Choosing Math Features for BM25 Ranking with Tangent-L

Dallas Fraser
Cheriton School of Computer Science
University of Waterloo
Waterloo, Ontario, Canada
d6fraser@uwaterloo.ca

Andrew Kane
Cheriton School of Computer Science
University of Waterloo
Waterloo, Ontario, Canada
arkane@uwaterloo.ca

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

ABSTRACT

Combining text and mathematics when searching in a corpus with extensive mathematical notation remains an open problem. Recent results for Tangent-3 on the math and text retrieval task at NTCIR-12, for example, have room for improvement, even though formula retrieval appeared to be fairly successful.

This paper explores how to adapt the state-of-the-art BM25 text ranking method to work well when searching for math together with text. Following the approach proposed for the Tangent math search system, we use symbol layout trees to represent math formulae. We extract features from the symbol layout trees to serve as search terms to be ranked using BM25 and then explore the effects on retrieval performance of various classes of features. Based on the results, we recommend which features can be used effectively in a conventional text-based retrieval engine. We validate our overall approach using a NTCIR-12 math and text benchmark.

CCS CONCEPTS

• **Information systems → Content analysis and feature selection**; **Specialized information retrieval**; *Digital libraries and archives*;

KEYWORDS

Mathematics information retrieval (MIR), Mathematical content representation, MathML, Okapi BM25, Lucene

ACM Reference Format:
Dallas Fraser, Andrew Kane, and Frank Wm. Tompa. 2018. Choosing Math Features for BM25 Ranking with Tangent-L. In *DocEng '18: ACM Symposium on Document Engineering 2018, August 28–31, 2018, Halifax, NS, Canada.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3209280.3209527

1 INTRODUCTION

Mathematical formulae are included in documents from many different fields of study, spanning economics to physics. As such, an understanding of published mathematics is vital for developing new mathematics and its applications. Unfortunately, searching for mathematics within a corpus of such documents is not an easy task. The difficulty of searching for mathematical formulae using traditional search engines has inspired some to build their own digital mathematical libraries [17]. Sojka and Líška discuss the unresolved math search problem when integrating existing digital math libraries into EuDML [23]. In 2011, they evaluated several systems and concluded that there were still no satisfactory systems for searching mathematics. Indeed, seven years later it still remains a challenge to build an effective mathematics retrieval engine.

One frequently cited challenge in math information retrieval (MIR) is the lack of benchmarks. Prior to 2013, proposers for each math system had to create their own document corpus and corresponding set of queries, but it is time-consuming and expensive to develop reliable relevancy judgments for these queries. At the NTCIR-10 conference in 2013, the first Math Pilot Task was introduced [1], initiating a common workbench for evaluating math retrieval engines. This continued in subsequent years with the NTCIR-11 Math-2 task [2] and NTCIR-12 MathIR tasks [27], which attracted several participants from around the world. Even though systems have been improving, some of the results for the various NTCIR-12 tasks show that more work is still needed to design an effective math retrieval system.

Existing MIR systems are highly specialized, complicated, and often inefficient. We hypothesize that an effective search engine for math and text together can be created by adapting an effective text search engine to handle queries that include mathematical formulae. To prove this hypothesis, we build on the standard Lucene[1] text search platform and add methods adapted from the Tangent-3 math search engine [8], thus forming a new *Tangent-L* math search system variant. During this process, the following contributions are made:

1. We show how Okapi's BM25 text-based ranking can be applied to features extracted from the representation of mathematical formulae that may include wild card symbols.
2. We evaluate the effects of several math "presentation" features on retrieval effectiveness.
3. We investigate how to balance search for math formula query terms with traditional keyword search.
4. We recommend a set of math features to use in MIR search and a weighting between math and keyword search that achieves effective retrieval results.

Section 2 discusses how one can represent mathematics and briefly describes several state-of-the-art MIR systems. This section concludes with an introduction to Tangent and summarizes some of the difficulties encountered by Tangent-3 [8]. Section 3 describes how approaches used in Tangent can be adapted to work with BM25. Section 4 introduces several novel math features suitable for our new Tangent-L MIR system, and these are evaluated in

[1]https://lucene.apache.org

Section 5. The paper concludes with some recommendations and a brief description of future work.

2 RELATED WORK

This section provides a brief overview of existing work on which this paper depends. The survey by Guido and Sacerdoti Coen provides a more thorough review of related literature [13].

2.1 Formula Representations

When mathematical expressions are included in a document, they are typically linearized and then encoded as text, using a language such as LaTeX. One benefit of this representation is that math formulae can be tokenized and then indexed along with the text in a conventional search engine. Unfortunately simple tokenization loses structural information, which limits its effectiveness during retrieval [15], even after applying canonical orderings, unification of variables, and the replacement of all constants by a single shared symbol.

An alternative text encoding is MathML, a W3C recommendation [6] proposed for exchanging mathematics between software tools. MathML is supported by the Firefox browser, and the MathML Association is working to improve support in browsers. Because most scientific authors write mathematics in LaTeX, MathML is often produced by converting LaTeX into MathML using tools such as MathJax [7] or LaTeXML [12].

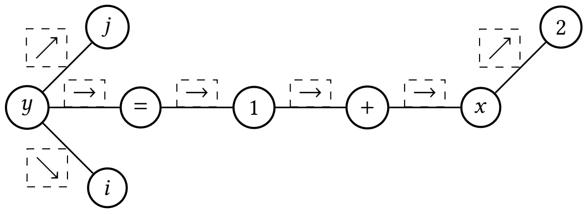
Even though MathML is a text representation, it reflects the structure of a mathematical formula as a tree: the structure can be encoded in Presentation MathML, representing how symbols are placed relative to each other on a printed page; alternatively the structure can be encoded in Content MathML, representing how mathematical expressions are built from operators and subexpressions. These two approaches to representing math expressions can be generalized as *Symbol Layout Trees* (*SLTs*) and *Operator Trees* (*OTs*), where SLTs are used to represent formulae visually and OTs are used to represent math expressions semantically, as illustrated in Figure 1. Both forms have been used to represent math formulae in MIR systems.

2.2 MIR Systems

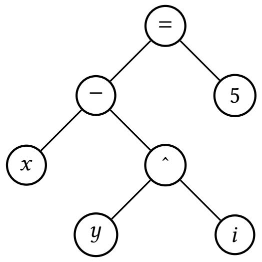
This section provides a brief overview of the state-of-the-art MIR systems that participated in the NTCIR-12 MathIR tasks and are summarized in Table 1.

MCAT [14] encodes path information and sibling information as features extracted from the trees representing math formulae. In addition, it stores several features computed as hash-based signatures that each captures some global properties of the trees, such as polynomial degrees and repeated use of variable names.

In addition to the various encodings of math formulas, for each math expression MCAT stores information that captures words used in its proximity, words inside a certain context window size, and noun phrases. One indexed feature is the set of all words used within a context window size of 20 surrounding a math expression. In addition, all nouns that appear in the same sentence as the math expression are stored as well. A math formula that appears multiple times has its context words combined. MCAT also creates a math dependency graph, where a directed edge from expression-1 to expression-2 indicates that the former expression contains the latter as a subexpression. This helps MCAT relate the text associated with one expression to the text associated with related expressions, which is useful when an expression is presented and named in one paragraph and is used subsequently as a subexpression within another named formula.



(a) SLT for $y_i^j = 1 + x^2$



(b) OT for $x - y^j = 5$

Figure 1: Examples of a Symbol Layout Tree and an Operator Tree

MCAT is the top-performing search engine for all NTCIR-12 MathIR tasks except for the MathWiki Task.

Math Indexer and Searcher (MIaS) [21] uses Presentation MathML for each formula, encoding it in a compact string representation. Each formula and all its logical subparts are subjected to the following transformations:

(1) ordering: the formula is rewritten in a canonical form,
(2) unification of variables: all variables are replaced by a unified symbol,
(3) unification of constants: all constants are replaced by a unified symbol,
(4) unification of operators: all additive operators are replaced by a unified symbol,
(5) structural unification: substructures are replaced by a unified symbol in a series of layers that are based on the MathML tree,

Weights are assigned to specify the specificity of a formula: version with fewer replacements of unification symbols are weighted higher. All the unified versions of a formula are stored when indexing.

Given a query, MIaS poses independent searches for several subsets of the query's keywords and formulae. The results from these subqueries are interleaved with an approach dubbed strip-merging [20]: the m subqueries are considered in increasing order of the number of keywords or formulae removed; then the first m results are taken from the first subquery, the first $m - 1$ results are taken from the second subquery, and so on until only the top

Table 1: Participant System Configurations for NTCIR-12 arXiv Main Task [27]

System	*Presentation MathML*	*Content MathML*	*Tree Structure*	*Search Platform*
MCAT	Yes	Yes	Yes	Solr
MIaS*	Yes/No	Yes/No	Yes	Lucene
SMSG5	Yes	No	No	Elastic Search
Tangent-3	Yes	No	Yes	Lucene for Text and Custom for Math
WikiMIR	Yes	No	Yes	Custom

* Had multiple configurations with some using both Presentation MathML and Content MathML and some using just one.

result is taken from the last subquery; then the next m results are taken from the first subquery, and so on until the desired number of results have been produced.

SMSG5 [25] uses Elastic Search to index text and formulae with all variables unified in addition to the original formulae. A search then returns the top-k documents ranked using basic term frequency/inverse document frequency (tf·idf) scoring. SMSG5 then forms three other re-ranked orders of those results using:

- Doc2Vec-based scoring,
- LDA-based scoring,
- pattern-based scoring

where the Doc2Vec-based scoring applies the paragraph vector distributed bag of words (PV-DBOW) model [16], extended to represent two-dimensional expression trees as dense vectors; the LDA-based scoring uses a Latent Dirichlet Allocation model [4] to search for documents that match the topics implied by the query formulae and keywords; and the pattern-based scoring is derived from associating the defining vocabulary for a formula with the formula itself based on natural language patterns observed in the corpus being searched.

The documents are finally re-ranked using Borda Count, an algorithm that combines the preferences of many experts [3]. For SMSG5, each of the ranked lists is considered to be a voter where its top ranked documents gets c votes and its second ranked documents gets $c - 1$ votes. Unfortunately, the evaluation of SMSG5 against the NTCIR-12 arXiv Main Task benchmark reflects the preliminary ranking mechanism only, and its performance suffers when no re-ranking is enabled.

Tangent-3 [8] uses features extracted from the Presentation MathML representations of all math formulae to rank documents based on their math formulae. The truncated list of math results is combined with the top ranked documents returned from matching the keyword components of a query. A detailed description is provided in Section 2.3.

WikiMir [11] is a hybrid MIR system that uses operator trees (OTs) as its representation for math expressions. The OTs are traversed recursively at different levels to extract both original and generalized formula terms. WikiMir uses RankBoost [10] to learn how to rank retrieved formulae. For RankBoost, a collection of weak learners are used to produce a score for some feature, and each weak learner is assigned a learned weight obtained from training. A final score is produced by summing each weak learner multiplied by its weight. Finally, WikiMir re-ranks the top-k matches using regular expressions to match document formulae against query formulae that include wild cards. The features used when learning to rank cover a broad spectrum of sources, including the individual formulae, the individual documents, and the match between query formulae and formulae found in the documents.

This system achieves the best results on the NTCIR-12 MathWiki Task, perhaps because the training data is obtained from Wikipedia, thus overfitting to this dataset.

2.3 Tangent-3

2.3.1 Formula Representation. Tangent-3 is a two-stage formula retrieval system [28]. First Presentation MathML is transformed into a simpler representation of a SLT. Nodes represent operators, variables, constants, fractions, matrices, function arguments, and parenthesized expressions, and each node label reflects its type and its value:

- numbers (N!n)
- variables names (V!v)
- text fragments (T!t)
- fractions (F!)
- radicals (R!)
- matrices, tabular structures and parenthesized expressions (M!$frxc$)
- wildcard symbols (*w)
- mathematical operators

Each directed edge has a label that reflects the spatial relationship from the source node to the target node:

(1) next ($\rightarrow$)
(2) within ($\boxdot$)
(3) element ($\in$)
(4) above ($\nearrow$)
(5) below ($\searrow$)
(6) pre-above ($\nwarrow$)
(7) pre-below ($\swarrow$)
(8) over ($\uparrow$)
(9) under ($\downarrow$)

Given a SLT representing a math formula, Tangent-3 extracts so-called *symbol pairs*, which are features in the form of triples (s_1, s_2, R), where s_1 and s_2 are two symbols and R is the sequence of labels on the directed path between the nodes labelled by those symbols. The symbol pairs corresponding to the SLT in Figure 1 are shown in Table 2.

2.3.2 Search Strategy. Symbol pairs form the search units used by Tangent-3's core engine, which contains an inverted index mapping triples to formulae within documents. The core engine assigns a score to each formula within each document based on the similarity of a query and formula as reflected by Dice's coefficient. After the core engine retrieves a large ranked list of formulae, they are

Table 2: A list of symbol pairs for $y_i^j = 1 + x^2$

S_1	S_2	R
$V!y$	$V!j$	$\nearrow$
$V!y$	$V!i$	$\searrow$
$V!y$	$=$	$\rightarrow$
$=$	$N!1$	$\rightarrow$
$N!1$	$+$	$\rightarrow$
$+$	$V!x$	$\rightarrow$
$V!x$	$N!2$	$\nearrow$
$V!y$	$N!1$	$\rightarrow\rightarrow$
$=$	$+$	$\rightarrow\rightarrow$
$N!1$	$V!x$	$\rightarrow\rightarrow$
$+$	$N!2$	$\rightarrow\nearrow$
$V!y$	$+$	$\rightarrow\rightarrow\rightarrow$
$=$	$V!x$	$\rightarrow\rightarrow\rightarrow$
$N!1$	$N!2$	$\rightarrow\rightarrow\nearrow$
$V!y$	$V!x$	$\rightarrow\rightarrow\rightarrow\rightarrow$
$=$	$N!2$	$\rightarrow\rightarrow\rightarrow\nearrow$
$V!y$	$N!2$	$\rightarrow\rightarrow\rightarrow\rightarrow\nearrow$

re-ranked using Maximum Subtree Similarity (MSS) [28] to select the final top-k to be returned to the user. The MSS metric examines SLTs so as to account for symbol unification and to rank subexpressions based on structural similarity. This evaluation is far more costly than using conventional scoring functions for text retrieval, thus limiting the number of results that can be re-ranked.

2.3.3 Beyond Formula Retrieval. Tangent-3 produced state-of-the-art results for the NTCIR-11 Wikipedia formula retrieval task, which focuses solely on formula retrieval [28]. However, general mathematical information retrieval must support queries that include both math expressions and keywords. To this end, Tangent-3 supplements the two-stage engine for retrieving math formula with a Lucene index for the remainder of the text, combining the results from each using a linear weighted combination of their scores [8]. However, the disappointing results for the NTCIR-12 arXiv Main Task indicate that separate retrieval models for math and text may be the wrong approach to adopt [27].

3 APPLYING TRADITIONAL TEXT RETRIEVAL TO MATH

This paper explores using traditional text retrieval approaches for formula retrieval so that it can be easily integrated with keyword retrieval, thus combining text and math at a low level to improve effectiveness, while using a single inverted index for efficiency. The first aim is to determine how traditional text retrieval methods can be applied to formula retrieval to achieve results that are comparable to using expensive math-specific scoring functions. If such methods are adequate, they may provide a pathway to directly combining formula retrieval and text retrieval, allowing for better support of queries that include both text and math.

3.1 Applying BM25 to Math

The state-of-the-art scoring function used in traditional search engines is Okapi BM25 [19]. A variant, BM25+ [18], has been shown to have comparable performance to BM25 in general but to be more effective for long documents [26]. We use this variant because preliminary experiments with our datasets show it to be slightly more effective than the standard implementation of BM25.

Given a collection of documents D containing $|D|$ documents and a query q consisting of a set of query terms, the score for a document $d \in D$ is given by

$$\mathrm{BM25}^{+}(q,d) = \sum_{w \in q} \left(\frac{(k+1)tf_w}{k\left(1.0 - b + b\frac{|d|}{\bar{d}}\right) + tf_w} + \delta \right) \log\left(\frac{|D|+1}{|D_w|}\right)$$

where k, b, and δ are constants (following common practice, chosen to be 1.2, 0.75, and 1, respectively); tf_w is the number of occurrences of term w in document d; $|d|$ is the total number of terms in document d; $\bar{d} = \sum_{d \in D} \frac{|d|}{|D|}$; and $|D_w|$ is the number of documents in D containing term w.

In order to apply BM25[2] to the query parts that represent mathematical formulae, those parts must be expressed as a set of terms that are indexed similarly to other text terms in the document collection. To this end, a math formula is first converted to a collection of features and then each feature is encoded as a string that is tokenized as a single term. For example, each symbol pair in Tangent can be used as a feature, and the triple $(V!x, N!2, \nearrow)$ can be encoded unambiguously as `V_x__N_2__a`, which can then be treated as a token to be indexed by a conventional search engine and considered exactly like any other text term by a standard BM25 scoring function. Potential features to use when a formula is represented by a SLT are described in Section 4.

Two complications arise from considering formula features as terms to be evaluated by BM25: how to handle features that occur more than once within a query formula and how to handle wild card symbols in a query formula. Consider, for example, the query formula $\frac{x+1}{?+1}$ where ? can match any symbol or subexpression. The list of symbol pairs for this formula includes two instances of $(+, N!1, \rightarrow)$ and an instance of $(?, +, \rightarrow)$, among others.

3.1.1 Dealing with Repeated Features. Although BM25 is normally defined as accepting a set of query terms, in fact it can easily be applied to a bag of query terms; if a term is repeated in the query, the corresponding score for that term is naïvely accumulated multiple times. Is this the best approach to adopt for formula retrieval?

In fact, this raises the question of the appropriateness of considering term frequency when searching for math formulae. Through extensive experience, it has been shown that documents that have many instances of a term t are more likely to be relevant to a query that includes query term t than documents that have fewer occurrences of that term. However, it is difficult to argue that when searching for a formula that contains an occurrence of +1, formulae with more occurrences of +1 are likely to be more relevant than formulae with only one occurrence. In fact, it would seem that the best matches may well be the ones in which the number of occurrences of +1 closely matches the number of occurrences of +1 in the query formula. Should BM25 for math be changed to try

[2]For the remainder of the paper, the use of BM25 refers to BM25+ unless otherwise indicated.

to match repetitions in query formulae and formula appearances in the document collection?

Our preliminary experiments have determined that it makes very little difference whether repetitions are included or ignored in either the query or the document formulae. Therefore, to maintain compatibility with conventional practice in search engines, we recommend treating the features extracted from a query formula as a set (i.e., ignoring duplicate terms), but continuing to use term frequency in scoring with BM25.

3.1.2 Dealing with Wild Cards. When a feature extracted from a query formula includes a wild card symbol ("query variables"), it is presumed to match a corresponding feature from a document's formula where that symbol is replaced by any symbol whatsoever. Thus, for example, the symbol pair $(?, +, \rightarrow)$ can match $(V!x, +, \rightarrow)$, $(N!3, +, \rightarrow)$, or $(M!()2x3, +, \rightarrow)$. (This approach is adequate even if the user's intention is that a wild card should match a subexpression that is larger than a single symbol: the system effectively will succeed in matching the last symbol appearing on the baseline of of that subexpression.)

There are two possible techniques for matching features that contain wild cards. First, at query time, a search for a feature that includes a wild card symbol can be expanded to search for all possible symbol substitutions. In practice this means that at query time we need to find all postings lists that correspond to symbol substitutions for the wild card and then consider each list independently or combine them in some way. The core engine of Tangent-3 adopts this approach: if a query symbol pair includes a wild card, the query engine forms the union of all postings lists that correspond to symbol pairs where the wild card is replaced by some other symbol and uses that list as part of its normal processing. To reduce excessive computation, Tangent-3 ignores symbol pairs that include more than one wild card symbol, and it ignores the symbol pair representing a wild card symbol followed by the end of a line. It is unclear, however, how to modify an existing BM25-based search engine to compute the score for a document using this technique without making significant changes to the engine's scoring subsystem.

Alternatively, at indexing time, in addition to creating a postings list for each feature, one or more postings lists can be created for synthetic features that include a wild card symbol. Thus if a formula includes a symbol pair $(V!x, +, \rightarrow)$, the index could include an entry on the postings list for that feature and also entries on the postings lists for the synthetic features $(?, +, \rightarrow)$ and $(V!x, ?, \rightarrow)$, that is, something precedes a plus sign and something comes after an x. Again, to avoid excessive computation (and to save index space), synthetic features can be limited to including one wild card only, and some features might not be suitable for creating any corresponding synthetic features at all. The advantage of using this technique is that a search for a feature that includes a wild card can be processed by a BM25-based engine without making any alteration to the search execution or scoring system. A disadvantage is that this approach assumes that multiple instances of a wild card in a query are indistinguishable, as all wild card symbols are effectively unified.

Table 3: A list of terminal symbols for $y_i^j = 1 + x^2$

S	!0
$V!j$	!0
$V!i$	!0
$N!2$	!0

4 NEW FEATURES

The principal features used in Tangent-3 are symbol pairs corresponding to parents and children in the SLT, that is, where the path length between the nodes labelled by those symbols has length one [28]. Additional features have the potential to increase retrieval effectiveness. Any new features should expand the information captured to increase the rank of a math expression that is structurally similar to the query formula. To this end, we examine the following new features: terminal symbols, compound symbols, expanding symbol pairs to include location, edge pairs, and long paths.

4.1 Terminal Symbols

Tangent-3 includes End-of-Line (EOL) symbols, represented in symbol pairs using the form $(s, !0, \rightarrow)$, where s is a symbol on a node that has no outedge labelled $\rightarrow$, and !0 denotes end-of-line. When using Dice's coefficient and MSS for ranking, wild card expansion for EOL symbols is large and retrieval time increases without improving effectiveness [28]. Rather than dropping EOL values that are especially useful for finding small expressions, Tangent-3 limits the inclusion of EOL symbol pairs to small expressions only: those that have a SLT with tree height at most two.

To test the value of preserving information about end of lines under BM25 scoring, we propose *terminal symbols*, features that represent EOL using the form $(s, !0)$, where s is the label on a leaf in the SLT (i.e., the node containing s has no outgoing edges). This feature can be applied to all math expressions but is expected to remain particularly helpful when searching for small math expressions. Table 3 depicts the terminal symbols for the SLT in Figure 1.[3]

4.2 Compound Symbols

Symbol pairs reflect information about paths in a SLT, but they do not capture information about branching in the tree. For example, the lists of symbol pairs using window size one for $x_1^2 - x$ and $x_1 - x^2$ are identical. Additional information could be captured by including a *compound symbol* of the form $(s, [e_1, e_2, ..., e_k])$ for every symbol labelling a node with more than one outedge, those edges being labelled $e_1, e_2, ..., e_k$. This distinguishes the two formulae by including the feature $(x, [\nearrow, \rightarrow, \searrow])$ for the former and the feature $(x, [\rightarrow, \searrow])$ for the latter. Table 4 provides the list of compound symbols for the SLT in Figure 1.

4.3 Symbol Pairs with Location

When searching for a query formula, a document formula that matches exactly should have a higher score than one in which

[3]This is slightly different from the convention adopted by Tangent-3, where the criterion for introducing a symbol pair for EOL is that there is no "next" symbol, but there might be outedges for exponents or subscripts, for example. Thus Tangent-3 additionally includes a symbol pair representing the feature that the node labelled $V!x$ has no outgoing edge labelled $\rightarrow$.

Table 4: Compound Symbols for $y_i^j = 1 + x^2$

S	$[E]$
$V!y$	$[\nearrow, \rightarrow, \searrow]$

Table 5: Edge Pairs for $y_i^j = 1 + x^2$

E_1	E_2	s
$\rightarrow$	$\rightarrow$	$=$
$\rightarrow$	$\rightarrow$	$N!1$
$\rightarrow$	$\rightarrow$	$+$
$\rightarrow$	$\nearrow$	$V!x$

a subexpression provides an exact match to the query. However, BM25 rewards features that match without penalizing features that occur in the document but not in the query (other than a small penalty for increased overall document length).

Symbol pairs include the path between symbols, but the path from the root of the SLT to the first symbol, its *location*, provides additional information that could be used to reward document formulae that match a query formula exactly. For example, for the formula $x^2 + y^2$, the location of the symbol pair $(y, 2, \nearrow)$ is $\rightarrow\rightarrow$, the label sequence for the path from the root of the SLT to the node labelled y, and this could be used to increase the score of any document formula that includes y^2 in a similar position.

If matching any symbol pairs were to require that the symbol pair's location also match, however, it would no longer be possible to match a query formula to a document formula's subexpression (other than a subexpression at the start of that formula). Therefore, a feature that includes location should only be considered as supplementary to that feature without location, not as a replacement. This expands the size of the index, but avoids the tradeoff between being too restrictive and not capturing location information.

4.4 Edge Pairs

Symbol pairs capture the relationships between pairs of symbols on each path in an SLT. An alternative is to capture the relationships between pairs of edges along those paths. Thus, similar to a symbol pair, an *edge pair* is a tuple with the following form (e_1, e_2, s) where e_1 and e_2 are edge labels that share a common symbol s. Table 5 shows a list of edge pairs for the SLT in Figure 1.

4.5 Long Paths

Tangent-3 includes a parameter called "window size" that limits the length of the path between symbols when forming symbol pairs. For example, with a window size of one, only pairs of symbols in a parent-child relation can form a pair, eliminating the final 10 symbol pairs shown in Table 2. When using Dice's coefficient and MSS for ranking, limiting the window size to one proved to be most suitable [28]. In the presence of BM25, however, it may be valuable to include features for *long paths* to reflect all symbol pairs along all paths.

For large formulae, extremely long paths may occur between a pair of symbols. A long path contains a lot of information, but

Table 6: Symbol Pairs with Abbreviated Paths for $y_i^j = 1 + x^2$ and a window size of 1

S_1	S_2	R
$V!y$	$N!1$	$\rightarrow\rightarrow$
$=$	$+$	$\rightarrow\rightarrow$
$N!1$	$V!x$	$\rightarrow\rightarrow$
$+$	$N!2$	$\rightarrow\nearrow$
$V!y$	$+$	$\rightarrow\rightarrow$
$=$	$V!x$	$\rightarrow\rightarrow$
$N!1$	$N!2$	$\rightarrow\nearrow$
$V!y$	$V!x$	$\rightarrow\rightarrow$
$=$	$N!2$	$\rightarrow\nearrow$
$V!y$	$N!2$	$\rightarrow\nearrow$

may be too restrictive when matching subexpressions. This is especially problematic in the presence of wild cards in the queries, where large subexpressions in a formula might be matched by the wild card. For example, a query for $1 + ? + x^3$ should match an expression regardless of how many terms appear between 1 and x^3. With unbounded windows, this query includes the feature $(N!1, N!3, \rightarrow\rightarrow\rightarrow\rightarrow\nearrow)$ among others. The query is intended to match $1 + x + x^2 + x^3$, but for that formula the corresponding symbol pair is $(N!1, N!3, \rightarrow\rightarrow\rightarrow\rightarrow\rightarrow\rightarrow\nearrow)$: the detailed information in the path causes an erroneous mismatch.

To accommodate this situation, we set a window size within which paths must be matched exactly, but beyond which we preserve only the prefix and suffix of a path. We explore two variants of this feature type in Section 5.2: The first, *long paths empty*, is to include no path information at all for symbol pairs outside the window (i.e., both the preserved prefix and the preserved suffix of paths outside the window are empty). The second variant, *long paths abbreviated*, is to preserve only the first and last labels of each path outside the window. Table 6 provides the abbreviated symbol pairs for the SLT in Figure 1 when using a window size of one.

5 EVALUATION

We present experiments designed to test the effectiveness of the math features described in Section 4. We then explore how well the text and math components work together in a single system.

5.1 Implementation

Lucene, an open-source library for information retrieval, is used to implement the engine for indexing and searching. This implementation requires pre-processing documents to convert MathML expressions into Tangent's SLTs, from which features are extracted as tuples. We adapted Lucene's standard tokenizer to accept math feature tuples as tokens that are distinguishable from the tokens extracted from running text. Additionally, we adapted Lucene's query parser to recognize math feature tuples.

5.2 Math Features

5.2.1 Math Workload. The main dataset chosen to explore the effectiveness of selected math features is the NTCIR-11 Wikipedia collection, which includes approximately 30,000 Wikipedia articles [22]. This allows for a direct comparison to earlier results reported for Tangent-3 [28], providing a direct measure of the performance of various math features with respect to formula retrieval.

The NTCIR-11 Wikipedia benchmark includes 100 queries, each with a specified target formula that is part of a particular document. For this benchmark, these target formulae are the only ones considered to be "relevant" when judging query performance.

The benchmark's protocol requires that a system returns a ranked list of (*documentId*, *formulaId*) pairs. *Document-centric* results are computed using a list of document identifiers in order of appearance after projecting on *documentId* and removing duplicates. *Formula-centric* results are computed using the ranked list with both document and formula identifiers. Using those ranked lists, recall is measured by mean Recall@10000 (the percentage of queries in which the specified document/formula is included within the top 10000 results), and precision is measured by mean reciprocal rank (MRR, the average of the inverses of the ranks of the specified documents/formulae over all queries, where the reciprocal rank for a list that does not include the specified document/formula is 0).

To prepare the Wikipedia math data for Lucene, we index all formulae as if they were each separate documents, and associate each indexed formula with the identification of its source in the Wikipedia corpus. In fact, two indexes are created: *Index-Emp* includes all the features listed in Section 4 where the long paths are empty, and *Index-Abb* has all the same features but the long paths are abbreviated, as in Table 6.

5.2.2 Feature Effectiveness. Our baseline is to use one feature only when querying: Tangent-3's symbol pairs with window size one and no EOL symbols. A series of experiments then tests the performance of including various combinations of the additional features for querying. Thus, query configurations cover the inclusion or omission of terminal symbols, the inclusion or omission of compound symbols, the inclusion or omission of symbol pairs with locations, the inclusion or omission of edge pairs, and the inclusion or omission of long paths. Queries against the two indexes provide two possibilities for paths outside the window: querying with empty paths and querying with abbreviated path information.

The NTCIR-11 Wikipedia task is run for all configurations, and two metrics (Recall@10000 and MRR) are recorded for each run. For each index containing five features there are 16 configurations that include the feature and 16 configurations that do not. The average difference between a configuration including a feature and excluding it is shown in Table 7.

Using symbol pairs as the only feature has a high recall of 97% at both the document and formula level, leaving only three queries that fail to recall the target formula. Two of the missing target formulae are so small that their feature sets include no symbol pairs at all. As a result, any configuration that includes terminal symbols has 99% recall, which outperforms the best version of Tangent-3 (Table 8). The third missing target formula can be recalled by including both compound symbols and location.

Using symbol pairs alone results in a MRR of 75% for the document-centric metric, which is relatively low when compared to Tangent-3's document-centric MRR of 82%. However, including several of the additional features improves precision, as shown in Table 7. Including edge paths or long abbreviated paths has a negative impact on precision. Including long empty paths has a negative impact on precision for the formula-centric measure and a small positive impact for the document-centric measure. All other features improve performance with respect to precision for both the formula-centric measure and the document-centric measure.

Based on this analysis, we recommend including features to reflect terminal symbols, compound symbols, and locations of symbol pairs. This *recommended configuration* produces a recall of 100%, a precision of 80% for the document-centric measure, and a precision of 77% for the formula-centric measure. On this benchmark, Tangent-L's recommended configuration has better recall than Tangent-3 (by 2%) but slightly worse precision (by around 2%), as seen in Table 8. However, using this set of features with BM25 avoids the need to implement the complex re-ranking algorithm used by Tangent-3, which might make it more suitable for use when extending the task to document retrieval in the presence of both formula and keyword query terms.

5.3 Text and Math

To explore the precision of Tangent-L for both math and text, we tune an additional parameter using the NTCIR-12 MathWiki Task benchmark and then evaluate performance against the NTCIR-12 arXiv Main Task benchmark. The former benchmark includes 30 queries with both math formulae and keywords posed against a collection of 319,689 Wikipedia articles. The latter one includes 29 queries with both math formulae and keywords posed against a collection of 8,201,578 fragments of arXiv articles that mostly include mathematical formulae. Further information about these benchmarks, including how relevance assessments are assigned, is included in the NTCIR-12 MathIR overview [27]. Both collections are indexed with the optimal configuration (terminal symbols, compound symbols, symbols pairs with location) determined by the experiments reported in Section 5.2.

Every math formula used in a query can produce many tuples as features, and by default, BM25 assigns each math feature the same weight as it assigns to each keyword in the query. Thus the presence of many math features effectively causes much more weight to be assigned to math formulae than to keywords. Our first experiments explore how weighting the math and text affects precision of the system. For this purpose, we adopt Lucene's flexible architecture to assign weights to query terms as follows:

$$\mathrm{BM25}^{+}_{\mathrm{w}}(q_t \cup q_m, d) = \mathrm{BM25}^{+}(q_t, d) + \alpha \cdot BM25^{+}(q_m, d)$$

where q_t is the set of keywords in a query, q_m is the set of math feature tuples in that query, and α is a parameter to adjust the relative weight applied to math tuples.

Figure 2 shows the effect of α on the average of four precision measurements (P@5, P@10, P@15, P@20) for both relevant and partially relevant results against the NTCIR-12 MathWiki Task benchmark. As α increases in the range $0.05 \leq \alpha \leq 0.50$, there is a gain in precision, for both relevant and partially relevant assessments. The maximum for relevant assessments occurs at $\alpha = 0.47$ and maximum for partially relevant assessments occurs at $\alpha = 0.41$. For larger values of α, there is a gradual decline.

Table 7: Comparison of Tangent-L feature configurations measured against NTCIR-11 Wikipedia Retrieval benchmark. avg(Δ *Recall*@10000) is the average percentage change of recall for targets; avg(Δ *MRR*) is the average percentage change of mrr.

	avg(Δ *Recall*@10000)		*avg*(Δ *MRR*)	
Feature	Documents	Formula	Documents	Formula
Terminal Symbols	2.00	2.00	1.23	0.92
Compound Symbols	0.00	0.50	0.25	0.28
Symbol Pairs with Location	1.00	0.50	0.84	0.48
Edge Pairs	0.00	0.00	-0.04	-0.14
Long Paths Empty	0.00	0.00	0.01	-0.76

(a) Tangent-L with Index-Emp (all features indexed; long paths empty)

	avg(Δ *Recall*@10000)		*avg*(Δ *MRR*)	
Feature	Documents	Formula	Documents	Formula
Terminal Symbols	2.00	2.00	1.12	0.98
Compound Symbols	0.00	0.50	0.32	0.33
Symbol Pairs with Location	1.00	0.50	0.89	1.21
Edge Pairs	0.00	0.00	0.00	0.00
Long Paths Abbreviated	0.00	0.00	-0.33	-0.32

(b) Tangent-L with Index-Abb (all features indexed; long paths abbreviated)

Table 8: Comparison of MIR systems measured against NTCIR-11 Wikipedia Retrieval benchmark.

	Recall@10000 (%)		Mean Reciprocal Rank (%)	
System*	Documents	Formula	Documents	Formula
TUW Vienna	97	93	80	82
NII Japan	97	94	74	72
Tangent-3 Core (w=1, EOL)	98	98	81	77
Tangent-3 Re-rank (w=1, EOL)	98	98	82	79
Tangent-L (symbol pairs)	97	97	75	72
Tangent-L (recommended features)	100	100	80	77

* Previous results from Zanibbi et al. [28].

Table 9: Comparison of MIR systems measured against NTCIR-12 arXiv Main Task benchmark.

	Relevant				**Partially Relevant**			
System Run*	P@5	P@10	P@15	P@20	P@5	P@10	P@15	P@20
MCAT$_{af-lr}$	0.2552	0.2379	0.2092	0.1845	0.5586	0.5379	0.5080	0.4810
MCAT$_{af-lr-u}$	0.2621	**0.2448**	0.2092	0.1845	0.5586	0.5483	0.5218	0.4931
MCAT$_{af-nw-u}$	**0.2897**	**0.2448**	**0.2276**	**0.2000**	**0.5793**	**0.5552**	**0.5402**	**0.5121**
MCAT$_{nd-lr-u}$	0.2345	0.2000	0.1793	0.1621	0.4828	0.4793	0.4828	0.4759
MIaS$_{cm-r-10}$	0.1241	0.1345	0.1218	0.1069	0.3931	0.3690	0.3425	0.3224
MIaS$_{pm-r-10}$	0.1172	0.0828	0.0897	0.0810	0.3379	0.2690	0.2943	0.2672
MIaS$_{pcm-l-10}$	0.1310	0.1000	0.0782	0.0845	0.3862	0.3483	0.2989	0.2793
MIaS$_{pm-l-10}$	0.0690	0.0793	0.0736	0.0793	0.2897	0.2897	0.2644	0.2586
SMSG5$_{tfidf}$	0.0690	0.0931	0.0874	0.0810	0.3517	0.3724	0.3586	0.3397
Tangent-3$_1$	0.2552	0.2000	0.1586	0.1345	0.5517	0.4517	0.3908	0.3483
Tangent-3$_2$	0.2621	0.2000	0.1632	0.1362	0.5448	0.4552	0.3908	0.3517
Tangent-3$_3$	0.1862	0.1552	0.1425	0.1259	0.5448	0.4931	0.4575	0.4414
Tangent-3$_4$	0.1862	0.1586	0.1425	0.1276	0.5310	0.5034	0.4644	0.4448
WikiMir	0.2207	0.1828	0.1609	0.1379	0.5379	0.4931	0.4437	0.4172
Tangent-L$_{\alpha=0.47}$	0.2828	0.2138	0.1655	0.1414	0.3862	0.3552	0.3172	0.2793
Tangent-L$_{\alpha=0.41}$	0.2759	0.2138	0.1678	0.1431	0.3931	0.3655	0.3241	0.2931

* Previous results from Zanibbi et al. [27].

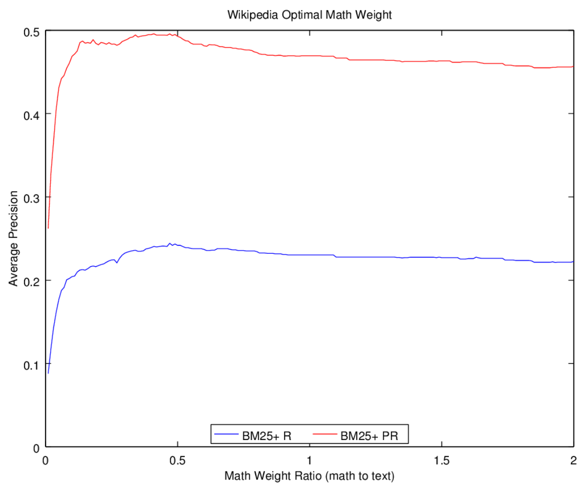


Figure 2: The effect of α on average precision for NTCIR-12 MathWiki Task using relevant (R) and partially relevant (PR) judgements.

Table 10: Breakdown of judgements made on documents returned at various levels of precision for Tangent-L$_{\alpha=0.47}$.

Precision Level	Relevant	Partially Relevant	Not Relevant	Not Judged	Total
$P@5$	40	15	24	66	145
$P@10$	62	41	40	147	290
$P@15$	72	66	54	243	435
$P@20$	82	80	67	351	580
$P@1000$	246	408	451	27895	29000
$P@\infty$	413	876	2962		

As a final experiment, Tangent-L using both $\alpha = 0.47$ and $\alpha = 0.41$ is evaluated against the full NTCIR-12 arXiv Main Task benchmark using the trec_eval standard tool.[4] As shown in Table 9, Tangent-L has better precision than Tangent-3 for relevant judgements at any level of precision. However, Tangent-L has worse precision than Tangent-3 for all of the partially relevant judgements.

Examining these results more closely, we find that Tangent-L returns many results that have no relevance judgements assigned, and all unjudged documents are assumed to be "not relevant" when calculating precision. Table 10 shows the number of documents judged and not judged across all queries for various levels of precision. There is some likelihood that at least some of the unjudged documents would be deemed to be relevant or partially relevant were they to be judged by an unbiased assessor.

One measure that was designed for comparing retrieval systems' effectiveness against a collection with incomplete judgements is bpref [5], which is a commonly used TREC statistic and included in the trec_eval tool. Bpref is calculated as:

$$\text{bpref} = \frac{1}{|R|}\sum_{r\in R}\left(1 - \frac{|\{n \in N \text{ and } n \text{ ranked higher than } r\}|}{\min(|R|, |N|)}\right)$$

where R is the set of documents judged to be relevant and N is the set of documents judged to be not relevant.

Table 11 shows the bpref score for each run of the various systems, including Tangent-L, against the NTCIR-12 arXiv Main Task benchmark. Both runs of Tangent-L have considerably larger scores than all runs of the other systems, except for the top-performing system configuration among the original participants, MCAT$_{af-nw-u}$, with which Tangent-L is competitive. This demonstrates that, after adjusting for unjudged documents, Tangent-L supports our hypothesis that an integrated text-and-math approach to retrieval is superior to merging ranked lists from independent text and math retrieval systems and deserves further investigation.

Table 11: Bpref scores for NTCIR-12 arXiv Main Task.

System Run	Relevant	Partially Relevant
MCAT$_{af-lr}$	0.2232	0.3669
MCAT$_{af-lr-u}$	0.2281	0.3699
MCAT$_{af-nw-u}$	**0.2906**	0.4025
MCAT$_{nd-lr-u}$	0.2157	0.3629
MIaS$_{cm-r-10}$	0.1150	0.2622
MIaS$_{pm-r-10}$	0.0927	0.2347
MIaS$_{pcm-l-10}$	0.0986	0.2276
MIaS$_{pm-l-10}$	0.0640	0.2070
SMSG5$_{tfidf}$	0.0695	0.2434
Tangent-3$_1$	0.1768	0.2967
Tangent-3$_2$	0.1811	0.2980
Tangent-3$_3$	0.1433	0.3500
Tangent-3$_4$	0.1447	0.3522
WikiMir	0.1717	0.2474
Tangent-L$_{\alpha=0.47}$	0.2752	0.4087
Tangent-L$_{\alpha=0.41}$	0.2728	**0.4199**

5.4 Efficiency

Efficient execution was not a priority in building our search engine, but it is important to show that efficiency is not problematic with our approach. This section shows that there is no major loss with respect to index size or retrieval time for Tangent-L compared to Tangent-3.

For these tests, we use a Ubuntu Linux 14.04.5 LTS machine with AMD FX-8370E Eight-Core Processor (3.3GHz) and 16 GB RAM. All times reported reflect the use of a single thread for processing.

5.4.1 Indexing. Four indexes are created for the formula search experiments performed in Section 5.2. Table 12 shows the size of the four indexes, with the largest being 575.5 MB and the smallest being 32.3 MB. These are comparable to the largest and smallest index sizes for Tangent-3. For the NTCIR-12 arXiv Main Task, the index used by Tangent-L with the recommended features requires 5.74 GB compared to Tangent-3's index size of 8.15 GB for math formulae only [8] and is much smaller MCAT's index size of 428.54 GB [14]. Such index sizes are acceptable with today's hardware.

[4]https://github.com/usnistgov/trec_eval

Table 12: Comparing Tangent-3 and Tangent-L index sizes for NTCIR-11 Wikipedia collection.

Configuration*	Index Size (MB)
Tangent-3 (No-EOL,w=1)	64.2
Tangent-3 (EOL,w=1)	73.7
Tangent-3 (EOL,w=unbounded)	503.1
Tangent-L (symbol pairs)	32.3
Tangent-L (recommended features)	124.2
Tangent-L (all features; long paths empty)	261.3
Tangent-L (all features; long paths abbreviated)	575.5

* Tangent-3 sizes reported by Zanibbi et al. [28].

5.4.2 Retrieval Times. The time used by Tangent-3 to run the 100 NTCIR-11 Wikipedia queries against the Wikipedia corpus and return the top 100 formulae has been previously reported [28] for execution on a Ubuntu Linux 14.04 server with 24 Intel Xeon processors (2.93GHz) and 96GB of RAM: the core engine requires 8.5 seconds and the total time with re-ranking using MSS requires 106 seconds. For comparison, we repeat this experiment with Tangent-L. Execution requires less than 3.0 seconds to load and convert the queries into math tuples and open the Lucene index. Averaging the run times to execute each query 10 times results in a mean query execution time of 0.034 seconds. This gives an average execution time of under 6.4 seconds in total to run 100 formula queries, each producing 100 results. Thus the efficiency of Tangent-L is acceptable and comparable to just using the core engine in Tangent-3; the high cost for re-ranking has been eliminated. Additionally, the average time over 10 runs for each of the 29 queries from the arXiv Main Task requires 0.69 seconds (mean time per query), with the fastest query requiring 0.02 seconds on average and the slowest query requiring 2.56 seconds. This drastically outperforms the reported times for Tangent-3 (mean=27.54, minimum=2.77, maximum=178.51) [8].

6 CONCLUSIONS

After performing extensive experiments, we find that using a traditional search engine and ranking appropriately chosen math features with *BM*25 performs comparably to the state-of-the-art math retrieval system. The recommended features to use are symbol pairs with window size one, terminal symbols, compound symbols, and symbol pairs with location. With this configuration, Tangent-L has a better bpref than all other system runs on the NTCIR-12 arXiv Main Task for partially relevant hits.

Future Work. Our experiments use weighted $\mathrm{BM25}_{\mathrm{w}}^{+}$ with two different weights (0.47 and 0.41) for math query features, but other variations of BM25 [26] may also be worth exploring. Alternative ranking functions that incorporate proximity between terms [24] may also prove to be useful. Techniques that combine SLTs and Operator Trees into a single retrieval model [9] also remain to be explored.

ACKNOWLEDGMENTS

We gratefully acknowledge financial assistance received from the University of Waterloo (partially as a David R. Cheriton Graduate Scholarship) and from NSERC, the Natural Sciences and Engineering Research Council of Canada. We also thank Japan's National Institute of Informatics for allowing us to use the NTCIR document data and test collections for our experiments.

REFERENCES

[1] Akiko Aizawa, Michael Kohlhase, and Iadh Ounis. 2013. NTCIR-10 Math Pilot Task Overview. In *NTCIR-10*. 654–661.
[2] Akiko Aizawa, Michael Kohlhase, Iadh Ounis, and Moritz Schubotz. 2014. NTCIR-11 Math-2 Task Overview. In *NTCIR-11*. 88–98.
[3] Javed A Aslam and Mark Montague. 2001. Models for metasearch. In *SIGIR 2001*. 276–284.
[4] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet Allocation. *Journal of Machine Learning Research* 3 (2003), 993–1022.
[5] Chris Buckley and Ellen M Voorhees. 2004. Retrieval evaluation with incomplete information. In *SIGIR 2004*. 25–32.
[6] David Carlisle, Patrick Ion, and Robert Miner. 2014. *Mathematical Markup Language (MathML) Version 3.0 2nd Edition*. W3C Recommendation. http://www.w3.org/TR/2014/REC-MathML3-20140410/.
[7] Davide Cervone. 2012. MathJax: a Platform for Mathematics on the Web. *Notices of the AMS* 59, 2 (2012), 312–316.
[8] Kenny Davila, Richard Zanibbi, Andrew Kane, and Frank Wm Tompa. 2016. Tangent-3 at the NTCIR-12 MathIR Task. In *NTCIR-12*. 338–345.
[9] Kenny Davila Castellanos. 2017. *Symbolic and Visual Retrieval of Mathematical Notation using Formula Graph Symbol Pair Matching and Structural Alignment*. Ph.D. Dissertation. Rochester Institute of Technology.
[10] Yoav Freund, Raj Iyer, Robert E Schapire, and Yoram Singer. 2003. An Efficient Boosting Algorithm for Combining Preferences. *Journal of Machine Learning Research* 4, Nov (2003), 933–969.
[11] Liangcai Gao, Ke Yuan, Yuehan Wang, Zhuoren Jiang, and Zhi Tang. 2016. The Math Retrieval System of ICST for NTCIR-12 MathIR Task. In *NTCIR-12*. 318–322.
[12] Deyan Ginev and Bruce R Miller. 2013. LaTeXML 2012—a year of LaTeXML. In *CICM 2013*. 335–338.
[13] Ferruccio Guidi and Claudio Sacerdoti Coen. 2016. A Survey on Retrieval of Mathematical Knowledge. *Mathematics in Computer Science* 10, 4 (2016), 409–427.
[14] Giovanni Yoko Kristianto, Goran Topic, and Akiko Aizawa. 2016. MCAT Math Retrieval System for NTCIR-12 MathIR Task. In *NTCIR-12*. 323–330.
[15] Ray R Larson, Chloe Reynolds, and Fredric C Gey. 2013. The Abject Failure of Keyword IR for Mathematics Search: Berkeley at NTCIR-10 Math. In *NTCIR-10*. 662–666.
[16] Quoc V. Le and Tomas Mikolov. 2014. Distributed Representations of Sentences and Documents. In *ICML 2014*. 1188–1196.
[17] Daniel W Lozier, Bruce R Miller, and Bonita V Saunders. 1999. Design of a Digital Mathematical Library for Science, Technology and Education. In *ADL'99*. 118–128.
[18] Yuanhua Lv and ChengXiang Zhai. 2011. Lower-bounding Term Frequency Normalization. In *CIKM'11*. 7–16.
[19] Stephen Robertson and Hugo Zaragoza. 2009. The Probabilistic Relevance Framework: BM25 and Beyond. *Foundations and Trends in Information Retrieval* 3, 4 (2009), 333–389.
[20] Michal Růžička, Petr Sojka, and Martin Líška. 2014. Math Indexer and Searcher under the Hood: History and Development of a Winning Strategy. In *NTCIR-11*. 127–134.
[21] Michal Růžička, Petr Sojka, and Martin Líška. 2016. Math Indexer and Searcher under the Hood: Fine-tuning Query Expansion and Unification Strategies. In *NTCIR-12*. 331–337.
[22] Moritz Schubotz, Abdou Youssef, Volker Markl, and Howard S Cohl. 2015. Challenges of Mathematical Information Retrievalin the NTCIR-12 Math Wikipedia Task. In *SIGIR 2015*. 951–954.
[23] Petr Sojka and Martin Líška. 2011. The Art of Mathematics Retrieval. In *DocEng 2011*. 57–60.
[24] Tao Tao and ChengXiang Zhai. 2007. An Exploration of Proximity Measures in Information Retrieval. In *SIGIR 2007*. 295–302.
[25] Abhinav Thanda, Ankit Agarwal, Kushal Singla, Aditya Prakash, and Abhishek Gupta. 2016. A Document Retrieval System for Math Queries. In *NTCIR-12*. 346–353.
[26] Andrew Trotman, Antti Puurula, and Blake Burgess. 2014. Improvements to BM25 and Language Models Examined. In *ADCS'14*. 58–65.
[27] Richard Zanibbi, Akiko Aizawa, Michael Kohlhase, Iadh Ounis, Goran Topic, and Kenny Davila. 2016. NTCIR-12 MathIR Task Overview. In *NTCIR-12*. 299–308.
[28] Richard Zanibbi, Kenny Davila, Andrew Kane, and Frank Wm Tompa. 2016. Multi-stage Math Formula Search: Using Appearance-based Similarity Metrics at Scale. In *SIGIR 2016*. 145–154.