Layout and Semantics: Combining Representations for Mathematical Formula Search

Kenny Davila
Rochester Institute of Technology
1 Lomb Memorial Drive
Rochester, New York 14623
kxd7282@rit.edu

Richard Zanibbi
Rochester Institute of Technology
1 Lomb Memorial Drive
Rochester, New York 14623
rlaz@cs.rit.edu

ABSTRACT

Math-aware search engines need to support formulae in queries. Mathematical expressions are typically represented as trees defining their operational semantics or visual layout. We propose searching both formula representations using a three-layer model. The first layer selects candidates using spectral matching over tree node pairs. The second layer aligns a query with candidates and computes similarity scores based on structural matching. In the third layer, similarity scores are combined using linear regression. The two representations are combined using retrieval in parallel indices and regression over similarity scores. For NTCIR-12 Wikipedia Formula Browsing task relevance rankings, we see each layer increasing ranking quality and improved results when combining representations as measured by Bpref and nDCG scores.

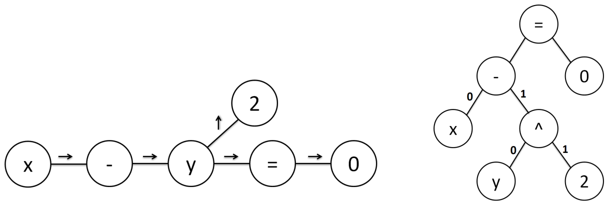
CCS CONCEPTS

•Information systems →Similarity measures; *Rank aggregation; Language models;*

KEYWORDS

Formula Retrieval, Operator Tree, Symbol Layout Tree

ACM Reference format:
Kenny Davila and Richard Zanibbi. 2017. Layout and Semantics: Combining Representations for Mathematical Formula Search. In *Proceedings of SIGIR '17, Shinjuku, Tokyo, Japan, August 07-11, 2017,* 4 pages.
DOI: 10.1145/3077136.3080748



(a) Symbol Layout Tree (b) Operator Tree

Figure 1: Tree representations for $x - y^2 = 0$

1 INTRODUCTION

Math-aware search engines deal with information needs where documents containing particular math expressions are sought after, or where document similarity is defined by text and formulae. An expression can be represented semantically by its operations using an Operator Tree (OPT) or visually by a Symbol Layout Tree (SLT) [16]. Figure 1 shows an SLT and OPT for $x - y^2 = 0$.

Many researchers in *Mathematical Information Retrieval (MIR)* assume OPTs provide better formula retrieval results than SLTs, but each has limitations for retrieval. For SLTs, mathematical notation can change meaning based on context - a symbol may be an operator in one context, and a variable in another, for example. In contrast, well-formed OPTs are mathematically unambiguous. Online, most write math expressions using SLT representations (e.g., LaTeX). SLTs can be converted to OPTs using parsers, but semantics are often undefined or ambiguous, producing errors [3].

Our previous work (Tangent-3[1] [1, 17]) uses a two-stage SLT model for formula retrieval. First, top-k candidates are identified using a bag-of-words model, using symbol pairs in SLTs as 'words.' Then, the top-k candidates are re-ranked after aligning query and candidate SLTs. Candidates are re-ranked using the harmonic mean of symbol and relationship recall (the *Maximum Subtree Similarity*) and two tie-breakers: symbol precision after unification, and symbol recall without unification.

We present an extended model, **Tangent-S**, that works with OPTs and uses a stricter unification model to avoid matching functions to variable names. A third stage is added using a linear combination of the structure similarity scores for re-ranking. Stronger formula retrieval results are obtained by retrieving SLTs and OPTs independently, and then linearly combining their similarity scores. This supports the view of OPTs and SLTs as complementary for formula retrieval.

2 BACKGROUND

Approaches to formula search may be classified by the primitives used for indexing as *text-based*, *tree-based*, and *spectral* [17]. Detailed analysis of existing methods can be found elsewhere [3].

Text-Based Approaches. Formulae are converted to a sequence of tokens using linearization of formula trees. To increase the likelihood of finding matches, some methods use canonicalization to simplify expressions, and to identify commutative operators and equivalences [8, 12]. It is also common to enumerate identifiers to support generalized variable matching and/or unification [2, 12–14].

[1]https://cs.rit.edu/ dprl/Software.html#tangent

Converting math to text allows use of existing optimizations in text search engines, such as ranking by TF-IDF [10], topic modeling, and word embedding [13].

Tree-Based Approaches. These approaches index formulas as complete SLTs or OPTs. Typically, the hierarchical structures in formulae are mapped directly, and organized within tree-based indexing structures [4, 6, 18]. In these approaches, all subexpressions in formulae are indexed to support partial matching, with common subexpressions labeled and shared to reduce index sizes [6].

Spectral Approaches. Here paths in OPTs/SLTs or features extracted from trees are used as retrieval primitives. Simpler primitives allow more partial matches, increasing recall. Path-based methods store sets of paths from the root to internal nodes [5] and leaves [18]. Paths may reflect operator commutativity by inserting symbols [18] or using unordered paths [7]. Some use hashing to encode subtrees [7, 11]. In Tangent-3 SLT symbol pairs along with their relative paths are used to index math expressions [1, 17].

In this work, we extend the Tangent-3 system [17] to retrieve formulae using both SLTs and OPTs, and make additional improvements detailed below.

3 METHODOLOGY

Tangent-3 [17] retrieval model has two stages, one for fast candidate selection using spectral matching, and the second for re-ranking top-k candidates. We add a third stage, using a linear combination of similarity scores computed from the second layer to produce a final re-ranking of the top-k candidates. Rather than use a learning-to-rank technique [9], a simpler model was chosen for better understanding of the relevance of each similarity factor.

3.1 Formula Representation

To define and constrain the behavior of matching and unification algorithms in SLTs and OPTs, we assign each symbol a type. Edges between symbols are labeled by their order in OPTs, or by the visual location of a child symbol with respect to its parent (e.g., for superscript relationships) in SLTs (see Fig. 1).

Common Symbol Types. These include: *Variables, Numbers, Groups* (matrices, vectors, sets, lists, ...), *Functions, Operators, Text, White Space, Query Wildcard* and *Error* (e.g., for parsing errors). Real data often provides strong clues for symbol types, but in some cases symbol type can be hard to infer without context, leading to incorrect symbol types and invalid unifications.

Symbol Layout Trees. This representation is built around writing lines (*baselines*), leading to deep trees with few branches. The children of a node in this representation are assigned to a spatial relationship class (edge label): *Next, Above, Pre-above, Below, Pre-Below, Over, Under, Within,* and *Element.*

Operator Trees. This representation is built around the hierarchy of operators in a formula, resulting in shallow trees with many branches. We distinguish between commutative operators (e.g., '+') and non-commutative operators (e.g., '-'). We ignore the order of children for commutative operators. In Figure 1.b, all edges to children of the equals sign have the same label.

3.2 Pair-based Index Model

A symbol pair is represented by the tuple (A, D, R) where A and D are the ancestor and descendant symbols, and R is the sequence of edge labels in the path from A to D. We use an inverted index, with symbol tuples as keys, and each posting list storing references to formulae containing the tuple. We use independent formula indices for SLTs and OPTs. Two parameters control the symbol tuple generation process: a window size w and an End-of-Baseline (EOB) flag. Window size w defines the maximum path length between an indexed symbol pair. If EOB is true, the system creates dummy pairs between the last symbol on each baseline and *null*, to help with matching small expressions ($depth <= 2$). Details may be found elsewhere [17].

3.3 Formula Retrieval

Applying detailed similarity metrics can be prohibitively expensive. For this reason, a three-layer retrieval process is used.

Layer 1: Initial Candidate Selection. Candidates are selected by matching query symbol tuples in the index. For each candidate, the harmonic mean of precision and recall of matched symbol tuples is used to assign an initial score [17].

Layer 2: Structural Match Scoring. The largest connected match between the query and candidates is obtained using a greedy algorithm, evaluating pairwise alignments between trees. Symbols (nodes) of similar type are unified, and query wildcards are matched to subtrees. Connected matches may contain holes (unmatched intermediate nodes) as long as edge labels between tree structures always match. For SLTs, matching works mostly as defined for Tangent-3 [1], except that we now restrict unification to occur between single character identifiers, or within identifiers with two or more characters, but not between these two groups. This mitigates the issue of spurious unification matches between variables and functions leading to bad candidates being ranked too high.

For OPTs, the order of arguments for commutative operators is ignored to capture matches between equivalent expressions such as $x + y = 0$ and $0 = y + x$. However, testing all possible permutations of children at matching time has a factorial time complexity. We use a greedy pair-wise matching algorithm that considers all pair-wise alignments between children of matching commutative operators, and greedily chooses 1-to-1 matches between children maximizing the predicted number of matches after unification, breaking ties by preferring alignments with more exact matches. While suboptimal, this greedy approach can still be computed in polynomial time and it allows us to match $x + y = 0$ and $0 = y + x$ perfectly.

The output of matching is a subtree of the candidate formula that has been successfully aligned to the query. For re-ranking, we use the same three structural matching scores from Tangent-3 [17]: Maximum Subtree Similarity (MSS), negative count of candidate nodes matched with unification, and negative count of query nodes matched without unification. To better support linear regression, we replace the negative counts by the equivalent unified precision and recall without unification, producing the same lexicographic ordering as before using values in the range $[0, 1]$. The recall and precision computed here differ from those used by other formula retrieval methods, in that they are computed from subtrees after matching constraints are enforced.

Table 1: NTCIR-12 optional MathIR Wikipedia Formula Browsing Task. Average Precision@K per topic

	Relevant				Partially Relevant			
Method	P@5	P@10	P@15	P@20	P@5	P@10	P@15	P@20
MCAT [7]	**0.4900**	**0.3900**	**0.3317**	**0.2825**	**0.9100**	**0.8400**	**0.8067**	**0.7687**
Tangent-3 [1]								
Core	0.4150	0.3150	0.2650	0.2200	0.8100	0.7450	0.7117	0.6737
Matching	0.4450	0.2925	0.2517	0.2200	0.8250	0.6825	0.6533	0.6100
SLT								
Core	0.4000	0.3025	0.2567	0.2125	0.7900	0.7275	0.6950	0.6562
Matching	0.4550	0.3450	0.2817	0.2462	0.8350	0.7725	0.7400	0.6913
Regression	0.3900	0.2900	0.2450	0.2000	0.6300	0.5525	0.5117	0.4675
OPT								
Core	0.3650	0.2550	0.2050	0.1700	0.6250	0.4825	0.4200	0.3662
Matching	0.3550	0.2475	0.2017	0.1787	0.5550	0.4400	0.3800	0.3425
Regression	0.3150	0.2475	0.2000	0.1700	0.6250	0.4975	0.4317	0.3850
Combined	0.4400	0.3150	0.2583	0.2162	0.7000	0.6075	0.5550	0.5112

Table 2: NTCIR-12 optional MathIR Wikipedia Formula Browsing Task. Average Bpref per topic.

	Core		Matching		Regression		
Matches	SLT	OPT	SLT	OPT	SLT	OPT	Comb.
Relevant	0.4207	0.4227	0.4786	0.4760	0.5240	0.5127	**0.5530**
Partially Relevant	0.5126	0.4241	0.5351	0.4206	0.5569	0.5492	**0.5620**

Layer 3: Linear Regression. Using relevance judgments data from the NTCIR-12 Wikipedia Formula Retrieval task, we train a least squares linear regressor to combine the three scores from Layer-2 and produce a final rank score. While more complex functions could have been used in this step, we choose a simple method to avoid over-fitting the limited training data available, and to clearly observe which re-ranking scores best predict relevance.

Combined SLT/OPT Retrieval Approach. We combine results from SLTs and OPTs in a simple way. First, we perform symbol pair-based retrieval within a separate index for each representation. The top-k candidates obtained from each index are merged into a single list. Then, for each candidate we apply the detailed *matching* and *scoring* processes using both SLTs and OPTs representations, and we concatenate the similarity scores into a single vector. Finally, a linear regressor assigns a relevance score for the formula. In our experiments, we see that this simple combination obtains better rankings than using scores from just SLTs or OPTs.

4 EXPERIMENTS

To evaluate our approach, we use data from the NTCIR-12 MathIR competition [15]. Specifically, we use data from the optional MathIR Wikipedia Formula Browsing Task which has a corpus of 319,689 articles from English Wikipedia with more than half a million formulae. The task has 40 topics for isolated formula retrieval: 20 are concrete (without wildcards) and 20 include wildcards. Each wildcard query is a derived from a concrete query, with portions of the concrete query replaced by wildcards.

At NTCIR-12, the top-20 results for each topic from 8 submissions were evaluated for relevance. Each result was assessed by two human evaluators who scored them from 0 (irrelevant) to 2 (relevant). These scores were combined and each formula has a final relevance score between 0 and 4. A total of 2687 relevance assessments were produced by this method.

We evaluated the ranks produced by the model for each representation (SLT, OPT) at each retrieval stage (*Core, Matching, Regression*). The combined SLT/OPT approach (Section 3.3) is also considered, for a total of seven conditions. At the first stage, we select the top 1000 candidates for each query using $w = All$ (all tuples) and $EOB = True$. Given the limited number of relevance assessments available, for conditions using Linear Regression we grouped each concrete query with its corresponding wildcard version, and created 20 data folds. We then used cross-validation, repeatedly using one fold to test and the remaining 19 folds to train a linear regressor, until all queries have been processed.

Table 3: Average nDCG@K of ranks per topic for judged NTCIR-12 Wikipedia Formulae.

	All Topics		Concrete Only		Wildcard Only	
Condition	@5	@20	@5	@20	@5	@20
SLT						
Core	0.7109	0.7002	0.7991	0.7727	0.6236	0.6277
Matching	0.7534	0.7218	0.8033	0.7776	0.7036	0.6659
Regression	0.7943	0.7723	0.8031	0.7958	0.7855	0.7488
OPT						
Core	0.6978	0.7184	0.7889	0.7891	0.6066	0.6478
Matching	0.7459	0.7446	0.7889	0.8018	0.7028	0.6874
Regression	0.7519	0.7331	0.8008	0.7773	0.7031	0.6888
Combined	**0.8136**	**0.7908**	**0.8131**	**0.8088**	**0.8141**	**0.7728**

Table 1 compares our three-stage ranking systems, and systems participating in the NTCIR-12 competition. We used the TREC eval tool to compute the values of Precision@K with K in $\{5, 10, 15, 20\}$. Unlike the Tangent-3 submissions at NTCIR-12 [1], we ensured that the lexicographic order used for ranking after structural matching (the *Matching* conditions) was preserved by the TREC eval tool, by computing the ranks produced by the fixed-order MSS and tie breaker scores, and then using the reciprocal rank $(1/r)$ of each score vector as the final score. Formulae with identical score vectors are re-ranked by the TREC eval tool based on document id.

Mainly because of a large number of unrated formulas that are unfairly assumed to be irrelevant, Precision@K scores for the proposed conditions, specially the OPT-based, are lower than those for systems in the competition, with the exception of SLT Matching which are higher than Tangent-3 Matching. However, Tangent-S is simpler and faster than MCAT [1, 7], the best performing system in the competition.

The binary preference (Bpref) metric ignores unrated matches, and quantifies the ability of the ranking method to keep judged relevant matches ranked higher than irrelevant ones. Table 2 shows a comparison of Bpref values across different conditions of our own method. Unfortunately, Bpref values are not available for the original systems in the competition.

In terms of Bpref, we can see that in most cases the values increase at each layer of the retrieval model. For partially relevant results, with SLTs Bpref goes from 0.4207 in the initial set of candidates to 0.5240 after using linear regression, and from 0.4227 to 0.5127 for OPTs. When SLT and OPT scores are combined, Bpref increases to 0.5530. This confirms that each step of the pipeline improves the quality of the produced ranks, and that combining representations helps.

As the number of expressions judged per topic is relatively small, and given that our new conditions produce many unrated results in the top-20, we have used a different approach to further analyze the rankings produced. We re-ranked all judged formulas for each topic using each stage of our retrieval model, and we compared these ranks against the ideal rankings using nDCG@K with K = $\{5, 20\}$ as shown in Table 3. Consistently, the *Matching* stage improves the

Table 4: Top-5 formula results for Query-12 for each ranking stage for each representation

	Query-12: $O(mn \log m)$						
	Symbol Layout Trees			Operator Trees			Combined
	Core	**Matching**	**Regression**	**Core**	**Matching**	**Regression**	**Regression**
1.	$O(mn \log m)$	$O(mn \log m)$	$O(mn \log m)$	$O(mn \log m)$	$O(mn \log m)$	$O(mn \log m)$	$O(mn \log m)$
2.	$O(mn)$	$O(VE \log V)$	$O(mn \log p) = O(n \log n)$	$O(mn)$	$O(nk \log(n))$	$O(mn \log p) = O(n \log n)$	$O(mn \log p) = O(n \log n)$
3.	$O(mnp)$	$O(VE \log V \log(VC))$	$O(mn) = O\left(n^3 \log n\right)$	$O(mnp)$	$O(VE \log V)$	$O\left(mnr^2 \log \frac{1}{\epsilon}\right)$	$O\left(M(m) \log^2 m\right) = O(M(n) \log n)$
4.	$O(m + \log n)$	$O(Tm) = O\left(n^2 m \log n\right)$	$O\left(d^5 n \log^3 B\right)$	$\Theta(mnp)$	$O(n \log n)$	$O(pn(m+n \log n))$	$O\left(mnr^2 \log \frac{1}{\epsilon}\right)$
5.	$O\left(m\sqrt{n} \log n\right)$	$O(nk \log k)$	$O\left(mnr^2 \log \frac{1}{\epsilon}\right)$	$O(mr)$	$O(nk \log k)$	$O(Tm) = O\left(n^2 m \log n\right)$	$O(mn) = O\left(n^3 \log n\right)$

ranking quality produced by the initial spectral symbol pair matching for both representations. While linear regression consistently increases nDCG for SLTs, it produces a small decrease for OPTs compared to the lexicographic order of the same scores from the *Matching* stage. This may be due to a greater collinearity between Recall of nodes and MSS in the OPT space. In general, nDCG values for Concrete topics are higher than for the Wildcard topics. In the current matching procedure, we have observed that allowing Wildcards to match subtrees in the presence of poor unifications can cause bad partial matches to be ranked high.

An analysis of relevance against similarity scores reveals that while the means of MSS and node recall increase with the relevance of judged matches on both representations, node precision after unification is not well correlated with relevance and might hurt linear regression predictions and even re-ranking in general. This is not surprising since some bad partial matches have low recall but high precision. For example, a single query wildcard can be matched to an entire arbitrary expression tree with perfect precision.

Table 4 illustrates the differences in the Top-5 ranks for each stage of the model for the query $O(mn \log m)$, for both SLTs and OPTs. Differences in structure for each representation change the initial set of candidates extracted from the collection. The *Matching* columns show how the unification process helps in increasing the rank of partial matches that become exact after unification. This query illustrates how linear regression can sometimes produce less intuitive rankings than the simpler lexicographic match score ordering. It also shows how the OPT representation can give better rankings to equivalent expressions that have a slightly different layout like the candidate $O(nk \log(n))$ after unification.

It is important to acknowledge noise in the NTCIR-12 formula data. Many expressions are incorrectly but consistently converted into Content MathML from LaTeX. For example, the sub expression $f(x)$ is almost always converted to the tree corresponding to $f \times x$. Such errors in the source can lead to many undesirable partial matches for OPTs at retrieval time.

5 CONCLUSIONS

We have presented a comparison of the performance for two math expression representations using a three-layer retrieval model. We also presented a simple way to combine Symbol Layout Tree (SLT) and Operator Tree (OPT) representations into a single retrieval model. Overall, this combined model produced better rankings than the individual representations. Our results suggest that additional restrictions are needed for unification, to prevent undesirable matches and resulting rankings. In this study, we simply used the similarity scores proposed in the original retrieval model that we extended. However, the method proposed here may be used to incorporate and study additional similarity scores that may be better predictors for relevance (e.g., symbol and structure recall before unification). Similarity scores can also be combined with non-linear models for more accurate candidate selection and relevance prediction, while maintaining fast retrieval.

ACKNOWLEDGMENTS

We thank Frank W. Tompa and the reviewers for their valuable feedback. This material is based upon work supported by the National Science Foundation (USA) under Grant No. HCC-1218801.

REFERENCES

[1] Kenny Davila, Richard Zanibbi, Andrew Kane, and Frank Wm Tompa. 2016. Tangent-3 at the NTCIR-12 MathIR task. In *Proc. NTCIR-12*. 338–345.

[2] Liangcai Gao, Ke Yuan, Yuehan Wang, Zhuoren Jiang, and Zhi Tang. 2016. The math retrieval system of ICST for NTCIR-12 MathIR task. *Proc. NTCIR-12* (2016), 318–322.

[3] Ferruccio Guidi and Claudio Sacerdoti Coen. 2015. A survey on retrieval of mathematical knowledge. In *Proc. CICM*. Springer, 296–315.

[4] Radu Hambasan, Michael Kohlhase, and Corneliu-Claudiu Prodescu. 2014. MathWebSearch at NTCIR-11. In *Proc. NTCIR-11*. 114–119.

[5] H. Hiroya and H. Saito. 2013. Partial-match Retrieval with Structure-reflected Indices at the NTCIR-10 Math Task. In *Proc. NTCIR-10*. 692–695.

[6] Shahab Kamali and Frank Wm Tompa. 2013. Structural similarity search for mathematics retrieval. In *Intelligent Computer Mathematics, LNCS vol. 7961*. Springer, 246–262.

[7] Giovanni Yoko Kristianto, G Topić, and A Aizawa. 2016. The MCAT math retrieval system for NTCIR-12 MathIR task. In *Proc. NTCIR-12*. 323–330.

[8] Xiaoyan Lin, Liangcai Gao, Xuan Hu, Zhi Tang, Yingnan Xiao, and Xiaozhong Liu. 2014. A Mathematics Retrieval System for Formulae in Layout Presentations. In *SIGIR*. ACM, New York, NY, USA, 697–706.

[9] Tie-Yan Liu. 2009. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval* vol. 3 (2009), 225–331.

[10] Bruce R Miller and Abdou Youssef. 2003. Technical aspects of the digital library of mathematical functions. *Annals of Mathematics and Artificial Intelligence* vol. 38, 1-3 (2003), 121–136.

[11] Shunsuke Ohashi, Giovanni Yoko Kristianto, Goran Topić, and Akiko Aizawa. 2016. Efficient Algorithm for Math Formula Semantic Search. *IEICE TRANSACTIONS on Information and Systems* 99, 4 (2016), 979–988.

[12] M Rûzicka, Petr Sojka, and M Liška. 2016. Math indexer and searcher under the hood: Fine-tuning query expansion and unification strategies. In *Proc. NTCIR-12*. 331–337.

[13] Abhinav Thanda, Ankit Agarwal, Kushal Singla, Aditya Prakash, and Abhishek Gupta. 2016. A document retrieval system for math queries. (2016), 346–353.

[14] Ke Yuan, Liangcai Gao, Yuehan Wang, Xiaohan Yi, and Zhi Tang. 2016. A mathematical information retrieval system based on RankBoost. In *Proc. JCDL*. IEEE, 259–260.

[15] Richard Zanibbi, Akiko Aizawa, Michael Kohlhase, Iadh Ounis, Goran Topić, and Kenny Davila. 2016. NTCIR-12 MathIR Task Overview. In *Proc. NTCIR-12*. 299–308.

[16] Richard Zanibbi and Dorothea Blostein. 2012. Recognition and retrieval of mathematical expressions. *IJDAR* 15, 4 (2012), 331–357.

[17] Richard Zanibbi, Kenny Davila, Andrew Kane, and Frank Tompa. 2016. Multi-Stage Math Formula Search: Using Appearance-Based Similarity Metrics at Scale. *SIGIR* (2016), 145–154.

[18] Wei Zhong and Hui Fang. 2016. OPMES: A Similarity Search Engine for Mathematical Content. In *ECIR*. Springer, 849–852.