Math Formula Graph Retrieval Using Contrastive Learning Over Visual and Semantic Embeddings

Bryan Amador
ma5339@rit.edu
Rochester Institute of Technology
Rochester, NY, USA

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

Abstract

A key opportunity to make mathematical information more easily accessible is creating search engines that leverage relationships between visual and semantic formula representations. In this paper we introduce a formula retrieval model where visual and semantic embeddings are queried separately, but trained jointly. Embeddings are produced using two Relational Graph Convolutional Neural Networks (R-GCNs), which are jointly optimized using a self-supervised training task with a contrastive loss, where pairs of visual and semantic formula nodes are classified as being from the same formula or different formulas. To avoid information loss, we use node embeddings to retrieve visual and semantic formula graphs, with each scored separately using a greedy alignment between query and candidate nodes in the manner of ColBERT. We explore combining and selecting visual and semantic relevance scores in different ways. We present results for two math formula retrieval benchmarks, ARQMath and NTCIR-12. Results show comparable results against the state-of-the-art and significant improvement when combining visual and semantic information from formulas.

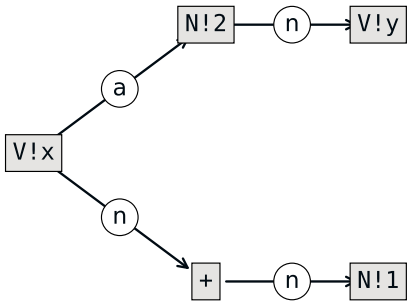
CCS Concepts

• **Computing methodologies** → *Neural networks*; • **Information systems** → **Mathematics retrieval**; *Document structure*.

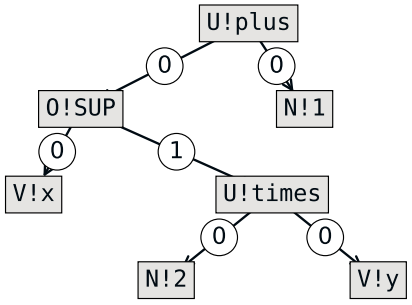
Keywords

Formula retrieval, Graph embeddings, Constrastive learning, Multi-vector retrieval, Math IR.

ACM Reference Format:
Bryan Amador and Richard Zanibbi. 2025. Math Formula Graph Retrieval Using Contrastive Learning Over Visual and Semantic Embeddings. In *Proceedings of the 2025 International ACM SIGIR Conference on Innovative Concepts and Theories in Information Retrieval (ICTIR) (ICTIR '25), July 18, 2025, Padua, Italy*. ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3731120.3744589



(a) Symbol Layout Tree



(b) Operator Tree

Figure 1: Symbol Layout Tree (SLT) and Operator Tree (OPT) for the formula $x^{2y}+1$. N!, V!, O!, and U! in node labels give symbols types as (N)umbers, (V)ariables, (O)rdered operators, and (U)nordered operators.

1 Introduction

Math-aware search has been of growing interest for the academic community due to its application in many fields of science and technology [6, 7, 14, 23]. Existing commercial search engines such as DuckDuckGo, Google and Bing lack effective access to math formulas since these are treated as plain text, which loses structural information. To address this issue, math-aware search engines have been built where formulas are handled differently than plain text. Formulas are naturally represented by trees rather than token sequences. Formula appearance is represented by the placement of symbols on writing lines using Symbol Layout Trees (SLTs), while the represented mathematical operations are captured in Operator Trees (OPTs). Examples of these are shown in Figure 1 for the formula $x^{2y}+1$. In the SLT, the root of the tree is the left-most symbol with edges identifying spatial relationships between symbols, while

ICTIR '25, Padua, Italy

ACM ISBN 979-8-4007-1861-8/2025/07
https://doi.org/10.1145/3731120.3744589

Table 1: Formula node & Symbol Layout Tree edge types.

	Example			Example	
Node Type	**Label**	**Exp.**	**SLT Edge**	**Label**	**Exp.**
number (N)	N!2	$2x$	next (n)	x **next** $+$	$x+$
const. (C)	C!∞	$\int_{\infty}^{0}$	above (a)	e **above** x	e^{x}
var. (V)	V!x	$2x$	below (b)	x **below** i	x_i
func. (F)	F!sin	$\sin x$	over (o)	x **over** $-$	$\frac{x}{y}$
text (T)	T!val	$x + \text{val}$	under (u)	y **over** $-$	$\frac{x}{y}$
group (M)	M!( )	$\binom{a}{b}$	pre-a. (c)	x **pre-a.** i	${}_{i}x$
unord. (U)	U!+	$a + b$	pre-b. (d)	x **pre-b.** i	${}_{i}x$
ord. (O)	O!$\subset$	$A \subset B$	elem. (e)	a **elem.** b	$\binom{a}{b}$
			within (w)	x **within**	$\sqrt{x}$

the OPT gives a hierarchy of operations with variable and number arguments at the leaves, and edges numbered to show the order of operands for operations. In practice, the choice of representing formulas by SLTs or OPTs depends upon the intended task: for retrieval both have been used.

For the SLT and OPT representations used in this paper, nodes are defined using pairs of the form `(type!value)`, except for operator symbols in SLTs, where operators have only a single symbol (e.g., for '+' in Figure 1). The different node types and types of SLT edges are shown in Table 1. Edges in the Operator Tree define the order of operands in an expression. In Figure 1, for the *ordered* superscript operation representing exponentiation (O!SUP), node 0 is the base (x) and node 1 is the exponent ($2y$). If an operator has *unordered* arguments, such as for commutative operations like addition and multiplication, all edges to arguments are labeled with 0.

Zanibbi et al. [24] categorize formula retrieval models according to the representation(s) and models used for retrieval: text, tree, embedding, visual-spatial, or their combination. We center our attention on embedding models that work from tree-based formula representations. Embedding models use statistical machine learning to represent formula similarity by end-point or angular distances between vectors in a euclidean space. *Dense* retrieval models that use embeddings address some limitations of *sparse* retrieval models that use fixed pattern vocabularies for lookup, e.g., in an inverted index. The *vocabulary problem* faced by sparse retrieval systems arises due to exact matching of graphs, nodes, edges, or paths: if a formula has tokens that are not in the vocabulary, they cannot be used for retrieval without additional processing. In contrast, in embedding models context is used to embed patterns in euclidean space even when a particular pattern (e.g., symbol) is unseen during training. To fully leverage the structural information of formulas represented as OPTs and SLTs, in our model we have used Graph Neural Networks (GNNs), which have been used widely to learn information from graphs [5, 21].

Due to a scarcity of relevance assessment data for formula retrieval, dense retrieval models for formulas are primarily trained using self-supervised learning. A widely used technique is *contrastive learning*, where we modify embeddings with the goal of placing equivalent/similar objects closer than distinct objects using designated 'positive' and 'negative' examples [8]. When using contrastive learning with GNNs [20, 26], augmentation/expansion methods are needed to produce variations of individual graphs. These are used both to increase the number of positive examples, and the variety in negative examples. We adopt this approach, and leverage the fact that we already have two graph representations for each formula in our benchmark data sets: SLTs and OPTs.

Motivated by the effectiveness of part-based matching and greedy alignment of token vectors for ColBERT [9] for text retrieval tasks, we present a contrastive learning-based retrieval model that operates at the formula graph node level, and uses the max similarity from ColBERT. This technique is used at the retrieval stage to perform multi-vector retrieval over node embeddings as well. We adopt the idea of adding backward edges to SLTs and OPTs from Ahmed et al.[1], and apply contrastive learning to formula retrieval in a manner similar to Wang et al. [20], but use node-level rather than formula-level embeddings. The code and model weights for our model are available online.[1]

The research questions that we pursue in this paper are:

- **RQ1:** Do formula representations that combine semantic and visual information through a contrastive loss improve retrieval results?
- **RQ2:** Does combining separate semantic and visual formula retrieval results using score fusion functions such as average and reciprocal rank fusion improve performance?
- **RQ3:** Is formula retrieval using the proposed granular contrastive learning framework comparable in effectiveness to current formula retrieval models?

2 Related Work

Formula Retrieval with GNNs and Contrastive Learning. An early use of GNNs for formula search is the work of Pfahler and Morik [16]. They embed math formulas into a single vector using a Graph Convolutional Network trained by node masking and contextual similarity, under the assumption that related expressions appear in similar contexts. Lin et al. [10] embed logical formula graphs using a Graph Convolutional Network and perform contrastive learning where positive pairs are logically equivalent formulas. The produced embeddings are used for entailment checking and premise selection. Song and Chen [18] embed OPTs into single vectors after merging nodes that represent the same symbol. They presented experiments using three different types of GNNs: Graph Convolutional Network, Graph SAmple and aggreGatE (GraphSAGE), and Graph Isomorphism Network. Models were trained in a self-supervised way using node masking and context prediction. Edge features are added to node features at the message passing step.

Ahmed et al. [1] used a Graph Convolutional Network to encode formula graphs in a single vector combined with visual features of formula images using regression. Their formula graphs add backward edges to SLTs and OPTs, which are called *Symbol Layout Graphs (SLGs)* and *Operator Graphs (OPGs)*, respectively. Edge and node features are concatenated at the message passing step. Training is performed by minimizing a triplet loss over positive and negative pairs constructed using graph symmetric scores, after which the learned embeddings are used for isolated formula retrieval. Wang et al. [20] used graph-based contrastive learning for single vector

[1] Source code and model weights: https://gitlab.com/ma5339/new_graph_retrieval

formula retrieval, starting from Tangent-CFT node embeddings for SLTs and OPTs [13]. They then apply contrastive self-supervised learning models using graph augmentation to generate positive and negative pairs. Wang and Tian [19] apply transformer-based models as enconders over MathML and LaTeX text tokens, and trained them jointly to produce embeddings for isolated formula retrieval. Presentation MathML and Content MathML (Symbol Layout Tree and Operator Tree) tokens are first trained jointly using a single encoder. The second training stage uses the encoder embeddings along with a second encoder for the LaTeX tokens. Their final formula representation is a single vector (i.e. one vector per formula).

Formula Retrieval Evaluation. Benchmarks for formula retrieval include include the NII Testbeds and Community for Information Access Research (NTCIR) (in their 10^{th}, 11^{th} and 12^{th} editons) and ARQMath (in their 1^{st}, 2^{nd} and 3^{rd} editions). The ARQMath data was obtained from Math Stack Exchange (MSE)[2], a popular mathematics question answering forum. Questions and answers from 2010-2018 are used as the test collection. Topics used for evaluation (i.e., test queries) are taken from questions appearing on MSE in 2019, 2020 and 2021 for ARQMath 1, 2 and 3 respectively, ensuring that no test questions appeared in the training set. The NTCIR-12 dataset for the MathIR task was created using math tagged documents from Wikipedia.

Some insights gleaned from these benchmarks are that combining formula appearance and mathematical operations tend to produce the best retrieval results [22], and sparse tree-based retrieval models performed best until the recent emergence of strong embedding models [11]. The state-of-the-art for the NTCIR12 dataset is MathBERT+Approach0 [15] , which trains a BERT model on Content MathML (OPT) and LaTeX tokens along with surrounding text, and results from this are combined with results from a sparse retrieval model using leaf-root tree paths extracted from OPTs (Approach0). The state-of-the-art model for ARQMath-3 Task 2 dataset is Approach0+ColBERT [25], which includes the surrounding text of formulas to contextualize formula representations.

Existing formula retrieval systems perform single vector retrieval, which we believe degrades performance: aggregating symbols and relationships in formulas into a single vector loses information. We address this by applying multi-vector retrieval similar to ColBERT.

3 Visual and Semantic Graph Embeddings

To produce effective embeddings for retrieval, both the visual and semantic information present in SLTs and OPTs must be taken into consideration. Formula relevance for some queries may depend more on formula appearance (e.g., to find a specific formula), or more on underlying operations (e.g., to find semantically equivalent formulas for theorem proving). For some queries, both visual and semantic elements may come into play (e.g., when browsing for formulas 'similar' to a query formula).

Formula graphs with reverse edges (SLG and OPG). Math formulas are commonly represented in the web using LaTeX or MathML, and for MathML most commonly Presentation MathML (i.e., 'visual' SLT representations described in Section 1). We parse MathML using the Tangent-S code base [4], producing Symbol Layout Trees (SLTs) from Presentation MathML and Operator Trees (OPTs) from Content MathML provided in the benchmarks used, where Content MathML was previously produced offline from LaTeX.[3]

We start with SLTs and OPTs, and adopt the idea of adding backward edges to them as proposed in Ahmad et al. [1]. This prevents nodes whose in-degree is 0 from never being updated during graph network training. These new edges contain a new label for reversed versions of the original edge. We define the resulting representations as Symbol Layout Graph (SLG) and Operator Graph (OPG), as shown in Figure 2.

R-GCN Embedding Architecture. To leverage graph structure in our formulas, we encode them using the Relational Graph Convolutional Network (R-GCN) proposed by Schlichtkrull et al. [17]. R-GCN extends grid convolution from Convolutional Neural Networks to graphs, and leverages edge types. Normally GNNs are trained using message passing, where each node state n_i is updated by receiving *messages* from its neighbors N_i. In R-GCN, messages are filtered by edge type, and each node is updated by aggregating neighbors separately for each relation (edge type). We apply this model because we expect the edge labels to carry important information. Similar GNN models for formula retrieval concatenate or add edge features to node features, which can decrease their influence.

The updating function of this model is given by Equation 1:

$$h^{(l+1)}i = \sigma\left(\sum_{r\in R}\sum_{j\in N_i^r}\frac{1}{c_{i,r}}W_r^{(l)}h_j^{(l)} + W_0^{(l)}h_i^{(l)}\right) \tag{1}$$

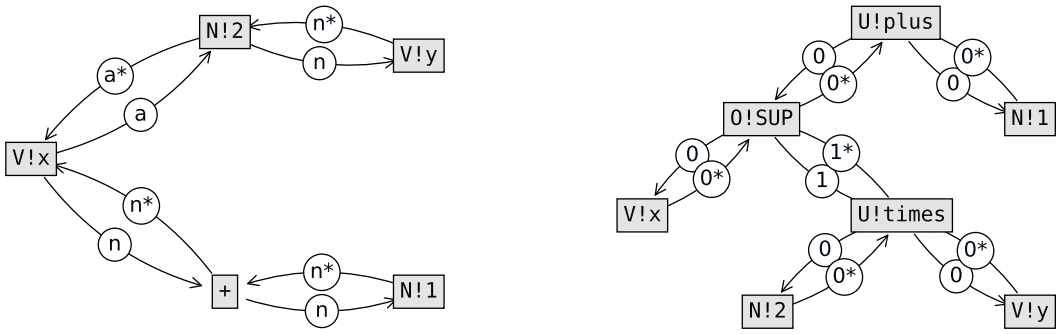
where h_i^{l+1} is the updated vector for node i. l is the number of the current layer. R is the set of relations of node i. N_i^r denotes the set of neighbors of node i under the relation r. $c_{i,r}$ is a normalization factor set to $|N_i^r|$. σ is an activation function (ReLU in our case). W_r represents the learning parameters for the relation r.

Training: Node Pair Classification & Sub-expressions. To leverage both visual and semantic representation and preserve granular interactions, we propose a self-supervised contrastive training at the node level. The granular interaction in this context comes from pairing nodes across a pair of graphs. For a given pair of graphs, we pair each node from the first graph with the node from the second graph with the maximum cosine similarity (in the maxsim manner of ColBERT). In the positive pairs, both graphs correspond to the same formula. Negative pairs are created using graphs from different formulas within a training batch.

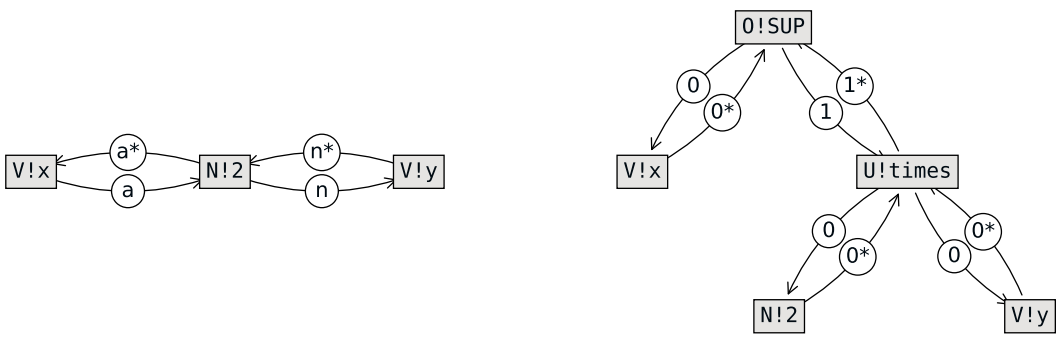
The core learning task will be to classify pairs of nodes as being from the *same* formula (positive pair), or two different formulas (negative pairs). However, we will make use of formulas both within and across SLG and OPG representations. In addition, we include SLG and OPG subexpressions, deciding 'positive' or 'negative' based on whether or not the subexpressions are taken from the same original formula in SLG *or* OPG form. Figure 2c and 2d show sub-expressions for the formula $x^2y + 1$ after removing the node `[+]` from the SLG and the corresponding node `[U!plus]` from the OPG. We use this subexpression strategy for augmentation rather than traditional graph augmentation techniques (add/remove edges,

[2]https://math.stackexchange.com/

[3]For an overview of known limitations in such a conversion, see Zanibbi et al. [24].



(a) Symbol Layout Graph (SLG) **(b) Operator Graph (OPG)**



(c) SLG sub-expression **(d) OPG sub-expression**

Figure 2: Example formula graphs for $x^{2y}+1$. Shown are the visual SLG (a), semantic OPG (b) and example sub-expressions from these (c,d). Both sub-expressions are obtained by removing the addition node in the SLG and OPG (`[+]` and `[U!plus]`).

feature perturbation, etc) because they may produce nonsensical formulas. By utilizing subexpressions, we hope to improve the ability both to retrieve formulas by parts (i.e., similar shared subexpressions), and to capture a larger variety of contexts for nodes and symbols for a single formula graph.

To select a sub-expression from a formula, we randomly remove an internal node from the graph (SLG or OPG) and select the largest connected component that remains after the removal. We do not remove leaf nodes because in OPTs these are operands, and may carry important information essential for retrieval. For instance, if we have a greek letter that appears very few times in the dataset and input a query formula that contains it, the retrieval of related formulas may be largely determined by this symbol.[4] This reasoning does not apply for SLTs, but use the same strategy for simplicity and consistency between representations. This sub-expression selection is made at every epoch to improve generalization.

We use two formula graphs to produce the positive and negative node classification pairs used in our contrastive learning. We have six different ways to produce these graph pairs, which use four graph types: SLG, OPG, SLG sub-expression and OPG sub-expression. Note that a pair of graphs is not commutative; the first graph is the reference point for matching nodes across graphs by maximum cosine similarity. Note also that each graph type pairing may come from the *same* formula (producing positive node pairs)

[4]This is related to TF-IDF models, where rarer symbols have a higher impact on relevance scores, on the assumption that rarer terms carry more information.

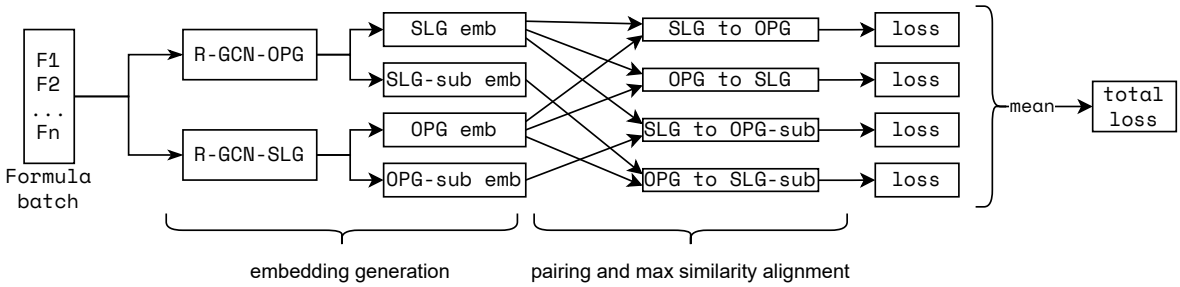


Figure 3: Training with contrastive learning across SLG and OPG nodes using the `Outer` set of graph augmentations. Positive and negative node pairs are created by maxsim alignment of nodes from the first to second type in each graph pairing. This occurs both for complete formulas (i.e., whole OPGs/SLGs), and for complete formulas in one representation vs. sub-expressions in the other representation (e.g., (SLG, OPG-sub). The final loss is averaged across the four graph type pairings.

or a *different* formula within the training batch (producing negative node pairs). The six approaches used to produce graph pairs and then positive/negative node pairs are listed below.

(1) **SLG**: Visual information only: (SLG, SLG-sub)
(2) **OPG**: Semantic information only: (OPG, OPG-sub)
(3) **SLG+OPG**: Complete formula graphs *across* visual/semantic representations: (SLG, OPG) and (OPG, SLG)
(4) **Inner (1 ∪ 2 ∪ 3)**: Sub-expressions *within* representation types and complete graphs *across* representation types:
(SLG, SLG-sub), (OPG, OPG-sub), (SLG, OPG) and (OPG, SLG).
(5) **Outer**: Complete formulas and sub-expression pairs *across* representations:
(SLG, OPG), (OPG, SLG), (SLG, OPG-sub), (OPG, SLG-sub).
(6) **Inner+Outer (4 ∪ 5)**: All six graph pairing types above:
(SLG, SLG-sub), (OPG, OPG-sub),
(SLG, OPG-sub), (OPG, SLG-sub),
(SLG, OPG) and (OPG, SLG).

Figure 3 illustrates the training process for augmentation condition 5 (*Outer*). In a given batch we generate embeddings for four graph types: SLT, OPG, SLG sub-expression and OPG sub-expression, using two R-GCNs: one for OPGs, and one for SLGs. Once we have these embeddings, we build our node pairs for training using the maxsim alignment described earlier. For instance, the most similar nodes in Figure 2a and Figure 2d would form positive pairs of the type (SLG, OPG sub-expression), with Figure 2a acting as the reference or 'query' graph when computing similarities.

We used as loss function for a node i a variation of the formula presented by Chen et al. [2] showed in Equation 2:

$$\ell_i = -\log \frac{\exp\left(\text{sim}(z_i, z_j^+)/\tau\right)}{\sum_{k=1}^{N-1} \exp\left(\text{sim}(z_i, z_k^-)/\tau\right)} \tag{2}$$

where N is the total number of formulas in a batch, τ is a parameter that controls the smoothness of the distribution (set to 0.1 after a grid search), z_i is the embedding for the current node, z_j^+ is the embedding of its positive partner and z_k^- is the embedding for the k^{th} negative partner; each node has 1 positive example and $N-1$ negative examples. The final loss is given by Equation 3:

$$\mathcal{L} = \frac{1}{2}\left(\frac{1}{M^l}\sum_{i=1}^{M^l}\ell_i + \frac{1}{M^o}\sum_{i=1}^{M^o}\ell_i\right) \tag{3}$$

where M^l is total number of nodes for all SLGs in the batch and M^o represents total of nodes in all OPGs.

4 Indexing and Retrieval Model

We implement indexing and retrieval using the vector database Qdrant [5], which uses the Hierarchical Navigable Small Worlds (HNSW) for search and provides an implementation of max similarity alignment for scoring candidates. As in ColBERT, HNSW is used to retrieve a first set of candidates by running a query per each node vector in the query graph, and later re-rank the candidate list using max sim alignment to produce the final results. We use a value of $k = 20$ (i.e. 20 neighbors in HNSW algorithm) for initial retrieval of graph nodes to obtain formula candidates.

Retrieval is performed separately for visual and semantic graphs using node embeddings, with separate SLG and OPG node embedding indexes. The resulting retrieval scores are then combined to produce the final retrieval result. In each case, we measure visual or semantic similarity by using multiple node vectors for each query and candidate.

We employ the greedy alignment proposed for ColBERT by Khattab et al. [9]. Similarity between query and candidate node vectors is obtained from the sum of maximum cosine similarities between node pairs. Given a query graph Q and candidate graph C, we compute node embeddings for each of them, producing the matrices E_Q and E_C. Each matrix has rows for nodes, and the number of columns is given by the embedding size. We denote the size of each graph in nodes using $|E_Q|$ and $|E_C|$, and enumerate query graph nodes using i and candidate graph nodes using j. The similarity S is then given by Equation 4:

$$S = \sum_{i \in \{1,\ldots,|E_Q|\}} \max_{j \in \{1,\ldots,|E_C|\}} E_{Q_i} \cdot E_{C_j}^T \tag{4}$$

[5] https://qdrant.tech/

Since we have embeddings for both SLG and OPG, we use this ensemble to score similarity using Equation 4 separately for each representation. We then fuse the similarity scores using static combination functions of different types, including MAX, MEAN, F1 and RRF (Reciprocal Rank Fusion [3]).

5 Results

Datasets. We present results on ARQMath-3 Formula Retrieval Task (Task 2) and NTCIR12 MathIR Task. For ARQMath-3 [11], participants return the top 1000 formulas given a formula query taken from a test question post. Returned formulas from all participants where then pooled and evaluated for relevance by math undergraduate and graduate students, taking the context of the post containing a retrieved formula into account. The full ARQMath dataset contains around 28 million formulas; approximately 9.3 million formulas are visually unique. The test collection contains 76 formulas queries (topics) with human-annotated assessments using a relevance scale of 0 to 3 (0 non-relevant, 3 high relevance). For binary metrics, 2 and 3 are considered relevant and the others not relevant. We make usage of the official metrics for the competitions: NDCG'@1000, MAP'@1000 and P'@10. Here, prime (') indicates that only evaluated hits are considered when computing the metrics. This insures that models are compared using the same pool of formulas with known relevance, whether they participate in the original task or run an evaluation afterward.

The NTCIR12 MathIR Wikipedia corpus contains 319,689 articles from English Wikipedia and contains around 590,000 formulas in the corpus [22]. There are around 250,000 visual unique formulas in this dataset. The task is to return top 1000 formulas for a given formula query chosen by the organizers. The test collection consists of 40 formula queries (topics) which results were human-annotated in a relevance scale of 0 to 4 (0 not relevant, 4 high relevance). For binary metrics, 3 and 4 are considered relevant and the others not relevant. We only use the first 20 queries because the remaining queries contain wildcards, and have been usually omitted when reporting NTCIR-12 results. Official metrics are Partial Binary Preference (Bpref@1000 full) and Full Binary Preference (Bpref@1000 full). Additional details about these test collections and evaluation metrics are available elsewhere [24].

Training Data. We trained our models using the visually unique formulas from the ARQMath-3 and NTCIR-12 test collections separately. This means that results shown in Tables 2 – 4 for ARQMath were produced with models trained using approximately 9.3 million visual unique formulas from ARQMath. However, results in Tables 2 and 5 for NTCIR-12 were produced using only about 250,000 visually unique formulas from NTCIR-12. As mentioned in Section 3, the training procedure is entirely self-supervised.

Efficiency Metrics. These metrics reported below were computed using ARQMath visually unique formulas (around 9.8 million) and the ARQMath-3 topics (76 formula queries). The system used was an Intel(R) Xeon(R) Gold 6326 CPU @ 2.90GHz, with 256 GB RAM, NVIDIA A40 GPU, and 2TB SSD disk.

Training time: around 6 hours per epoch.

Embedding (inference) time for indexing: around 3 hours using an embedding space of 100 dimensions. A sub-process for loading embedded formula batches into the indexing tool (Qdrant) runs concurrently, and so inference may be faster than reported here.

Indexing time: After all embedded node vectors are loaded in Qdrant, it takes around 24 hours for the index to be ready (green state) for retrieval. This includes creating the HNSW index data structures. It is important to note that we create 2 indexes: one for Symbol Layout Graph embeddings, and one for Operator Graph embeddings. The two indexes together contain around 400 million vectors for 9.8 millions formulas.

Retrieval time: around 2 minutes for top k=1000 retrieval for the 76 test queries. This is around 1.6 seconds per query. This includes querying the 2 indexes mentioned above.

Table 2: Retrieval results using different node pair augmentations. Significant differences vs. baseline SLG shown using † (Bonferroni corrected t-tests, $p < 0.05$).

	ARQMath-3 (76)			NTCIR-12 (20)	
Model	**NDCG'**	**MAP'**	**P'@10**	**Bpref p.**	**Bpref f.**
Single representation					
SLG	0.649	0.446	0.547	0.611	0.534
OPG	0.682	0.476	0.587	0.642	0.547
Combined SLG + OPG relevance scores					
SLG+OPG MAX	0.690	0.498	0.591	**0.647**	**0.572**
Inner MAX	0.694†	0.497†	**0.600**	0.646	0.555
Outer MAX	0.693†	0.496†	0.599	0.602	0.562
Inner + Outer:					
MAX	**0.701**†	**0.505**†	0.597	0.636	0.535
AVG	0.684†	0.486†	0.581	0.639	0.526
F1	0.670	0.475	0.576	0.635	0.532
RRF	0.678	0.485	0.599	0.623	0.523

RQ1 and RQ2: Effect of combining visual and semantic information. Table 2 shows a comparison of the different training variations in terms of how graph pairs are used to generate node pairs for classification. Also shown is the effect of using different combination functions for SLG and OPG retrieval scores: max, average, F1 and RRF. We selected the SLG-only model as our baseline to evaluate how our proposed method of combining both sources of information performs. We identified statistically significant differences using a multiple comparison t-test with a Bonferroni correction and a p-value of 0.05.

For the ARQMath collection, we can see a significant improvement against the baseline SLG for models where both semantic and visual information are used. This suggests that combining information at the node level enriches representations. We also ran multiple comparison t-tests (w. Bonferroni correction) against the base OPG model, but no significant differences were found. This suggests that this representation may be sufficient on its own. It is worth noting that a number of state-of-the-art models use only semantic information to perform retrieval. Nevertheless, we see an improvement in terms of deep metrics when representations are combined, in NCCG' and MAP'. For NTCIR-12, we did not observe significant differences between graph pairing conditions, possibly because of the small number of queries (20).

Table 3 shows results for two test queries from ARQMath-3. We show the top 5 formulas returned with an associated relevance rating in the qrels file. For the first query, the top 10 formulas with assessments are all of High or Medium relevance, giving a P'@10 of 100%, and NDCG'@1000 is 81.46%. We can attribute the success of our model in this query to the granular interaction: we can see that sub-expressions of the query appear in the retrieved formulas. We have a similar behavior (not shown) for query B.394:

$$\forall \epsilon > 0, \exists \delta < 0, |x - a| < \delta \implies |f(x) - f(a)| < \epsilon$$

where we observe 90% for P'@10, and 92.83% NDCG'@1000. Again, the granular interaction allows the models to retrieve formulas that contain sub-expressions of the query.

Table 3 also shows the top 5 (evaluated) results for query B.400. This is the second worst performing query for P'@10 with 10% (just 1 of the top 10 results is relevant), and an NDCG'@1000 of 56.5%. Note that the top hit just differs in 2 symbols with the query. Looking at the formula in isolation, we might be tempted to argue that this is a relevant hit – however, this was annotated as not relevant by accessors in ARQMath based on the Math Stack Exchange posts where the query and retrieved formulas appeared. Since we do not consider any context in our model, our results are penalized in these cases. In the future, we plan to incorporate the surrounding content for formulas.

RQ3: Comparison against existing models. Table 4 compares our system with Approach0 [25], a state-of-the-art formula search model that uses sparse retrieval and dynamic-programming based structural alignment for re-ranking. Leaf-root paths from OPTs are used for the sparse first-stage retrieval. We obtain stronger 'deep' metrics (NDCG' and MAP'), and the difference in P'@10 is less than 2%. This suggests that our approach is promising for isolated formula retrieval. Table 4 also shows models that implement contextualization of formulas by adding its surrounding text as an extra source of information. This is particularly important for the ARQMath dataset, where exact matches for certain queries may be irrelevant because their contexts are different. This explain why combining ColBERT with Approach0 performs slightly better.

Table 5 compares existing graph-based embedding systems for isolated formula retrieval in the NTCIR-12 dataset. In this case, we retrain our model from scratch using only roughly 250,000 visually unique formulas from the NCTIR-12 test collection. As mentioned earlier, the NTCIR-12 'concrete' formula evaluation is limited because of its low number of queries (20). However, this dataset has been widely used to evaluate isolated formula retrieval models, so we keep these results for the record. Note that with additional training data, we expect to obtain stronger results (e.g., closer to those for ARQMath-3).

Regarding transformers/LLMs for formula retrieval, Peng and Yuan [15] proposed a model called MathBERT. They included LaTeX and Operator Tree tokens in a text token sequence, and then used BERT to embed formulas and perform retrieval. This was evaluated on NTCIR-12 dataset. and their results are shown in Table 5. Their approach improves partial relevance matches over state-of-the-art graph-based formula search modes, but they obtain lower full Bpref than the EARN model. We believe that with the usage of text, we can match, or even exceed the performance of MathBERT.

Table 3: ARQMath-3 Formula Search Task Top-5 hits *with assessments* returned by the Inner + Outer MAX model for two queries.

Topic B.308: $\zeta(s) = \sum_{n=1}^{\infty} \frac{1}{n^s} = \frac{1}{\Gamma(s)} \int_0^{\infty} \frac{x^{s-1}}{e^x-1} dx$		
1.	$\zeta(s) = \sum_{n=1}^{\infty} \frac{1}{n^s} = \frac{1}{\Gamma(s)} \int_0^{\infty} \frac{x^{s-1}}{e^x-1} dx$	3 - High
2.	$\Gamma(s) = \sum_{n=1}^{\infty} \frac{\mu(n)}{n^s} \int_0^{\infty} \frac{x^{s-1}}{e^x-1} dx.$	3 - High
3.	$(1-2^{1-z})\zeta(z) = \sum_{s=1}^{\infty} \frac{(-1)^s}{s^z} = \frac{1}{\Gamma(z)} \int_0^{\infty} \frac{t^{z-1}}{e^t+1} dt$	3 - High
4.	$\zeta(s) = \frac{1}{\Gamma(s)} \int_0^{\infty} \frac{x^{s-1}}{e^x-1} dx.$	3 - High
5.	$\zeta(s) = \frac{1}{\Gamma(s)} \int_0^{\infty} \frac{x^{s-1}}{e^x-1} dx,\ s > 1.$	3 - High
Topic B.400: $\left(\frac{1}{2^{n-2}}\right)^2$		
1.	$\frac{1}{2^{(n-1)^2}}$	0 - Non-Rel.
2.	$\frac{1}{2^{n-2}} \leq 1$	0 - Non-Rel.
3.	$\left(\frac{1+i}{2}\right)^{2^n} = \frac{1}{2^{2^n-1}}$	0 - Non-Rel.
4.	$1/2^{n-2}$	2 - Med.
5.	$\frac{1}{2^{n-2}}$	1 - Low

Table 4: ARQMath-3 Formula Search Task Results (76 Test Queries). Approach0 is a state-of-the-art formula retrieval model. Text + formula search results are shown for context.

Model	NDCG'	MAP'	P'@10
Formula search			
Approach0 [25]	0.639	0.501	**0.615**
In.+Out *(ours)*	**0.701**	**0.505**	0.597
Text + formula search			
Approach0 + ColBERT [25]	**0.720**	**0.568**	**0.688**
TanCFT MathAMR [12]	0.640	0.388	0.478

More generally, in our model we currently treat all formula nodes equally. As a result, we may miss that some nodes are more important than others, which may in turn hurt performance in exact matching. In future work, we will explore attention mechanisms and similar strategies to address this.

6 Conclusion and Future Work

We have presented a new graph-based dense retrieval model for math formulas. The model uses a granular node-level training with self-supervised contrastive learning on visual and semantic formula representations. Two Relational Graph Convolution Networks (R-GCNs) are used to produce node embeddings for Symbol Layout Graphs (SLGs) and Operator Graphs (OPGs). The embeddings are

Table 5: NTCIR12 Formula Search Results for Queries Without Wildcards (20 Test Queries)

Model	Bpref partial	Bpref full
Graph-based formula search		
EARN [1]	**0.673**	**0.694**
GraphEmb [18]	0.540	0.630
SLG+OPG *(ours)*	0.647	0.572
Text + formula search		
MathBERT + Approach0 [15]	**0.736**	**0.613**

trained using contrastive learning to classify nodes as being from the same or different formulas, both within SLG and OPG represenations, and between them. Retrieval is performed at the node level, using a ColBERT-style max similarity alignment between formulas to produce relevance scores. Our experiments show that our approach is competitive with state-of-the-art isolated formula retrieval models, and that combining representations produces better results, particularly for partial matches as seen in deep metrics (NDCG' and MAP').

In the future, we will extend our model to a multi-modal (formula+text) retrieval pipeline, and explore new ways of using symbol pairs in training. We also want to explore attention mechanisms or similar strategies to emphasize salient regions in formulas during retrieval. Our approach is general, and could be used for other graph-based retrieval tasks, such as searching for molecules and chemical reactions. Such a system might be used by chemists to assist with literature searches and reaction planning.

Acknowledgments

This work was supported by the National Science Foundation (USA) through Grant #2019897 (Molecule Maker Lab Institute (MMLI), an NSF AI Center).

References

[1] Saleem Ahmed, Kenny Davila, Srirangaraj Setlur, and Venu Govindaraju. 2021. Equation Attention Relationship Network (EARN) : A Geometric Deep Metric Framework for Learning Similar Math Expression Embedding. In *2020 25th International Conference on Pattern Recognition (ICPR)*. 6282–6289. doi:10.1109/ICPR48806.2021.9412619

[2] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A Simple Framework for Contrastive Learning of Visual Representations. In *Proceedings of the 37th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 119)*, Hal Daumé III and Aarti Singh (Eds.). PMLR, 1597–1607. https://proceedings.mlr.press/v119/chen20j.html

[3] Gordon V. Cormack, Charles L A Clarke, and Stefan Buettcher. 2009. Reciprocal rank fusion outperforms condorcet and individual rank learning methods. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval* (Boston, MA, USA) *(SIGIR '09)*. Association for Computing Machinery, New York, NY, USA, 758–759. doi:10.1145/1571941.1572114

[4] 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* (Shinjuku, Tokyo, Japan) *(SIGIR '17)*. Association for Computing Machinery, New York, NY, USA, 1165–1168. doi:10.1145/3077136.3080748

[5] Vijay Prakash Dwivedi, Chaitanya K Joshi, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. 2020. Benchmarking graph neural networks. (2020).

[6] Deborah Ferreira, Marco Valentino, Andre Freitas, Sean Welleck, and Moritz Schubotz (Eds.). 2022. *Proceedings of the 1st Workshop on Mathematical Natural Language Processing (MathNLP)*. Association for Computational Linguistics, Abu Dhabi, United Arab Emirates (Hybrid). https://aclanthology.org/2022.mathnlp-1.0

[7] Ferruccio Guidi and Claudio Sacerdoti Coen. 2016. A Survey on Retrieval of Mathematical Knowledge. *Math. Comput. Sci.* 10, 4 (2016), 409–427.

[8] Longlong Jing and Yingli Tian. 2021. Self-Supervised Visual Feature Learning With Deep Neural Networks: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 43, 11 (2021), 4037–4058. doi:10.1109/TPAMI.2020.2992393

[9] Omar Khattab and Matei Zaharia. 2020. ColBERT: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval* (Virtual Event, China) *(SIGIR '20)*. Association for Computing Machinery, New York, NY, USA, 39–48. doi:10.1145/3397271.3401075

[10] Qika Lin, Jun Liu, Lingling Zhang, Yudai Pan, Xin Hu, Fangzhi Xu, and Hongwei Zeng. 2023. Contrastive Graph Representations for Logical Formulas Embedding. *IEEE Transactions on Knowledge and Data Engineering* 35, 4 (2023), 3563–3574. doi:10.1109/TKDE.2021.3139333

[11] Behrooz Mansouri, Vít Novotný, Anurag Agarwal, Douglas W. Oard, and Richard Zanibbi. 2022. Overview of ARQMath-3 (2022): Third CLEF Lab on Answer Retrieval for Questions on Math. In *Experimental IR Meets Multilinguality, Multimodality, and Interaction*, Alberto Barrón-Cedeño, Giovanni Da San Martino, Mirko Degli Esposti, Fabrizio Sebastiani, Craig Macdonald, Gabriella Pasi, Allan Hanbury, Martin Potthast, Guglielmo Faggioli, and Nicola Ferro (Eds.). Springer International Publishing, Cham, 286–310.

[12] Behrooz Mansouri, Douglas W Oard, and Richard Zanibbi. 2022. Dprl systems in the clef 2022 arqmath lab: Introducing mathamr for math-aware search. *Proc. CLEF 2022 (CEUR Working Notes)* (2022).

[13] 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*. 11–18.

[14] Jordan Meadows and Andre Freitas. 2022. A Survey in Mathematical Language Processing. arXiv:2205.15231 [cs.CL]

[15] Shuai Peng, Ke Yuan, Liangcai Gao, and Zhi Tang. 2021. Mathbert: A pre-trained model for mathematical formula understanding. *arXiv preprint arXiv:2105.00377* (2021).

[16] 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* (Virtual Event, CA, USA) *(KDD '20)*. Association for Computing Machinery, New York, NY, USA, 135–143. doi:10.1145/3394486.3403056

[17] Michael Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. 2018. Modeling Relational Data with Graph Convolutional Networks. In *The Semantic Web*, Aldo Gangemi, Roberto Navigli, Maria-Esther Vidal, Pascal Hitzler, Raphaël Troncy, Laura Hollink, Anna Tordai, and Mehwish Alam (Eds.). Springer International Publishing, Cham, 593–607.

[18] Yujin Song and Xiaoyu Chen. 2021. Searching for mathematical formulas based on graph representation learning. In *Intelligent Computer Mathematics: 14th International Conference, CICM 2021, Timisoara, Romania, July 26–31, 2021, Proceedings 14*. Springer, 137–152.

[19] Jingyi Wang and Xuedong Tian. 2023. Math Information Retrieval with Contrastive Learning of Formula Embeddings. In *Web Information Systems Engineering – WISE 2023*, Feng Zhang, Hua Wang, Mahmoud Barhamgi, Lu Chen, and Rui Zhou (Eds.). Springer Nature Singapore, Singapore, 97–107.

[20] Pei-Syuan Wang and Hung-Hsuan Chen. 2025. The Effectiveness of Graph Contrastive Learning on Mathematical Information Retrieval. In *Advances on Graph-Based Approaches in Information Retrieval*, Ludovico Boratto, Daniele Malitesta, Mirko Marras, Giacomo Medda, Cataldo Musto, and Erasmo Purificato (Eds.). Springer Nature Switzerland, Cham, 60–72.

[21] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. 2020. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems* 32, 1 (2020), 4–24.

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

[23] Richard Zanibbi and Dorothea Blostein. 2012. Recognition and retrieval of mathematical expressions. *International Journal on Document Analysis and Recognition (IJDAR)* 15, 4 (2012), 331–357.

[24] Richard Zanibbi, Behrooz Mansouri, and Anurag Agarwal. 2025. Mathematical Information Retrieval: Search and Question Answering. *Foundations and Trends® in Information Retrieval* 19, 1-2 (2025), 1–190. doi:10.1561/1500000095

[25] Wei Zhong, Yuqing Xie, and Jimmy Lin. 2023. Answer Retrieval for Math Questions Using Structural and Dense Retrieval. In *International Conference of the Cross-Language Evaluation Forum for European Languages*. Springer, 209–223.

[26] Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. 2021. Graph Contrastive Learning with Adaptive Augmentation. In *Proceedings of the Web Conference 2021* (Ljubljana, Slovenia) *(WWW '21)*. Association for Computing Machinery, New York, NY, USA, 2069–2080. doi:10.1145/3442381.3449802

Received 20 February 2007; revised 12 March 2009; accepted 5 June 2009