_n = a_0 + a_1 + \cdots + a_k.$	$\sum_{n=1}^{p} a_n^{+} + \sum_{n=1}^{q} a_n^{-} = a_{\sigma(1)} + \cdots + a_{\sigma(m_1)} + a_{\sigma(m_1+1)} + \cdots + a_{\sigma(n_1)},$	$a_1 + a_2 + \cdots + a_n$

Query 9: $f(x;\mu,c) = \sqrt{\frac{c}{2\pi}}\ \frac{e^{-\frac{c}{2(x-\mu)}}}{(x-\mu)^{3/2}}$

Rank	Lucene
1	$\int \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}\ \mathrm{d}x = \frac{1}{2}\left(\mathrm{erf}\ \frac{x-\mu}{\sigma\sqrt{2}}\right)$
2	$g(x;\mu,\sigma) = f\left(x;\mu,\frac{\sigma\sqrt{3}}{\pi}\right) = \frac{1}{4}\frac{\pi}{\sigma\sqrt{3}}\,\mathrm{sech}^2\left(\frac{\pi}{2\sqrt{3}}\ \frac{x-\mu}{\sigma}\right).$
3	$f(x;\mu,c) = \sqrt{\frac{c}{2\pi}}\ \frac{e^{-\frac{c}{2(x-\mu)}}}{(x-\mu)^{3/2}}$
4	$\lim_{x\to\infty} f(x;\mu,c) = \sqrt{\frac{c}{2\pi}}\ \frac{1}{x^{3/2}}.$
5	$t = \frac{\arccos\left(\sqrt{\frac{x}{r}}\right)+\sqrt{\frac{x}{r}\ (1-\frac{x}{r})}}{\sqrt{2\mu}}\ r^{3/2}$
6	$\mu(X) = \sigma\sqrt{\frac{\pi}{2}}\ \approx 1.253\sigma$
7	$v_j(x) = \sqrt{\frac{2}{L}}\sin(\frac{j\pi x}{L})$
8	$f(x;\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}}\ e^{-\frac{(x-\mu)^2}{2\sigma^2}}$
9	$K\left(\frac{\sqrt{2}}{2}\right) = \frac{1}{4\sqrt{\pi}}\ \Gamma\left(\frac{1}{4}\right)^2$
10	$x(u) = r\cos u; \qquad z(u) = r\sqrt{2}\left[E(u,\frac{1}{\sqrt{2}}) - \frac{1}{2}F(u,\frac{1}{\sqrt{2}})\right]$ for $u\in[0,\frac{\pi}{2}]$

Rank	Tangent F-Measure	Tangent Distance	Tangent Prefix
1	$f(x;\mu,c) = \sqrt{\frac{c}{2\pi}}\ \frac{e^{-\frac{c}{2(x-\mu)}}}{(x-\mu)^{3/2}}$	$f(x;\mu,c) = \sqrt{\frac{c}{2\pi}}\ \frac{e^{-\frac{c}{2(x-\mu)}}}{(x-\mu)^{3/2}}$	$f(x;\mu,c) = \sqrt{\frac{c}{2\pi}}\ \frac{e^{-\frac{c}{2(x-\mu)}}}{(x-\mu)^{3/2}}$
2	$\sqrt{\frac{c}{2\pi}}\ \frac{e^{-\frac{c}{2(x-\mu)}}}{(x-\mu)^{3/2}}$	$\sqrt{\frac{c}{2\pi}}\ \frac{e^{-\frac{c}{2(x-\mu)}}}{(x-\mu)^{3/2}}$	$\sqrt{\frac{c}{2\pi}}\ \frac{e^{-\frac{c}{2(x-\mu)}}}{(x-\mu)^{3/2}}$
3	$\lim_{x\to\infty} f(x;\mu,c) = \sqrt{\frac{c}{2\pi}}\ \frac{1}{x^{3/2}}.$	$\lim_{x\to\infty} f(x;\mu,c) = \sqrt{\frac{c}{2\pi}}\ \frac{1}{x^{3/2}}.$	$\lim_{x\to\infty} f(x;\mu,c) = \sqrt{\frac{c}{2\pi}}\ \frac{1}{x^{3/2}}.$
4	$F(x;\mu,c) = \mathrm{erfc}\left(\sqrt{\frac{c}{2(x-\mu)}}\right)$	$F(x;\mu,c) = \mathrm{erfc}\left(\sqrt{\frac{c}{2(x-\mu)}}\right)$	$f(x;c,k) = ck\frac{x^{c-1}}{(1+x^c)^{k+1}}$
5	$f(x;\mu,\lambda) = \left[\frac{\lambda}{2\pi x^3}\right]^{1/2}\exp\frac{-\lambda(x-\mu)^2}{2\mu^2 x}$	$f(x;\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}}\ e^{-\frac{(x-\mu)^2}{2\sigma^2}}$	$M(t;c)\ \stackrel{\mathrm{def}}{=}\ \sqrt{\frac{c}{2\pi}}\int_0^\infty \frac{e^{-c/2x+tx}}{x^{3/2}}\,dx$
6	$f(x;\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}}\ e^{-\frac{(x-\mu)^2}{2\sigma^2}}$	$f(x;\mu,\lambda) = \left[\frac{\lambda}{2\pi x^3}\right]^{1/2}\exp\frac{-\lambda(x-\mu)^2}{2\mu^2 x}$	$F(x;\mu,\sigma) = \left[\left(\frac{e^\mu}{x}\right)^{\pi/(\sigma\sqrt{3})}+1\right]^{-1}.$
7	$f(x;\mu,\sigma) = \frac{1}{\sqrt{2\pi\sigma^2}}e^{-(x-\mu)^2/(2\sigma^2)}.$	$f(x;\mu,\sigma) = \frac{1}{\sqrt{2\pi\sigma^2}}e^{-(x-\mu)^2/(2\sigma^2)}.$	$\mathrm{erfc}\left(\sqrt{\frac{c}{2(x-\mu)}}\right)$
8	$p(x\mid\mu,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}}e^{-\frac{(x-\mu)^2}{2\sigma^2}}$	$\mathrm{erfc}\left(\sqrt{\frac{c}{2(x-\mu)}}\right)$	$f(x\mid\mu,\kappa) =$
9	$f(x;\mu,b,n) = \frac{b^n(x-\mu)^{-1-n}}{\left(e^{\frac{b}{x-\mu}}-1\right)\Gamma(n)\zeta(n)}$	$f(x) = \sqrt{\frac{\tau}{2\pi}}\ e^{\frac{-\tau(x-\mu)^2}{2}}.$	$f_L(x;\mu,\sigma) = \prod_{i=1}^n\left(\frac{1}{x\,i}\right)\ f_N(\ln x;\mu,\sigma)$
10	$f(x\mid\mu,b) = \frac{1}{2bx}\exp\left(-\frac{\lvert\ln x-\mu\rvert}{b}\right)$	$p(x\mid\mu,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}}e^{-\frac{(x-\mu)^2}{2\sigma^2}}$	$f(x;\mu,b,n) = \frac{b^n(x-\mu)^{-1-n}}{\left(e^{\frac{b}{x-\mu}}-1\right)\Gamma(n)\zeta(n)}$

Query 10: $D4\sigma = 4\sigma = 4\sqrt{\frac{\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} I(x,y)(x-\bar{x})^2\,dx\,dy}{\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} I(x,y)\,dx\,dy}}$

Rank	Lucene
1	$\bar{x} = \frac{\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} I(x,y)x\,dx\,dy}{\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} I(x,y)\,dx\,dy}$
2	$D4\sigma = 4\sigma = 4\sqrt{\frac{\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} I(x,y)(x-\bar{x})^2\,dx\,dy}{\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} I(x,y)\,dx\,dy}}$
3	$M_{pq} = \int\limits_{-\infty}^{\infty}\int\limits_{-\infty}^{\infty} x^p y^q f(x,y)\,dx\,dy$
4	$\int_1^\infty \left(\int_1^\infty \frac{x^2-y^2}{(x^2+y^2)^2}\ dy\right)\ dx = -\frac{\pi}{4}\ .$
5	$\int_0^\infty\ x^4 e^{-\alpha x^2} dx = \frac{1}{2}\int_0^\infty\ x^{2((5/2)-1)} e^{-\alpha x^2} d(x^2)$
6	$\Phi(x) = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{x} e^{-\frac{y^2}{2}}\ dy = 1 - \frac{1}{\sqrt{2\pi}}\int_x^\infty e^{-\frac{y^2}{2}}\ dy$
7	$(F,G)'_\sigma = \displaystyle\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} F(x)\overline{G(y)}\|x-y\|^{2\sigma-1}\,dx\,dy.$
8	$= \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{\infty} e^{\frac{-x^2}{2}(1-2t)}\ dx.$
9	$\int_1^\infty \left(\int_1^\infty \frac{x^2-y^2}{(x^2+y^2)^2}\ dx\right)\ dy = \frac{\pi}{4}\ .$
10	$\Phi(a) = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{a} e^{-\frac{x^2}{2}}\ dx$

Rank	Tangent F-Measure	Tangent Distance	Tangent Prefix
1	$D4\sigma = 4\sigma = 4\sqrt{\frac{\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} I(x,y)(x-\bar{x})^2\,dx\,dy}{\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} I(x,y)\,dx\,dy}}$	$D4\sigma = 4\sigma = 4\sqrt{\frac{\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} I(x,y)(x-\bar{x})^2\,dx\,dy}{\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} I(x,y)\,dx\,dy}}$	$D4\sigma = 4\sigma = 4\sqrt{\frac{\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} I(x,y)(x-\bar{x})^2\,dx\,dy}{\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} I(x,y)\,dx\,dy}}$
2	$\bar{x} = \frac{\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} I(x,y)x\,dx\,dy}{\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} I(x,y)\,dx\,dy}$	$\bar{x} = \frac{\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} I(x,y)x\,dx\,dy}{\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} I(x,y)\,dx\,dy}$	$\bar{x} = \frac{\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} I(x,y)x\,dx\,dy}{\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} I(x,y)\,dx\,dy}$
3	$\int_0^1\int_0^1 f(x,y)\,dy\,dx \neq \int_0^1\int_0^1 f(x,y)\,dx\,dy.$	$\mu_{pq} = \int\limits_{-\infty}^{\infty}\int\limits_{-\infty}^{\infty} (x-\bar{x})^p(y-\bar{y})^q f(x,y)\,dx\,dy$	$P = \int I(x,y)dxdy$
4	$\mu_{pq} = \int\limits_{-\infty}^{\infty}\int\limits_{-\infty}^{\infty} (x-\bar{x})^p(y-\bar{y})^q f(x,y)\,dx\,dy$	$\int_0^1\int_0^1 f(x,y)\,dy\,dx \neq \int_0^1\int_0^1 f(x,y)\,dx\,dy.$	$V = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)\,dxdy = 2\pi A\sigma_x\sigma_y.$
5	$T(w) = \frac{\int_{w/2}^{1}\int_0^{\sqrt{1-x^2}} f(x,y)f^*(x-w,y)dydx}{\int_0^1\int_0^{\sqrt{1-x^2}} f(x,y)^2dydx}$	$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \|x-y\| f(x,y)\,dx\,dy.$	$\int_{[a,b]}\int_{[c,d]} f(x,y)\,dxdy = 0$
6	$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \|x-y\| f(x,y)\,dx\,dy.$	$E[T] = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} t(x,y)f(x,y)\,dy\,dx$	$\langle x^2\rangle = \frac{1}{P}\int I(x,y)(x-\langle x\rangle)^2 dxdy,$
7	$\langle x^2\rangle = \frac{1}{P}\int I(x,y)(x-\langle x\rangle)^2 dxdy,$	$T(w) = \frac{\int_{w/2}^{1}\int_0^{\sqrt{1-x^2}} f(x,y)f^*(x-w,y)dydx}{\int_0^1\int_0^{\sqrt{1-x^2}} f(x,y)^2dydx}$	$m = \int\int \rho\ (x,y)\,dx\,dy$
8	$\int_{C_4} L(x,y)\,dx = \int_{C_2} L(x,y)\,dx = 0.$	$\langle x^2\rangle = \frac{1}{P}\int I(x,y)(x-\langle x\rangle)^2 dxdy,$	$(x+y)(x-y) = x^2 - y^2$
9	$E[T] = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} t(x,y)f(x,y)\,dy\,dx$	$V = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)\,dxdy = 2\pi A\sigma_x\sigma_y.$	$\int_{[0,1]\times[0,1]} f(x,y)\,dx\,dy$
10	$P = \int I(x,y)dxdy$	$\int_{C_4} L(x,y)\,dx = \int_{C_2} L(x,y)\,dx = 0.$	$T(w) = \frac{\int_{w/2}^{1}\int_0^{\sqrt{1-x^2}} f(x,y)f^*(x-w,y)dydx}{\int_0^1\int_0^{\sqrt{1-x^2}} f(x,y)^2dydx}$

Bibliography

[1] Dominique Archambault and V Moço. Canonical MathML to simplify conversion of MathML to Braille mathematical notations. *Computers Helping People with Special Needs*, pages 1191–1198, 2006.

[2] Shahab Kamali, Johnson Apacible, and Yasaman Hosseinkashi. Answering math queries with search engines. *Proceedings of the 21st international conference companion on World Wide Web - WWW '12 Companion*, pages 43–52, 2012.

[3] Shahab Kamali and Frank Wm. Tompa. A New Mathematics Retrieval System. *Proceedings of the 19th ACM international conference on Information and knowledge management - CIKM '10*, pages 1413–1416, 2010.

[4] Shahab Kamali and Frank Wm. Tompa. Structural Similarity Search for Mathematics Retrieval. *Intelligent Computer Mathematics*, pages 246–262, 2013.

[5] Michael Kohlhase and I Sucan. A search engine for mathematical formulae. *Artificial Intelligence and Symbolic Computation*, pages 23–24, 2006.

[6] PP Kumar, Arun Agarwal, and Chakravarthy Bhagvati. A Structure Based Approach for Mathematical Expression Retrieval. *Multi-disciplinary*

Trends in Artificial Intelligence, pages 23–34, 2012.

[7] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Number c. 2008.

[8] Bruce R. Miller and Abdou Youssef. Technical aspects of the digital library of mathematical functions. *Annals of Mathematics and Artificial Intelligence*, 38:121–136, 2003.

[9] Rajesh Munavalli and Robert Miner. MathFind: a math-aware search engine. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, volume 26 of *SIGIR '06*, page 735, New York, NY, 2006. ACM.

[10] G Salton and M J McGill. *Introduction to modern information retrieval*. McGraw-Hill computer science series. McGraw-Hill, 1983.

[11] Christopher Sasarak, Kevin Hart, Richard Pospesel, David Stalnaker, Lei Hu, Robert Livolsi, Siyu Zhu, and Richard Zanibbi. min : A Multimodal Web Interface for Math Search. In *Symp. Human-Computer Interaction and Information Retrieval*, Cambridge, MA, 2012. `http://ils.unc.edu/hcir2012/hcir2012\_submission\_11.pdf`.

[12] Matthew Schellenberg. *Layout-based substitution tree indexing and retrieval for mathematical expressions*. Msc thesis, Rochester Institute of Technology, 2011.

[13] Thomas Schellenberg, Bo Yuan, and Richard Zanibbi. Layout-based substitution tree indexing and retrieval for mathematical expressions. *Proc. Document Recognition and Retrieval XIX*, pages 8297OI–1 – 8297OI–8, January 2012.

[14] Petr Sojka and Martin Líška. Indexing and Searching Mathematics in Digital Libraries. In James H. Davenport, William M. Farmer, Josef Urban, and Florian Rabe, editors, *Intelligent Computer Mathematics*, volume 6824 of *Lecture Notes in Computer Science*, pages 228–243. Springer Berlin Heidelberg, 2011.

[15] Petr Sojka and Martin Líška. The Art of Mathematics Retrieval. *Proceedings of the 11th ACM symposium on Document engineering*, pages 57–60, 2011.

[16] A Youssef. Methods of Relevance Ranking and Hit-content Generation in Math Search. *Towards Mechanized Mathematical Assistants*, pages 1–15, 2007.

[17] Abdou Youssef. Roles of math search in mathematics. *Mathematical Knowledge Management*, (0208818):2–16, 2006.

[18] Richard Zanibbi and Dorothea Blostein. Recognition and retrieval of mathematical expressions. *International Journal on Document Analysis and Recognition (IJDAR)*, 15(4):331–357, October 2011.

[19] Richard Zanibbi and Bo Yuan. Keyword and image-based retrieval of mathematical expressions. *Proc. Document Recognition and Retrieval XVIII*, 7874:0I1–0I8, 2011.

[20] Justin Zobel and Alistair Moffat. Inverted files for text search engines. *ACM Computing Surveys (CSUR)*, 38(2), 2006.