Learning to Rank for Mathematical Formula Retrieval

Behrooz Mansouri
Rochester Institute of Technology
Rochester, NY, USA
bm3302@rit.edu

Richard Zanibbi
Rochester Institute of Technology
Rochester, NY, USA
rxzvcs@rit.edu

Douglas W. Oard
University of Maryland
College Park, MD, USA
oard@umd.edu

ABSTRACT

In Mathematical Information Retrieval (MIR), formulae can be used in a query to match other similar formulae in documents. However, due to the structural complexity of formulae, specialized processing is needed for formula matching. Formulae may be represented by their appearance in Symbol Layout Trees (SLTs) or by their syntax in Operator Trees (OPTs). Previous approaches for formula retrieval used one or both of these representations and used unification to improve search results for inexact matches (e.g., allowing different variable names to match). On these representations, models for matching full expressions (trees), subexpressions, and paths have been used. Recently embedding models were used to represent formulae as vectors. In this paper, the effectiveness of retrieval models and formula representations are studied to identify their relative strengths and weaknesses. Then, a learning to rank model is proposed, using SVM-rank over similarity scores from different formula retrieval models as features. Experiments on the ARQMath formula retrieval task results show that the proposed learning to rank model is effective, producing new state-of-the-art results.

CCS CONCEPTS

• **Information systems** → **Learning to rank**.

KEYWORDS

Formula Retrieval, Math Information Retrieval, Learning to Rank

ACM Reference Format:
Behrooz Mansouri, Richard Zanibbi, and Douglas W. Oard. 2021. Learning to Rank for Mathematical Formula Retrieval. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '21), July 11–15, 2021, Virtual Event, Canada.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3404835.3462956

1 INTRODUCTION

In Mathematical Information Retrieval (MIR), user information needs concern mathematical concepts. A common way to express mathematical information needs is mathematical formulae [17]. Some existing search engines such as SearchOnMath [21] and MathDeck [18] provide facilities for users to insert a formula query. Most commonly formulae are represented using LaTeX, but some accept formula images and handwriting as input (e.g., [18]). Formula search must accommodate queries containing single symbols up to complex structures that include operators, numbers, and variables organized on multiple writing lines (e.g., for superscripts) and in grid structures (e.g., matrices). As a result, traditional text retrieval systems are ineffective for formula search due to the structural complexity of formulae [31]. Instead of using LaTeX strings to represent formulae, current systems use the underlying trees represented by LaTeX. Symbol layout trees (SLTs) and operator trees (OPTs) are common tree-based representations, where the first captures formulae appearance and the second captures formula syntax. Figure 1 shows these representations for the Pythagorean theorem, $a^2 + b^2 = c^2$.

Formula retrieval has previously been studied in a number of shared tasks, including tasks at NTCIR 10, 11, and 12 [2, 3, 30], and ARQMath at CLEF 2020 [33]. In both the NTCIR-12 and ARQMath tasks, the most effective systems used both SLT and OPT formula tree representations, and supported inexact matching through unification of variables, identifiers, numbers, and operators. Also, previous works have shown that combining representations and models can improve the retrieval results. The best performing system in the ARQMath formula retrieval task (Tangent-S [5]) matches paths in both representations to obtain initial retrieval results. Another participating system at ARQMath (Tangent-CFTED) [15] achieved similar results by first selecting candidates with a formula embedding model, and then re-ranking candidates by tree edit distance to obtain a matching score for the full formula tree. The results from participating system [20], combines TF-IDF with unsupervised word embeddings to produce representations for math formulae which provided better results than each system in isolation.

Three main approaches have been applied for formula retrieval in previous models that yielded stronger retrieval results: sub-tree matching [5, 10, 34], full-tree matching [9, 15] and embedding models [6, 11, 16, 25]. Models can make use of both representations [5, 10, 15], or use only SLTs [9] or OPTs [34]. Despite the existence of these models, how different models and representations should be combined has not been studied before. In this paper, we aim to answer the following research questions:

- **(RQ1)**: What is the effectiveness of each type of retrieval model (sub-tree, full-tree and embedding)?
- **(RQ2)**: Which formula representation(s) work best with each type of retrieval model?
- **(RQ3)**: Can evidence from multiple retrieval models and/or formula representations be combined to further enhance retrieval results?

To answer our research questions, we used these systems for each matching model: (1) **sub-tree matching**: Tangent-S scores for initial ranking via tuple matching, maximum sub-tree similarity (a measure of structural similarity), and matching symbols by both exact values and types, (2) **full-tree matching**: unweighted, and weighted tree edit distances between formula trees, (3) **embedding**

SIGIR '21, July 11–15, 2021, Virtual Event, Canada

ACM ISBN 978-1-4503-8037-9/21/07…$15.00
https://doi.org/10.1145/3404835.3462956

models: Tangent-CFT formula embedding model. After analyzing the behavior of each retrieval model and formula tree representation, we introduce a Learning to Rank (LtR) model for mathematical formulae, using SVM-rank [8]. We use SVM-rank because of its speed, the effectiveness of SVMs on small samples (ARQMath has only 29 training queries), and the stability of SVM training, making replication easier.

In our evaluation, we first rank all formulae in the ARQMath collection to observe how our LtR model behaves, and obtain new state-of-the-art results for the task. We then apply LtR to the submitted runs from the ARQMath formula retrieval task and obtain higher effectiveness for all runs. All data and source code used for our paper is publicly available.[1]

In the next section, we review related work on formula retrieval; we then present the retrieval models and formula representations used in our LtR models in Section 3. We then introduce our ranking models in Section 4, followed by our experimental results in Section 5. Finally, we conclude and identify future directions in Section 6.

2 RELATED WORK

In this section, we first review how mathematical formulae are represented, and discuss existing collections for math information retrieval. Finally, we review existing formula retrieval models.

2.1 Formula Representations

Mathematical formulae are represented for retrieval by their mathematical (operational) syntax in operator tree (OPT) encodings such as Content MathML, or by their appearance in symbol layout tree (SLT) encodings such as LaTeX [31]. Figure 1 shows the OPT and SLT representation for the Pythagorean formula: $a^2 + b^2 = c^2$.

In this work, we use the OPT and SLT representations from the Tangent family of retrieval models [5, 32]. Nodes in OPTs and SLTs represent symbols and explicit aggregates (e.g., function arguments) in the form: (**Type**!value). More precisely, nodes can be numbers (**N!**n), identifiers such as variable names (**V!**v), text fragments, such as 'lim' and 'such that' (**T!**t), fractions (**F!**), radicals (**R!**), explicitly specified white-space (**W!**) and finally, matrices, tabular structures, and parenthesized expressions (**M!**f rxc); with f showing fence characters such as parentheses, r the number of rows, and c the number of columns. In the SLTs, operators do not have a type attribute, but for OPTs commutative and non-commutative operators have type (**U!**) and (**O!**), respectively. Commutative operators ignore the order of their arguments (e.g., equality) while the position/order of non-commutative operators is important (e.g., subtraction).

Edge labels in OPTs indicate argument position. For commutative operators edges are unlabelled. For non-commutative operators, edge labels are indexed from 0. In SLTs, edge labels give the spatial arrangement of symbols on writing lines. There are nine types of edge labels representing the spatial relationships between symbols (nodes): **next** ('n') indicates that a symbol appears to the right on the same writing line; **within** ('w') references the argument in a radical or the first element appearing in row-major order in a matrix/grid of type M!, **element** ('e') references the next element appearing in row-major order inside a structure represented by M!, **above** ('a') indicates a new writing line in the superscript position, **below** ('b') references a new writing line in the subscript position, **pre-above** ('SUP') indicates a prescripted superscript, **pre-below** ('SUB') indicates a prescripted subscript, **Under** ('U') indicates a writing line appearing below a node (e.g., a fraction's denominator), and finally **Over** indicates a writing line above an node (e.g., a fraction's numerator). For instance, edge label a in Figure 1 shows '2' is superscripted *above* 'b,' and edge label n shows that '=' is located *next* to 'b'.

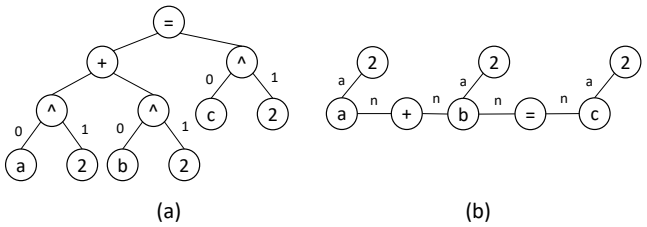

Figure 1: $a^2 + b^2 = c^2$ represented as an (a) Operator Tree (OPT) and a (b) Symbol Layout Tree (SLT). OPTs represent operations, with edge labels ordering arguments: arguments for commutative operators all have edges labeled '0' (unlabeled in (a)). SLTs represent appearance by the placement of symbols on writing lines, with edge labels indicating spatial relationships (e.g., superscript (a), next on writing line (n)).

2.2 Formula Retrieval Test Collections

Formula retrieval involves retrieving a set of relevant formulae for a given formula query. The task was first visited at NTCIR 10-12 [2, 3, 30], with a variety of settings, including the use of wildcards and constraints on symbols or sub-expressions [1] (e.g., requiring matched argument symbols to be variables or constants).

ARQMath 2020 uses Mathematics Stack Exchange posts for the collection to be searched [33]. Formula Retrieval was one task in ARQMath 2020, with some similarities in design to the NTCIR-12 Wikipedia Formula Browsing task. Some key differences were in how queries were defined, and how evaluation was performed. In particular, for evaluation ARQMath uses the *visually distinct* formulae in a ranked result set, rather than all (possibly identical) formula instances, as done for NTCIR-12. The primary evaluation measure was nDCG′ whereas for NTCIR-12 Precision@k measures were used to compare systems. The NTCIR-12 formula retrieval test collection also had a smaller number of queries, with 20 fully specified formula queries, plus 20 variants of these queries with sub-expressions replaced by wildcards. NTCIR-11 also had a formula retrieval task, with 100 queries, but in that case, systems searched only for exact matches and were evaluated using reciprocal rank. The ARQMath formula retrieval dataset contains 45 test queries, along with an additional 29 training queries. All queries are complete formulas, without wildcards.

Another important difference between ARQMath and the NTCIR tasks is how relevance is defined. For ARQMath relevance is scored using four ratings: high, medium, low, and non-relevant. Here are the definitions of each level of relevance:

(0) **Non-relevant:** Not expected to be useful.
(1) **Low:** There is some chance of finding something useful.

[1] https://github.com/BehroozMansouri/LtRMathIR

(2) **Medium:** Useful but not as good as the original formula would be.
(3) **High:** Just as good as finding an exact match to the query formula would be.

ARQMath annotators were shown the question where a formula query appeared, along with the question or answer post in which retrieved formulae appeared. Therefore, formulae were annotated in context, whereas in NTCIR-12 the annotators were shown the formulae without surrounding text. In this way, the ARQMath evaluation reflects both lexical and mathematical context in its annotations. We note here though, that the NTCIR-12 formula retrieval test collection may be particularly well suited for evaluating tasks where the lexical context may be absent, such as formula query auto-completion.

2.3 Math Formula Retrieval

Previous approaches for formula retrieval can be categorized as *text-based*, *tree-based* or *embedding* models. In text-based models, formulae are converted to a string such as LaTeX and then the same search method is used for both formulae and text. For instance, systems such as MIaS [28] or WikiMirs [7] used Term Frequency and Inverse Document Frequency (TF-IDF) for indexing XHTML documents. In another work, Lin et al. [13] adapt TF-IDF retrieval for SLTs by using vectors of subexpressions, considering canonicalization to simplify expressions and to identify commutative operators and equivalences. Text-based approaches lose the hierarchical nature of formulae and may fail to characterize formula structure well.

In tree-based models, formulae are represented directly as trees, often with sub-trees to support partial matching. Kristianto et al. [10] proposed the MCAT system that encodes path and sibling information from Presentation MathML (SLT) and Content MathML (OPT) representations, where paths act as the retrieval units. They also made use of a hashing-based formula structure encoding scheme, and text information at three levels of granularity. Approach0 [34], by contrast, retrieved formulae using only paths from operator trees generated by parsing LaTeX with a relatively small expression grammar. Candidates were scored based on up to three best-matching sub-trees. Like MCAT, Tangent-S [5] combined retrieval over both SLTs and OPTs. Candidates were first retrieved and scored using tuples representing relative paths between pairs of symbols. The top-k candidates were then aligned with the query to produce formula similarity scores (the Maximum Sub-tree Similarity, MSS). SLT results and OPT results were next combined via linear regression over alignment measures from each representation to produce final similarity scores.

While sub-tree matching is one common approach (e.g., using path-based retrieval models), another uses matching of complete formula trees. For example, the SimSearch system [9] calculated the Tree Edit Distance (TED) between SLT representations, with the distance normalized by tree sizes. Edit operation costs were adjusted in some conditions (e.g., a node being a leaf or internal node). While full tree-based approaches yielded strong results for near-exact matches, they were less effective for inexact matches.

To capture the full context of a tree while avoiding the issues associated with ranking by TED, we can use embedding models that convert trees to vectors. Early research on formula embedding was carried out by Thanda et al. [29] where a variant of the doc2vec algorithm, the distributed bag of words (PV-DBOW) [12] was introduced. They used binary expression trees and assigned each formula a real-valued vector such that formulae with similar structures are close to each other in the vector space. Gao et al. [6] introduced embeddings for both symbols (symbol2vec) and formulae (formula2vec). Symbol2vec was based on a Continuous Bags-of-Words (CBOW) architecture using negative sampling, while formula2vec used a distributed memory model of paragraph vectors [12]. The method proposed by Krstovski and Blei [11] generates embeddings for both words and equations, with a larger context window size for equations than words. They also proposed equation unit embedding, treating equations as sentences where the words are symbols, variables, and operators, each referred to as a "unit."

Pfahler and Morik [25] proposed embedding formula images using graph convolutional neural networks and improved their representations using two self-supervised tasks (contextual similarity, and a masking task) to preserve information about the raw input symbols. Using Bit Position Information Table (BPIT) [23] and Term-Document matrix [22], the proposed system in [4], transforms formulae into variable-sized weight vectors, with each weight indicating the occurrence count of a particular formula entity and its position in a BPIT. In another work, [20] combines TF-IDF and word embeddings to produce embedding representations of math documents and math formulae.

Tangent-CFT [16] is an n-gram embedding model for OPT and SLT representations. This model currently yields state-of-the-art results for *partial* matching on the NTCIR-12 formula task. However, the embedding loses some structural information, making it less effective for nearly exact matches, in direct contrast to TED-based methods. For instance, for the query $O(mn \log m)$, Tangent-CFT gives a higher rank to formulae such as $O(mn)$ compared to formulae such as $O(nk \log k)$, as the model is based on n-grams and the first formula shares more common characters with the query.

Each of these different approaches to formula retrieval has benefits and drawbacks. In this work, we study how to combine the strengths of these approaches using a learning to rank model. To the best of our knowledge, using formula matching scores as features for learning to rank has not been attempted before.

3 RETRIEVAL MODELS AND FORMULA REPRESENTATIONS

In this section, we first discuss the formula retrieval models whose rank scores are used as features in our learning to rank model. We then define the formula representations used to construct features.

3.1 Similarity Features

As mentioned above, there are three main approaches to formula retrieval: sub-tree matching, full-tree matching, and formula embedding, each with its benefits and limitations. We first explain our use of these models, and then demonstrate their benefits.

Sub-tree matching features. To do sub-tree matching, we used the initial candidate selection method from the Tangent-S system. Using the tree representation of a formula, a list of tuples is created

Table 1: Tuples for $a^2 + b^2 = c^2$. Tuples represent paths between symbols as *(parent, child, path, path-from-root[PFR])*. V!, N!, O!, and U! are node types (operators like '+' have no SLT node type), 'eob' end-of-baseline, and '-' an empty path.

SLT Tuples				OPT Tuples			
(Parent,	Child,	Path,	PFR)	(Parent,	Child,	Path,	PFR)
(**V!** a,	**N!** 2,	a,	-)	(**U!** eq,	**U!** plus,	0,	-)
(**N!2**,	eob,	n,	a)	(**U!** plus,	**O!** SUP,	0,	0)
(**V!** a,	+,	n,	-)	(**O!** SUP,	**V!**a,	0,	00)
⋮				⋮			

from a tree using a depth-first traversal. Each tuple has 4 elements: (parent, child, path, path-from-root [PFR]). Parent and child are the node values, path is showing the edge labels seen when traversing from parent to child, and PFR shows edges labels visited when moving from the root node to the parent node. Considering the example from Figure 1, the first few tuples created for SLT and OPT representations are shown in Table 1. The first SLT tuple indicates that a node with type *Variable* and value 'a' is connected to a node with type *Number* and value '2', and when moving from the first node to the second the edge label 'a' is visited. As the first node is the root of the tree, the path-from-root is empty. The second tuple is showing that the node with type *Number* and value '2' has no children and we have end of baseline (eob). The default edge label for the eob is 'n'. The path from the root of the tree to the '2' is a single edge labeled 'a'. Finally, the third SLT tuple shows the root of the tree connected to a node with the value '+' along with an edge label 'n'. After the tuples are generated, the harmonic mean of the ratio of query tuples matched (recall) and the ratio of candidate tuples matching the query (precision) is used for ranking.

Another feature that we use to compare sub-trees is the Maximum Sub-tree Similarity (MSS) score introduced in the Tangent-3 system [32]. MSS is computed from the largest connected match between a query and candidate formula obtained using a greedy algorithm, evaluating pairwise alignments between trees using unified node values. Another two re-ranking features used in this system are query node matching after alignment, both with unification and without.

Full-tree matching features. Another approach computes similarity using full formula trees, making the matching criterion more strict. We use tree-edit distance as a full-tree matching feature. Tree Edit Distance (TED) is defined as the minimum-cost sequence of node (or possibly node and edge) edit operations to transform one tree into another. In this study, three kinds of edit operations are considered:

- **Insertion:** If a node is inserted between a parent and its children, the new node becomes the parent of the child node.
- **Deletion:** If a deleted node has children, they will be connected to the deleted node's parent.
- **Substitution (i.e., relabeling):** a single operation combining a deletion and an insertion.

In [15], ignoring the edge labels yielded better retrieval results and so we ignore them here.

Once the tree-edit distance is calculated, the similarity score can be calculated as [9]:

$$sim(T_1, T_2) = 1 - \frac{TED(T_1, T_2)}{|T_1| + |T_2|} \quad (1)$$

where $|T_i|$ is the number of nodes in tree T_i. With this formula, the tree size is taken into consideration as a normalization factor, also mapping the similarity score to the range of 0 to 1.

An alternative to using editing operations with the same cost is the weighted TED, where operation costs may differ. Many related mathematical formulae share the same structure, but have different variable or function names. Therefore, we assume that even related formulae may require multiple substitutions to match a query, and so using a lower weight for substitutions could produce better retrieval results. While Kamali and Tompa [9] designed such weights by hand, we have learned weights using the 29 ARQMath train queries. Using values from 0 to 1 with a step size of 0.05, the weights are learned for each editing operation averaged over 29 training queries. We calculate TED using the "apted" python library implementation of the APTED TED algorithm [24].

Embedding features. While the features described above use exact tree representations, embedding features provide dense vector representations for each formula that can provide better matching scores for partially relevant formulae [16]. For the embedding features, we trained the n-gram embedding model used in the Tangent-CFT system.

We make a minor improvement to the original Tangent-CFT model in this work. In the original version, three n-gram embedding models are trained separately on SLT, OPT, and SLT Type trees (a unified SLT representation). The final representation was the sum of the 3 embedding vectors. In this study, we instead produce similarity scores separately for each of the three representations, and combine these using a learning-to-rank model.

Other tree and symbol features (OTS). This is an additional set of simple features to compare formulae directly. These include:

(1) Ratios of matching variables, numbers, and operators between the query and candidate formula.
(2) Difference between the number of nodes in query and candidate in SLT and OPT.
(3) Difference between OPT depth.
(4) Difference between SLT variation from the baseline, defined as the maximum number of edge labels visited when moving from a node to the root of the tree which do not have 'next' as their edge label. For instance, for the example in Figure 1, this value is one.

3.2 Formula Representations

Both SLT and OPT representations can capture different aspects of similarities between formulae. On one hand, SLTs can capture the appearance of formulae, but the OPT representation captures the operator syntax. For example, for query $A^2 + B^2 = C^2 + D^2$, formula $A^2 + B^2 + C^2 + D^2$ has a very similar appearance to the query, and will have a higher similarity using an SLT representation than an OPT representation (where the '=' is at the root).

Unification makes matching less strict and can improve search results by finding inexact, but relevant matches. We provide examples from the ARQMath dataset on why we need all these different

representations. For all the examples, we use the tuple matching strategy. To illustrate why we need both SLTs/OPTs with (Full) and without (Type) symbol values, consider the formula

$$\mathrm{lcm}(n_1, n_2) = \frac{n_1 n_2}{\gcd(n_1, n_2)}.$$

By ignoring the variable names, relevant formulae such as

$$\mathrm{lcm}(a, b) = \frac{ab}{\gcd(a, b)}.$$

can be easily found. While in this example unification helped, for the formula

$$\sin(x), \sin(2x), \sin(3x), ..., \sin(nx)$$

using the unified type representation, non-relevant formulae like

$$f(x), f(2x), f(3x), \ldots$$

obtain a high similarity score. Therefore, having similarity scores for both type (unified) and full tree representations is important.

As illustrated by these examples, SLT and OPT representations, in both their full and their unification-based type variants, can help find similar formulae. Our experiments use all four representations.

4 LEARNING TO RANK FOR FORMULA RETRIEVAL

To train our learning to rank model, we used SVM-rank [8] with a linear kernel because of its effectiveness with small samples (ARQMath has only 29 training queries), the stability of SVM training making replication easier, and the resulting combination can be computed relatively quickly (e.g., using a map-reduce framework). Because we use a linear kernel SVM, the weights of the features can also be examined.

We used SVM-rank trained on the 29 queries with relevance judgments for 3,909 formula instances that are provided in ARQMath. The penalty for misclassification during training (C) was 0.01, and the tolerance for termination criterion (epsilon) was 0.001.

The following feature sets were used in training (names in parentheses are used to refer to that feature set):

- 4 Tuple matching scores (Tuple)
- 2 Maximum Sub-tree Similarity (MSS)
- 4 Node Matching scores (Node Matching)
- 4 Unweighted tree edit distance scores (UTED)
- 4 Weighted tree edit distance scores (WTED)
- 4 Cosine similarity from Tangent-CFT model (CFT)
- 7 Other tree and symbol features (OTS)

For feature sets other than the 'OTS' and 'MSS', one feature is calculated for each of the four tree representations (SLT, OPT, SLT TYPE, OPT TYPE). As 'MSS' features use unification, the full and type representations would have the same scores, so we have only two features in that set. All feature values are MinMax normalized to be in the range [0,1]. For the sub-tree features, we are using the same Tangent-S parameters as were used for ARQMath 2020.

The learned weights for deletion, substitution and insertion operations in our weighted TED are respectively: SLT (0.43, 0.20, 0.41), OPT (0.31, 0.21, 0.29), SLT TYPE (0.37, 0.29, 0.33), and OPT TYPE (0.34, 0.25, 0.32).

Table 2: ARQMath Formula Retrieval Task test queries. An example for each complexity level is provided.

Count	Complexity	Example
21	Low	$i = \sqrt{-1}$
16	Medium	$\sum \frac{1}{n^{2+\cos n}}$
8	High	$\binom{s}{s} + \binom{s+1}{s} + ... + \binom{n}{s} = \binom{n+1}{s+1}$

5 EXPERIMENTS

In this section, we first study the ranking results on the whole ARQMath collection using each feature. Then, we study if a single model can benefit from having different formula representations. Finally, we explore how the learning to rank model works over the entire ARQMath formula collection and then when re-ranking previously reported runs. All our results are calculated with the same evaluation protocols as described in [33], but we used the visual ids provided for ARQMath2 [14] as more formulas have been successfully converted.

To compare the effectiveness of the systems, we consider 3 measures: Precision@5 (P′@5), Mean Average Precision (MAP′), and nDCG′@5. For P′@5 and MAP′ we binarize the graded relevance judgments by treating only high and medium as relevant (i.e., 'HM' binarization). Following the ARQMath evaluation protocols, all these measures are calculated on visually distinct formulae (after removal of duplicates, based on the duplicate list provided by ARQMath). Post-hoc use of a test collection raises the risk that unjudged formulae may be retrieved; we mitigate this effect by reporting the 'prime' version of each metric, as defined by Sakai [26], in which unjudged formulae are removed from all ranked listed before computing the evaluation measure. This is the same process that was used for the official ARQMath scoring.[2] All evaluation measures in this paper are computed using trec_eval.[3]

The ARQMath Formula Retrieval test collection has 29 train and 45 test queries, all of which were constructed using the same process after run submission and are now available for post-hoc use [33]. Each query has a complexity tag which indicated the organizers' a priori estimate of query difficulty. This annotation was initially intended to help balance the degree of difficulty of the queries in the test collection and it was not available to participating systems at the time; it is now available for post-hoc use to support analysis or system training. For instance, the complexity level of 'Low' indicates that the organizers were expecting this to be a comparatively easy query for the state of the art systems. Table 2 illustrates and provides statistics for these three complexity levels.

For baselines, we considered two systems [5, 15] from the 2020 ARQMath Formula Retrieval task that had the highest nDCG′ and P′@5 values. To provide a further basis for comparison, we also show results for the most effective run from each of two other

[2]In ARQMath, the pooling process ensured that no participating run had unjudged documents in the top 5 so the P@5 reported by ARQMath is numerically equal to P′@5 for those runs. The same may not be true for post-hoc runs, so P′@5 is a proper measure for post-hoc runs. We use the same process explained in the Errata at https://www.cs.rit.edu/~dprl/ARQMath.

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

Table 3: ARQMath results, average over 45 Test topics. Mean (μ) and standard deviation (σ) are shown for each measure.

System	$\mu(\sigma)$ P′@5	MAP′	nDCG′@5
Tangent-S [5]	0.51 (0.30)	**0.45 (0.22)**	**0.58 (0.27)**
Tangent-CFTED [15]	**0.53 (0.36)**	0.39 (0.27)	0.56 (0.26)
SCM [20]	0.09 (0.22)	0.06 (0.14)	0.10 (0.22)
FormulaEmbedding [4]	0.04 (0.09)	0.02 (0.03)	0.06 (0.13)

Table 4: ARQMath results for formula representations (columns) used in different retrieval models (rows).

System	Precision@5′ $\mu(\sigma)$ SLT	OPT	SLT Type	OPT Type
Sub-tree				
Tuple	0.56 (0.29)	0.51 (0.32)	0.45 (0.33)	0.39 (0.27)
MSS	0.47 (0.28)	0.49 (0.33)	0.47 (0.28)	**0.49 (0.33)**
Node Matching	0.56 (0.32)	0.52 (0.33)	0.36 (0.34)	0.40 (0.33)
Full Tree				
UTED	**0.61 (0.32)**	0.56 (0.32)	**0.52 (0.33)**	0.43 (0.27)
WTED	0.59 (0.29)	0.55 (0.32)	0.50 (0.32)	0.43 (0.29)
Embedding				
CFT	0.58 (0.32)	**0.59 (0.30)**	0.45 (0.35)	0.44 (0.31)

participating teams, SCM [20] and Formula embedding [4]. Table 3 shows the effectiveness measures for these systems.

5.1 Models with One Representation

To answer **RQ1**, we first look at retrieval results for each feature. The whole ARQMath collection is ranked with each feature, and unjudged formulae are removed. This is essentially re-ranking only the annotated formulae. This analysis provides insight into the limitations and strengths of each model and representation.

Table 4 shows P′@5 for each feature. (due to space limitations, nDCG′@5 and MAP′ are not shown, but they yield similar comparisons). The main findings are: (1) ranking with the Full SLT representation using the UTED model yields the highest P′@5, (2) full representations work better than type representations, and (3) SLTs provide better retrieval results than OPTs.

Detailed analysis. Table 4 shows that unification cannot provide better retrieval results than full representation. Considering the SLT representation, moving from the UTED model to the MSS model, P′@5 drops from 0.61 to 0.47. There are formulae for which the symbols should not be ignored, and unification leads to higher ranks for non-relevant formulae. For the query $\lim_{u\to\infty} \frac{u^m}{e^u} = 0$, a relevant formula such as $\lim_{t\to\infty} \frac{t^k}{e^t} = 0$ and a non-relevant formula like $\lim_{y\to\infty} \frac{y^{\alpha-1}}{e^y} = 0$ get similar matching scores. Another example is the CFT model for which there is 15% drop in P′@5 moving from full to type OPT representation. Table 5 shows the top-5 results returned for query $\int_{x=0}^{\infty} \frac{\sin(x)}{x}$ by OPT full and type representations using the CFT model. As this illustrates, unification cannot distinguish the non-relevant symbols in retrieved formulae.

Table 5: CFT Top-5 assessed hits for query $\int_{x=0}^{\infty} \frac{\sin(x)}{x}$ using OPT and OPT Type representations.

Rank	OPT	Relevance	OPT Type	Relevance
1	$\sum_{x=1}^{\infty} \frac{\sin(x)}{x}$	Low	$\sum_{n=0}^{\infty} \frac{\sin(n)}{n}$	Low
2	$\int_{0}^{\infty} \frac{\sin x}{x}$	Medium	$\sum_{n=1}^{\infty} \frac{\sin(n)}{n}$	Low
3	$\int_{x=0}^{\infty} \frac{\sin x}{x} dx$	Medium	$\sum_{n=0}^{\infty} \frac{\cos(n)}{n}$	Non-relevant
4	$\int_{-\infty}^{\infty} \frac{\sin x}{x}$	High	$\sum_{x=1}^{\infty} \frac{\sin(x)}{x}$	Low
5	$\int_{-\infty}^{\infty} \frac{\sin(x)}{x}$	High	$\sum_{n=1}^{\infty} \frac{\sin(n)}{n!}$	Non-relevant

SLT representations can provide a slightly better result than OPT representation, except for the CFT model. However, the retrieval results confirm our assumption in Section 3.2 that both representations are useful for retrieval. For example, for the query $A^2 + B^2 = C^2 + D^2$ (from examples in Section 3.2), with the tuple-based model, non-relevant formulae such as $D = A^2 + B^2 + C^2$ get higher ranks with OPT. For this query (with tuple-based features) P′@5 for OPT is 0 and for SLT P′@5 is 0.6. In contrast to this, for queries such as $(1 + i\sqrt{3})^{1/2}$ (from the examples in Section 3.2) the P′@5 for SLT and OPT are 0 and 0.4, respectively. The SLT gave higher ranks to low-relevance formulae such as $(1 + i\sqrt{3})/2$ that share a common appearance.

Finally, we focus on OTS (Other Tree and Symbol) features, some of which cannot be computed for some representations. These features yield weak similarity scores compared to tree or embedding matching. Three of these features compare the symbols between the query and the candidate using their operators, variables, and numbers. The other 4 features compare the general tree structure of two formulae. Ranking with these four features had less effectiveness with the average P′@5 of 0.02 for each feature. However, the matching features could achieve P′@5 of 0.32 for operator, 0.27 for number, and 0.26 variable matching. For example, for the query $(x + y)^k \geq x^k + y^k$, the P′@5 for operator matching is 1. With operator matching, formulae with nearly the same operators and different variable names such as $(x + y)^a \geq x^a + y^a$ get high similarity scores. For our next analysis, we just keep the three OTS matching features (matching variables, numbers, and operators), ignoring the other four OTS features.

5.2 Models with Multiple Representations

After looking at different representations, we focus on **RQ2** by training SVM-rank for each model using all representations: SLT, SLT Type, OPT, and OPT type. Our goal is to study if one model can provide better retrieval results using multiple representations.

Overall, the experiment results show that some models have strengths that other models do not. Table 6 shows the P′@5, MAP′, and nDCG′@5 for each of the models. Using an ANOVA followed by a Tukey HSD (Honest Significance Difference) test, OTS features yielded results that were significantly worse than all other models except MSS ($p < 0.01$), but the differences between other pairs of

Table 6: ARQMath results (45 topics) for SVM-rank over rank scores from retrieval models applied to four formula representations (SLT, SLT Type, OPT, and OPT Type).

	$\mu(\sigma)$		
Feature Set	P′@5	MAP′	nDCG′@5
Sub-tree			
Tuple	0.56 (0.30)	0.52 (0.24)	0.64 (0.25)
MSS	0.50 (0.32)	0.53 (0.26)	0.54 (0.30)
Node Matching	**0.60 (0.29)**	**0.57 (0.24)**	**0.67 (0.25)**
Full Tree			
UTED	0.55 (0.33)	0.54 (0.26)	0.62 (0.29)
WTED	0.57 (0.33)	0.56 (0.26)	0.63 (0.30)
Embedding / Other			
CFT	0.57 (0.33)	0.52 (0.24)	0.64 (0.26)
OTS	0.36 (0.32)	0.34 (0.26)	0.39 (0.33)

systems are not statistically significant. Comparing Tables 4 and 6, we see that MSS and Node Matching benefit from LtR over all four representations. For other models, using a single representation would be just as good or better, but only if the best representation were known a priori.

Formula Complexity vs. Retrieval Model. There are signs that each set of features may do better for different formula complexities. We recalculated the P′@5 values for each set of features per complexity category (see Table 7). For formulae with low complexity, ranking with CFT works best, for formulae with medium complexity the full-tree features are more effective, and for formulae with high complexity, ranking with sub-tree features does better.

We initially assumed that ranking with CFT features would provide better results for simpler formulae, where approximately similar formulae can be informative. However, we did not expect tuple features to outperform TED features for complex formulae, where it seemed whole-tree matching would be best. There are two possible reasons for this. First, most formulae identified as complex are larger than many other formulae. Therefore, finding partial matches can provide better retrieval results. Second, what has been annotated as medium complexity in the ARQMath collection might be misleading; a formula that seems complex from a human perspective might not be complex for a machine. For the 16 queries that have medium complexity, on average, there are 23 formulae considered as relevant with an average of 13 formulae having high or medium relevance. These numbers increase for the 8 formulae with high complexity, with an average of 30 relevant formulae per query and 19 formulae that have high or medium relevance.

Feature Weights. One useful property that SVM-rank provides is the weights for each feature. Table 8 shows the weights for each of the matching scores with different representations. Overall, in all the matching features, the full representations have higher weights compared to type representations, with OPT type being the less informative feature. This aligns with the finding in [16] where the OPT type representation was ignored.

Missing Features. There are some ARQMath queries that perform poorly for all of the models. For instance, for

$$Empty(x) \iff \nexists y(y \in x)$$

Table 7: Results from Table 6 broken down by query complexity (see Table 2) for different feature sets.

	P′@5 $\mu(\sigma)$		
Feature Set	Low (21)	Medium (16)	High (8)
Sub-tree			
Tuple	0.66 (0.32)	0.48 (0.28)	**0.55 (0.26)**
MSS	0.54 (0.34)	0.48 (0.29)	0.45 (0.37)
Node Matching	0.71 (0.26)	0.49 (0.32)	**0.55 (0.23)**
Full Tree			
UTED	0.58 (0.35)	**0.56 (0.32)**	0.45 (0.32)
WTED	0.66 (0.36)	**0.55 (0.29)**	0.40 (0.26)
Embedding / Other			
CFT	**0.72 (0.28)**	0.49 (0.29)	0.35 (0.35)
OTS	0.51 (0.33)	0.24 (0.26)	0.22 (0.27)

Table 8: SVM-rank weights for retrieval models using four formula tree representations (OPT, SLT, OPT Type, and SLT Type). Weight comparisons should be made within rows.

	OPT		SLT	
Model	Full	Type	Full	Type
Sub-tree				
Tuple	2.78	0.05	3.19	1.53
Node Matching	1.74	-0.09	2.32	1.17
Full Tree				
UTED	5.89	4.62	5.78	5.03
WTED	3.74	2.28	5.23	3.45
Embedding				
CFT	1.94	-0.98	4.98	3.40

only tuple matching features had a P′@5 greater than 0. There are seven relevant formulae in the collection for that query, but some would require symbolic manipulation based on mathematical rules to find. For instance, the formula '$\vdash \exists x \forall y (y \notin x)$' is highly relevant, but none of the feature sets will be able to capture the similarity of these two formulae. Therefore, canonicalization of mathematical formulae may improve results [19, 27].

There are also formulae in ARQMath where the proper interpretation is shaped by associated query text, which we have not used. Other formulae that may be similar to the query in terms of SLT and/or OPT may come from a different context, in which symbols refer to different operations or values (e.g., sets vs. real numbers), and so are annotated as non-relevant. For instance, the query $\frac{df}{dx} = f(x+1)$, appeared in a question where someone was asking for solving "differential equations". There are formulae such $\frac{dy}{dx} = f(x)$ in the collection that are annotated as non-relevant.

Additional Analysis. Ranking with SVM-rank trained on CFT features, we see that for the query $p_n = \frac{1}{2}p_{n-1}$, P′@5 is 0.8. By contrast, P′@5 is 0 for both full-tree features, and 0.2 for each set of sub-tree features. In this query, p_n represents a probability, but for example with TED features formulae such as $b_n = \frac{1}{2}b_{n-1}$ are found because a substituting p for b yields a match. However, we can see by looking at the post in which it appears that b_n in that retrieved formula refers to an element in a sequence and not a probability,

Table 9: Ranking with Other Tree and Symbol features (OTS): Top-5 hits for query: $a^3 + b^3 + c^3 - 3abc$.

Rank	Formula	Relevance Score
1	$a^3 + b^3 + c^3 - 3abc$	High
2	$a^3 + c^3 = 3abc - b^3$	Medium
3	$a^3 + b^3 + c^3 = 3abc$	Medium
4	$a^3 + b^3 + c^3 - abc$	High
5	$a^3 + b^3 + c^3 - 6abc$	High

and therefore this formula was correctly annotated as non-relevant. CFT features, by contrast, are capable of finding similar formulae by prioritizing formulae that share common symbols.

Sub-tree matching features can also do well. There are large and complex formula queries such as:

$$\iint_V f(x,y)dx\ dy = \iint_Q f(\Phi(u,v)) \left| \frac{\partial \Phi}{\partial u} \times \frac{\partial \Phi}{\partial v} \right|$$

for which sub-tree matching can be more effective. The P′@5 for ranking with tuple features is 0.8, whereas for CFT P′@5 is 0 and 0.2 for UTED and WTED, respectively. The first retrieved formula by CFT features is:

$$\iint_D f(x,y)dxdy = \iint_{D^*} \| \frac{\delta(x,y)}{\delta(u,v)} \| f(x(u,v), y(u,v)dudv$$

which is annotated as non-relevant. As can be seen, the same property of CFT features, giving high rank to formulae with common symbols, provides a better ranking for the previous formula but not this one.

Sub-tree matching cannot always provide relevant results. For instance, for the query:

$$\neg P \rightarrow A_1 \rightarrow \ ... \ \rightarrow A_n \rightarrow P$$

with the sub-tree matching feature tuple, a non-relevant formula

$$\neg A \rightarrow A \rightarrow B$$

is the first retrieved formula. For some of the queries, partial matching provides less relevant results. TED features can overcome this issue by looking at the whole tree. For the above query, ranking with tuple features results in P′@5 of 0.2 whereas with TED features P′@5 is 0.8.

As Table 6 shows, WTED features can slightly improve P′@5. For query $(1 + i\sqrt{3})^{1/2}$, ranking with UTED features has P′@5 of 0; for WTED features P′@5 is 0.4. As operation costs learned from training queries tend to give lower weight to substitutions, formula $(1 + \sqrt{3}i)^{1/2}$ is in the top-5 results for WTED features, whereas for UTED features in which all operations have the same weight a non-relevant formula such as $(1 + i\sqrt{3})^8$ gets a higher rank.

Ranking with other tree and symbol (OTS) features can also find relevant formulae for some of the queries. For four low-complexity queries, using this set of features P′@5 was 1. Table 9 shows the top-5 results returned when ranking with other-tree features for the query $a^3 + b^3 + c^3 - 3abc$. As these results indicate, for some queries, simple features such as the number of matching operators, variable names, and numbers can provide effective results.

Table 10: ARQMath results for SVM-rank using different feature subsets.

	$\mu(\sigma)$		
Features from Table 8	P′@5	MAP′	nDCG′@5
All features	0.61 (0.31)	0.57 (0.24)	0.68 (0.26)
Top-6 Weights	0.61 (0.30)	**0.58 (0.24)**	0.67 (0.25)
All features: no OTS	**0.64 (0.27)**	**0.58 (0.23)**	**0.69 (0.21)**

5.3 Multiple Models with Multiple Representations

Finally, we look at whether our learning to rank model can further improve retrieval results by using multiple retrieval models with multiple formula representations (**RQ3**). As noted before, our OTS features are less informative than the similarity scores from the retrieval models, so although we do train a LtR model with all features and retrieval models, we also train a LtR model without the OTS features. Because running a large number of retrieval models concurrently can be costly, we also create a third LtR model using only the retrieval models with the highest learned linear weights (in this case, with absolute weight ≥ 0.3) to see whether a smaller model can perform similarly.

Table 10 shows results for these three conditions. Removing OTS features improves all measures, and produces the highest P′@5, MAP′, and nDCG′@5 values yet reported for this test collection. Moreover, the LtR model using fewer ranking models as features yields results that are nearly as good at roughly half the computational cost. Using a one-way ANOVA followed by Tukey HSD tests on P′@5 values, all three of these models are significantly different from the ARQMath baseline systems ($p < 0.01$) in Table 3, but differences between models in Table 10 are not statistically significant.

For some queries, a single ranking model performs better than combining multiple ranking models, but such cases are rare in ARQMath. For instance, for query $\neg P \rightarrow A_1 \rightarrow \ ... \ \rightarrow A_n \rightarrow P$, sub-tree matching finds some formulae matching small parts of the expression that were assessed as not relevant. As a result, P′@5 decreases from 0.8 using UTED alone to 0.4 when using all features. However, more commonly the SVM-rank model using all ranking models performed better for complex formulae, where sub-tree matching often proved beneficial.

Feature Computation Times. For each retrieval model, we calculated the average time needed to compute the similarity between a query and a single candidate formula in milliseconds, averaging over formulae in all representations (SLT, OPT, SLT Type, OPT Type). The times shown in Figure 2 were computed using a single thread on an Intel(R) Xeon(R) CPU E5-2667 v4 @ 3.20GHz with 528GB RAM. Note that MSS and node matching features are computed together in one pass by Tangent-S. As can be seen, calculating similarity features for a large collection is impractical, particularly for tree edit distances. Given this, we next look at how our learning to rank model can be applied for re-ranking rather than ranking the whole collection.

Re-Ranking ARQMath Runs. To further validate our learning-to-rank model, we applied our best model (*All features: no OTS*) to

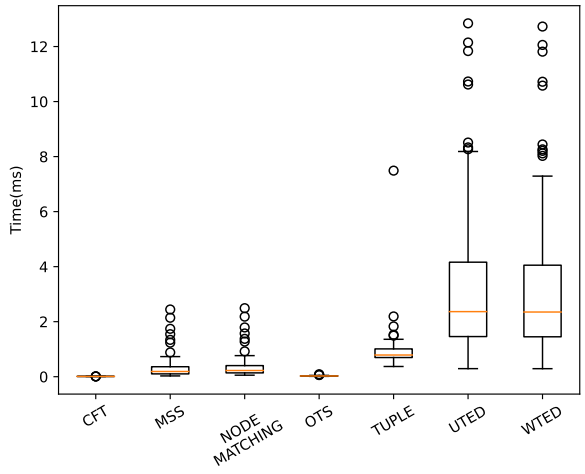

Figure 2: Per-query feature calculation time for a single formula tree in milliseconds, aggregated over 45 queries using OPT, SLT, OPT Type, and SLT Type representations.

Table 11: Re-ranking ARQMath Task 2 runs using the best SVM-rank model. All mean scores improve in every metric.

System	P′@5	$\mu(\sigma)$ MAP′	nDCG′@5
Original Runs			
Tangent-S	0.51 (0.30)	**0.45 (0.22)**	**0.58 (0.27)**
Tangent-CFTED	**0.53 (0.36)**	0.39 (0.27)	0.56 (0.26)
SCM	0.09 (0.22)	0.06 (0.14)	0.10 (0.22)
FormulaEmbedding	0.04 (0.09)	0.02 (0.03)	0.06 (0.13)
Re-ranked Runs			
Tangent-S	**0.62 (0.30)**	**0.52 (0.22)**	**0.69 (0.23)**
Tangent-CFTED	0.58 (0.32)	0.42 (0.27)	0.65 (0.26)
SCM	0.18 (0.29)	0.10 (0.16)	0.21 (0.20)
FormulaEmbedding	0.15 (0.21)	0.05 (0.11)	0.22 (0.25)

the best runs from participating teams in the ARQMath formula retrieval task. Each original run contained up to 1000 formulae per query. Table 11 compares submitted runs against our re-ranked results. Using the same system settings, on average, the time to re-rank the top-1000 results for a query is 3.10 seconds (time-averaged over 45 queries). The SVM-rank could be run using a map-reduce architecture, with each retrieval model executed in parallel. With the parallel execution, the retrieval time drops to an average of 1.53 seconds per query with tree-edit distance being the most time-consuming model.

The average value for all effectiveness measures improves after re-ranking. T-tests on both P'@5 and $nDCG'$@5 scores between each original and re-ranked result show a significant difference, except for Tangent-CFTED ($p < 0.01$). Providing examples on the results, Tangent-S uses sub-tree features for ranking, shown earlier to produce better results for complex formulas (see Table 7). With re-ranking P′@5 for low-complexity queries improves for this system. For example, P'@5 for query $(\mathscr{M}_{2\times2}(\mathbb{Q}), \times)$ increases from 0 to 0.8, and non-relevant formulae such as $T : \mathcal{M}_{2\times2}(\mathbb{R}) \rightarrow \mathcal{M}_{2\times3}(\mathbb{R})$ are pushed down in the results after re-ranking. The re-ranked Tangent-CFTED results also improved, by incorporating features other than full-tree matching; for the query $\int_0^\infty \frac{\sin x}{x^a}$, after re-ranking the P′@5 score for the query increases from 0.4 to 0.8.

6 CONCLUSION

Prior methods in which hand-engineered combinations across two or more representations (e.g., Symbol Layout Trees and Operator Trees, with or without unification) have pointed the way to the gains in ranking quality that can be achieved by combining evidence. In this paper, we have shown that combining evidence from multiple similarities computed on the same representation(s) provides complementary evidence that can yield further improvements. Moreover, we have shown that with relevance judgments for a modest number of training queries that it is possible to learn to combine that evidence in ways that yield a new state of the art for the formula retrieval task.

Our work opens several new directions that we plan to pursue. First, we can generate additional features that could lead to further improvements. As one simple example, we might measure the potential of using query complexity features of the type that are available in the ARQMath test collection. If that proves productive, we might then draw on the literature from query performance prediction to automatically generate informative features for characterizing query difficulty. We are also interested in exploring the use of graph and visual formula embeddings [25] for similar reasons. A second line of future work would be to explore alternative learning to rank frameworks, including techniques such as LambdaMART that are designed for robust results with limited training, or—for some settings—data-hungry neural techniques.

Unlike the formula retrieval task, the ARQMath community question answering answer retrieval task includes both text and math. The learning to rank framework that we have demonstrated in this paper would clearly be applicable to that task as well, with the presence of text (and the potential presence of multiple formulae) offering even greater scope for feature design. This is the third clear line of future work that we plan to explore.

What makes all of this possible is the existence of a sufficiently large test collection for the task. There is a synergy between system building and collection building; better systems can better target the effort invested in creating relevance judgments, which in turn can yield better training data for building better systems. We are still early in that process, but the first step along that path is to place formula retrieval on a firm foundation as a learning to rank task, as we have sought to do in this paper.

ACKNOWLEDGMENTS

This material is based upon work supported by the Alfred P. Sloan Foundation under Grant No. G-2017-9827 and the National Science Foundation (USA) under Grant No. IIS-1717997.

REFERENCES

[1] Akiko Aizawa and Michael Kohlhase. 2021. Mathematical Information Retrieval. In *Evaluating Information Retrieval and Access Tasks*. Springer, Singapore, 169–185.

[2] Akiko Aizawa, Michael Kohlhase, and Iadh Ounis. 2013. NTCIR-10 Math Pilot Task Overview. In *Proceedings of the 10th NTCIR Conference on Evaluation of Information Access Technologies, NTCIR-10, National Center of Sciences, Tokyo, Japan, June 18-21, 2013*, Noriko Kando and Tsuneaki Kato (Eds.). National Institute of Informatics (NII). http://research.nii.ac.jp/ntcir/workshop/OnlineProceedings10/pdf/NTCIR/OVERVIEW/01-NTCIR10-OV-MATH-AizawaA.pdf

[3] Akiko Aizawa, Michael Kohlhase, Iadh Ounis, and Moritz Schubotz. 2014. NTCIR-11 Math-2 Task Overview. In *In Proceedings of the 11th NTCIR Conference*.

[4] Pankaj Dadure, Partha Pakray, and Sivaji Bandyopadhyay. 2020. An Analysis of Variable-Size Vector Based Approach for Formula Searching. (2020).

[5] Kenny Davila and Richard Zanibbi. 2017. Layout and semantics: Combining representations for mathematical formula search. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*.

[6] Liangcai Gao, Zhuoren Jiang, Yue Yin, Ke Yuan, Zuoyu Yan, and Zhi Tang. 2017. Preliminary Exploration of Formula Embedding for Mathematical Information Retrieval: can mathematical formulae be embedded like a natural language?

[7] Xuan Hu, Liangcai Gao, Xiaoyan Lin, Zhi Tang, Xiaofan Lin, and Josef B Baker. 2013. WikiMirs: a mathematical information retrieval system for wikipedia. In *Proceedings of the 13th ACM/IEEE-CS joint conference on Digital libraries*. 11–20.

[8] Thorsten Joachims. 2006. Training linear SVMs in linear time. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. 217–226.

[9] Shahab Kamali and Frank Wm Tompa. 2013. Retrieving documents with mathematical content. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*. ACM, 353–362.

[10] Giovanni Yoko Kristianto, Goran Topic, and Akiko Aizawa. 2016. MCAT Math Retrieval System for NTCIR-12 MathIR Task. In *NTCIR*.

[11] Kriste Krstovski and David M Blei. 2018. Equation Embeddings.

[12] Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *International Conference on Machine Learning*.

[13] Xiaoyan Lin, Liangcai Gao, Xuan Hu, Zhi Tang, Yingnan Xiao, and Xiaozhong Liu. 2014. A mathematics retrieval system for formulae in layout presentations. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*. 697–706.

[14] Behrooz Mansouri, Anurag Agarwal, Douglas Oard, and Richard Zanibbi. 2021. Advancing Math-Aware Search: The ARQMath-2 Lab at CLEF 2021. In *European Conference on Information Retrieval*. Springer.

[15] Behrooz Mansouri, Douglas W Oard, and Richard Zanibbi. 2020. DPRL Systems in the CLEF 2020 ARQMath Lab. In *Working Notes of CLEF 2020-Conference and Labs of the Evaluation Forum*.

[16] Behrooz Mansouri, Shaurya Rohatgi, Douglas W Oard, Jian Wu, C Lee Giles, and Richard Zanibbi. 2019. Tangent-CFT: An embedding model for mathematical formulas. In *Proceedings of the 2019 ACM SIGIR International Conference on Theory of Information Retrieval*. ACM, 11–18.

[17] Behrooz Mansouri, Richard Zanibbi, and Douglas W Oard. 2019. Characterizing Searches for Mathematical Concepts. In *2019 ACM/IEEE Joint Conference on Digital Libraries (JCDL)*. IEEE, 57–66.

[18] Gavin Nishizawa, Jennifer Liu, Yancarlos Diaz, Abishai Dmello, Wei Zhong, and Richard Zanibbi. 2020. MathSeer: A Math-Aware Search Interface with Intuitive Formula Editing, Reuse, and Lookup. In *European Conference on Information Retrieval*. Springer, 470–475.

[19] Immanuel Normann and Michael Kohlhase. 2007. Extended formula normalization for ε-retrieval and sharing of mathematical knowledge. In *Towards Mechanized Mathematical Assistants*. Springer, 356–370.

[20] Vít Novotnỳ, Petr Sojka, Michal Štefánik, and Dávid Lupták. 2020. Three is better than one. In *CEUR Workshop Proceedings. Thessaloniki, Greece*.

[21] Ricardo M Oliveira, Flavio B Gonzaga, Valmir C Barbosa, and Geraldo B Xexéo. 2017. A distributed system for SearchOnMath based on the Microsoft BizSpark program. *arXiv preprint arXiv:1711.04189* (2017).

[22] Amarnath Pathak, Partha Pakray, and Alexander Gelbukh. 2018. A formula embedding approach to math information retrieval. *Computación y Sistemas* 22, 3 (2018), 819–833.

[23] Amarnath Pathak, Partha Pakray, Sandip Sarkar, Dipankar Das, and Alexander Gelbukh. 2017. Mathirs: Retrieval system for scientific documents. *Computación y Sistemas* 21, 2 (2017), 253–265.

[24] Mateusz Pawlik and Nikolaus Augsten. 2016. Tree edit distance: Robust and memory-efficient. *Information Systems* 56 (2016), 157–173.

[25] Lukas Pfahler and Katharina Morik. 2020. Semantic Search in Millions of Equations. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 135–143.

[26] Tetsuya Sakai. 2007. Alternatives to bpref. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*. 71–78.

[27] Mohammed Shatnawi and Abdou Youssef. 2007. Equivalence detection using parse-tree normalization for math search. In *2007 2nd International Conference on Digital Information Management*, Vol. 2. IEEE, 643–648.

[28] Petr Sojka and Martin Líška. 2011. The art of mathematics retrieval. In *Proceedings of the 11th ACM Symposium on Document Engineering*.

[29] Abhinav Thanda, Ankit Agarwal, Kushal Singla, Aditya Prakash, and Abhishek Gupta. 2016. A Document Retrieval System for Math Queries. In *NTCIR*.

[30] Richard Zanibbi, Akiko Aizawa, Michael Kohlhase, Iadh Ounis, Goran Topic, and Kenny Davila. 2016. NTCIR-12 MathIR Task Overview. In *NTCIR*.

[31] Richard Zanibbi and Dorothea Blostein. 2012. Recognition and retrieval of mathematical expressions. *International Journal on Document Analysis and Recognition (IJDAR)*.

[32] Richard Zanibbi, Kenny Davila, Andrew Kane, and Frank Wm Tompa. 2016. Multi-stage math formula search: Using appearance-based similarity metrics at scale. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*.

[33] Richard Zanibbi, Douglas W Oard, Anurag Agarwal, and Behrooz Mansouri. 2020. Overview of ARQMath 2020: CLEF lab on answer retrieval for questions on math. In *Experimental IR Meets Multilinguality, Multimodality, and Interaction Proceedings of the 11th International Conference of the CLEF Association*. Springer, 169–193.

[34] Wei Zhong and Richard Zanibbi. 2019. Structural Similarity Search for Formulas Using Leaf-Root Paths in Operator Subtrees. In *European Conference on Information Retrieval*.