The Effectiveness of Graph Contrastive Learning on Mathematical Information Retrieval*

Pei-Syuan Wang and Hung-Hsuan Chen[0000−0001−5137−4449]

National Central University, Taoyuan, Taiwan
peistu13333@gmail.com, hhchen1105@acm.org

Abstract. This paper details an empirical investigation into using Graph Contrastive Learning (GCL) to generate mathematical equation representations, a critical aspect of Mathematical Information Retrieval (MIR). Our findings reveal that this simple approach consistently exceeds the performance of the current leading formula retrieval model, TangentCFT. To support ongoing research and development in this field, we have made our source code accessible to the public at https://github.com/WangPeiSyuan/GCL-Formula-Retrieval/.

Keywords: Mathematical information retrieval · Graphical contrastive learning · Layout.

1 Introduction

Search engines have revolutionized information access, enabling users to locate relevant textual content from the Internet quickly. Meanwhile, academic search engines and digital libraries, such as Google Scholar, CiteSeerX, and PubMed [3,21], have become indispensable tools in the academic field, allowing researchers to discover related works from a vast amount of documents. Although a general-purpose search engine and an academic search engine may use different strategies to evaluate the quality of a document (e.g., a search engine may analyze the hyperlink structures to infer the importance of a webpage, while an academic search engine may rely on the citation counts to gauge the quality of a paper [4]), they often rely on similar strategies to define the relevance score between a query term and a document. Popular techniques include term frequency statistics (e.g., TFIDF and its variants [9]) and distributed representations learning (e.g., Word2Vec, fastText, and Transformer [15]).

Mathematical formulas commonly play a central role in scientific papers, facilitating the precise expression of abstract ideas. It is crucial to develop methodologies that can effectively retrieve documents that contain mathematical formulas similar to a target formula. Unfortunately, the search for mathematical formulas is very different from a regular text-based search. While textual search algorithms focus primarily on word frequencies, syntactic structures, and

* We appreciate support from the National Science and Technology Council of Taiwan under grant 110-2222-E-008-005-MY3.

semantic associations, formula search requires a more profound comprehension of mathematical expressions, their inherent structures, and relationships between mathematical entities. Developing mathematical information retrieval (MIR) algorithms involves two main challenges. First, the model needs to capture the notation structure effectively. In MIR, the notation structure is perhaps more critical than string matching and term frequencies. For example, the quadratic equations $ax^2 + bx + c = 0$ and $\alpha\theta^2 + \beta\theta + \gamma = 0$ may convey the same concept, although they contain very different symbols. However, their structures are identical if we represent both equations using parse trees. Second, the labeled relevance score between pairs of mathematical formulas is needed for supervised training. Unfortunately, such datasets are limited. As a result, it could be difficult to apply machine-learned ranking (a.k.a. learning-to-rank) [13] methodologies. These obstacles make MIR still an extremely challenging task.

In this paper, we experiment with applying graph contrastive learning (GCL) on the graph generated from the formula structure to capture the notation structure without the help of labeled relevance scores between formulas, thus addressing the above two challenges. We define the similarity score between each pair of formulas based on the cosine similarity between their embeddings. We conduct experiments using the NTCIR-12 MathIR Wikipedia Formula Browsing Task [24]. Experimental results show that our model consistently outperforms the TangentCFT model [14], the state-of-the-art model for retrieving mathematical formulas. Note that our study focuses exclusively on models that rely solely on mathematical formulas for MIR. Therefore, models like MathBERT or Coco-MAE [16,26], which also incorporate additional information such as contextual texts, fall outside the scope of our analysis.

2 Related Work

This section reviews various methodologies in mathematics information retrieval, with a particular focus on TangentCFT, a state-of-the-art mathematics information retrieval method.

2.1 Analyze formula using text

Text-based MIR methods convert mathematical formulas into text formats such as LaTeXand MathML and use text similarity measures to assess the similarity between formulas. An example of this approach is the TF-IDF-based method called Math Indexer and Searcher (MIaS) [17]. It represents formulas in MathML format within an XHTML document and considers text and math formula components. However, this method largely overlooks mathematical formulas' structural and semantic aspects and relies mainly on a textual comparison based on words and their frequencies.

Other approaches employ complex natural language processing models to handle semantic retrieval. For example, Thanda et al. [20] utilized the PV-DBOW model to learn the embeddings of text paragraphs. Gao et al. [7] proposed

the symbol2vec and formula2vec models, which are based on the Continuous Bag-of-Words (CBOW) and Doc2Vec architectures [15,12], respectively, to learn embeddings. These approaches generally transform formulas into vector representations using semantic representation methods. However, these methods minimally consider the structure of the formulas.

2.2 Analyze formula using graph/tree

Tree-based methods consider the symbol structure and arrangement of mathematical formulas by representing them in a structured format, such as a Symbol Layout Tree (SLT) or an Operator Tree (OPT), and compare the similarity of the structures to perform retrieval.

Among tree-based methods, some compared the similarity of two tree structures by matching paths from the root node to the child nodes [8,25]. However, a successful match requires complete matching of root-to-leaf paths. Yokoi et al. [22] propose a more flexible method by extracting subpaths from the root node to the child nodes and performing matching based on these subpaths, thus increasing the success rate of matching. The MCAT method [11] used both OPT and SLT for path extraction, incorporating path features and information about the sibling nodes, combined with text-based search, to achieve better results. Another method, Approach0 [27], used only OPT for path extraction, generating paths representing subexpressions of mathematical formulas. The similarity calculation is based on the largest common subexpression among the formulas.

2.3 Integrating both formulas and contextual texts

Some studies leveraged the formulas and contextual texts for math information retrieval. For example, MathBERT [16], motivated by the success of pre-trained language models in natural language processing, utilized math formula, its layout, and the contextual texts into a Transformer for training. Coco-MAE [26] integrated the formula and textual information by contrastive learning. These studies leverage mathematical expressions and contextual texts, often leading to promising results in precision and recall.

2.4 TangentCFT

TangentCFT analyzes a formula based only on the formula but not the contextual text. TangentCFT begins by representing mathematical formulas using OPT and SLT. Next, TangentCFT traverses the tree and converts the paths in the tree into tuple sequences. These paths are then encoded and used to train embeddings using the fastText model [1]. Finally, mathematical formula embeddings are obtained by averaging the embeddings of the tuples in a formula.

To the best of our knowledge, when considering the retrieval of mathematical formulas without leveraging contextual text information, TangentCFT is among the effective models currently available [16]. Therefore, this paper uses TangentCFT as the baseline model and compares it with our approach.

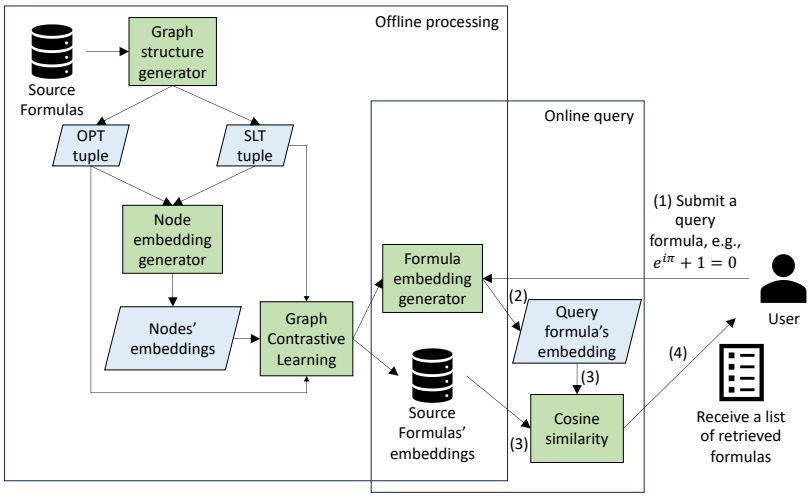


Fig. 1. The online and offline processing of the entire framework

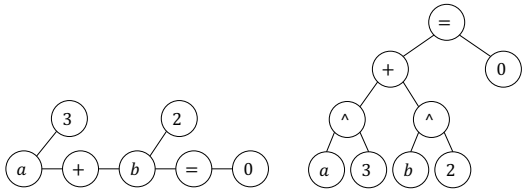
3 Methodology

We introduce the offline processing module and the online query module in this section. Figure 1 gives an overview of the whole workflow.

3.1 Offline Processing Module

The offline processing module includes a graph structure generator that outputs the OPT and SLT of a formula. We use TangentCFT to generate node (token) embedding, which will be the input of the graph contrastive learning models. The graph contrastive learning models generate formula embeddings based on contrastive learning; thus, the relevance scores between formula pairs are unnecessary. This section details the entire offline processing module.

Graph Structure Generator A mathematical symbol sequence can form graphs expressing semantic relationships between symbols. This study employs two graph structures to represent the relationship of the symbols in a mathematical formula: Symbol Layout Tree (SLT) and Operator Tree (OPT) [6]. The SLT is used primarily to indicate the spatial positioning of mathematical symbols in a written form. The OPT, on the other hand, is mainly used to capture the semantics of mathematical formulas. The OPT represents operators by an intermediate node, and the child nodes represent operands. Through the commutativity or associativity of operators, mathematically equivalent formulas with different appearances exhibit the same OPT structure.



(a) SLT example (b) OPT example

Fig. 2. The examples of the SLT and OPT representations of the formula $a^3 + b^2 = 0$

Table 1. A comparison of the properties of the graph contrastive learning models.

	InfoGraph	GraphCL	BGRL
Requires negative pairs	Y	Y	N
Requires graph augmentation	N	Y	Y
Contrastive pairs	Graph to node	Graph to graph	Node to node
Number of encoders	1	1	2

For example, given a formula $a^3 + b^2 = 0$, Figure 2 gives its SLT and OPT representations: SLT generates a graph that better preserves the layout of the writing, whereas the output of OPT captures the semantics of the equation.

Token Embedding Generator We may define the features for the nodes and edges of the SLT and OPT. For example, we could define a feature for a node that specifies whether the token represents an operator or operand. However, manually defining features can be tedious and perhaps subjective. Eventually, we decided to take advantage of the node and edge characteristics described in TangentCFT [14] and apply fastText [1] to the paths sampled by random walks to generate the embeddings for the nodes. We set each output embedding length to 100. These node embeddings are the building blocks for graph embeddings, which are representations of the formulas, as described below.

Formula Embedding Generator and Graph Contrastive Learning To generate formula embeddings, we need to assemble the node embeddings. There are at least two different ways to do it. The first is to compute the elementwise average for each node embedding in a graph, as TangentCFT does [14]. However, the simple average may be too naïve because the relationship among the math symbols (i.e., nodes) is missing. Another possibility is using the downstream task label as the ground truth and applying backpropagation to learn how to integrate the token embeddings. Unfortunately, our task has a limited number of relevance scores between pairs of formulas. Therefore, learning to integrate token embeddings based on a few labels will likely overfit the training data.

Eventually, we decided to employ graph contrastive learning methods to learn the embeddings of the formulas. GCL generates positive and negative graph pairs by manipulating the graph structures. Thus, the relevance score is not needed during training. We experimented with three representative GCL models: InfoGraph [18], GraphCL [23], and Bootstrapped Graph Latents (BGRL) [19]. Since these models require no training labels, we can generate the formula embedding even if a formula does not appear in the training data.

The InfoGraph model processes multiple graphs in one batch. InfoGraph learns to generate the local node embeddings and the global graph embeddings simultaneously such that a node n_i and a graph g_j have high mutual information if $n_i \in g_j$ and low mutual information otherwise. An advantage of InfoGraph is that it does not rely on graph augmentation techniques. However, InfoGraph assumes that a node's embedding alone can discriminate its belonging graph and other graphs, which could be an over-strong assumption.

The GraphCL model generates a positive graph pair by augmenting a given graph based on, for example, node dropping and edge perturbation. GraphCL regards a negative graph pair by the augmented graphs of two distinct graphs. The loss function encourages positive graph pairs to have similar embeddings and negative graph pairs to have dissimilar embeddings. Although such a technique works exceptionally well in computer vision [5], the data augmentation techniques used in graphs may merit further discussion. For example, in image classification, an image after standard augmenting procedures (e.g., rotating or resizing) would still likely be regarded as having the same label. However, standard graph-augmentation techniques make the graphs structurally different, especially when a graph is small. As a result, the performance of GraphCL may be substantially influenced by the graph-augmenting procedure.

Finally, the BGRL requires only positive pairs generated by graph augmentation. BGRL alleviates the need for negative pairs by applying two distinct encoders: one's parameters are learned via direct backpropagation, and the other's parameters are updated by an exponential moving average of the parameters in the first encoder. Although BGRL is highly scalable because it requires no negative pairs, BGRL still needs graph augmentation, which could still be an issue, as discussed above.

Table 1 compares these popular GCL models. Since each has its strengths and weaknesses, we tested all of them as formula embedding generation methods.

3.2 Online Query Module

A user submits to the system a query formula, which is used by the online query module to generate the query embedding based on the formula embedding generator trained offline. The system computes the cosine similarity between the query formula's embedding and each source formula's embedding. Finally, the system returns a list of the matched formulas by ranking the cosine similarities in descending order.

4 Experiments

4.1 Experimental Dataset and Evaluation Metrics

We used the NTCIR-12 MathIR Wikipedia Formula Browsing Task [24] as the data source for evaluation. Each relevance score is an integer between 0 and 4.

We used binary preference (bpref) and normalized discounted cumulative gain (nDCG) to evaluate the relevance of the returned formulas.

The bpref score evaluates a binary retrieval task (relevant/irrelevant) with incomplete information, i.e., the relevance scores of some documents can be unlabeled [2]. The bpref is a perfect evaluation score in our case because the relevance scores between most pairs of documents are unlabeled in our dataset (we have only $1,202$ labeled relevance scores). We consider a pair of documents relevant if their relevance score is 3 or 4; otherwise, they are irrelevant.

The definition of bpref is given in Equation 1.

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

where R and N represent the counts of relevant and irrelevant documents, respectively, with r as a relevant and n as an irrelevant document.

Although a binary judgment (relevant/irrelevant) is probably more straightforward for human evaluation [10], it fails to capture a fine-grained assessment. Therefore, we also applied the nDCG evaluation metrics because it allows for a graded relevance score. Equation 2 shows the formula of the nDCG score, which is the DCG score normalized by the ideal DCG score.

$$s_{\text{nDCG}} = \frac{s_{\text{DCG}}}{s_{\text{IDCG}}}, \tag{2}$$

where s_{IDCG} is the score of s_{DCG} when the top-K documents are perfectly ordered (i.e., they are ordered according to the relevance score in descending order). The DCG score is computed by Equation 3.

$$s_{\text{DCG}} = \sum_{i=1}^{K} \frac{r_i}{\log_2(i+1)}, \tag{3}$$

where r_i denotes the ith document's relevance score in the list ($r_i \in \{0, 1, 2, 3, 4\}$), and K is the count of returned documents ($K = 1,000$ in this experiment).

The nDCG is valid only with all returned documents scored for relevance. We filter out unjudged formulas from the list. Given each query has at most 90 judged formulas, and our $K = 1,000$ exceeds this, we utilize all judged documents.

In general, the nDCG score measures the effectiveness of a ranking algorithm by considering the relevance and position of items in a ranked list. It places a higher emphasis on the top positions. Additionally, nDCG accommodates graded relevance judgments, allowing for finer distinctions in the relevance of items. Meanwhile, the bpref allows unjudged documents in the list, and the binary judgment is likely more intuitive for most evaluators. Since the two metrics assess the quality of ranked lists from different perspectives, we use both for evaluations.

Table 2. The bpref scores of applying different models on SLT layout, OPT layout, and F1 score of the above two. TangentCFT is the baseline; InfoGraph, BGRL, and GraphCL are GCL models used by our approach.

Model	SLT	OPT	F1
TangentCFT	0.680 ± 0.0053	0.660 ± 0.0064	0.670
InfoGraph	0.691 ± 0.0066	0.685 ± 0.0070	0.688
BGRL	$\mathbf{0.701} \pm 0.0089$	0.683 ± 0.0077	0.692
GraphCL	0.685 ± 0.0090	$\mathbf{0.703} \pm 0.0072$	**0.694**

Table 3. The nDCG scores of applying different models on SLT layout, OPT layout, and F1 score of the above two. TangentCFT is the baseline; InfoGraph, BGRL, and GraphCL are GCL models used by our approach.

Model	SLT	OPT	F1
TangentCFT	0.841 ± 0.0032	0.830 ± 0.0041	0.835
InfoGraph	$\mathbf{0.860} \pm 0.0036$	0.851 ± 0.0063	0.855
BGRL	0.851 ± 0.0075	0.827 ± 0.0078	0.839
GraphCL	0.855 ± 0.0029	$\mathbf{0.864} \pm 0.0065$	**0.859**

4.2 Quantitative result

Table 2 and Table 3 present the quantitative evaluation results of the bpref and nDCG scores when applying different GCL models on either the SLT or OPT layouts. We repeat each experiment 5 times and report the mean and standard deviation in these tables. We also report the F1 score for the mean of SLT and the mean of OPT scores. Both the bpref and the nDCG metrics indicate that our self-supervised graph contrastive learning consistently achieves better retrieval performance than TangentCFT, and the results are very stable (since the standard deviations are close to 0). In particular, the bpref score implies that, on average, the genuinely relevant formulas retrieved by our model rank higher than irrelevant ones more often when compared to the state-of-the-art TangentCFT. The nDCG scores also indicate that our method is better at ranking the most relevant formulas near the top.

Interestingly, various GCLs are more effective with different layouts: GraphCL works better when OPT is used, while InfoGraph and BGRL are more successful when using SLT. We show the F1 score in the last columns of Table 2 and Table 3 to show the average effectiveness of each model on different layouts.

4.3 Case Study

This section shows the top retrieved formulas for two highly distinct query formulas. The first query involves a big-O expression with a logarithmic operation. Big-O notation is common in analyzing algorithms' time complexity. The inclusion of the log operation introduces mathematical complexity. Retrieving relevant

Table 4. The top returns of various models when querying "$O(mn \log m)$" (using SLT as the layout for graph construction.)

Rank	InfoGraph	GraphCL	BGRL
1	$O(mn \log m)$	$O(mn \log m)$	$O(mn \log m)$
2	$O(VE \log V)$	$O(n \log m)$	$O(m \log n)$
3	$O(nk \log k)$	$O(m \log n)$	$O(n \log m)$
4	$O(KN \log N)$	$O(mn)$	$O(mn)$
5	$O(m + \log n)$	$O(m^n)$	$O(m^n)$

Table 5. The top returns of various models when querying "$O(mn \log m)$" (using OPT as the layout for graph construction.)

Rank	InfoGraph	GraphCL	BGRL
1	$O(mn \log m)$	$O(mn \log m)$	$O(mn \log m)$
2	$O(n \log m)$	$O(n \log m)$	$O(mn)$
3	$O(m \log n)$	$O(mn \log(mn))$	$O(Mr)$
4	$O(n \log k)$	$O(m^2 n \log n)$	$O(mnp)$
5	$O(mn)$	$O(m \log n \log \log n)$	$\Theta(mn)$

results for such queries evaluates a model's capacity to deal with logarithmic functions, multiplications, and the big-O notation. The second query is an equation represented in matrix form. Equations involving matrices are prevalent in various scientific and engineering fields, including linear algebra, physics, and computer graphics. Retrieving relevant results for matrix equations is essential in applications like solving linear systems or optimizing operations on large datasets.

Tables 4 and 5 show the top-5 returns of the GCL models for the query with the big-O and logarithm expression. All the best-matched formulas are precisely the query formula. Additionally, all returns involve the big-O notation, except the 5th return of BGRL using OPT, which retrieves a highly relevant big-Θ notation. Also, some formulas with semantics identical to "$O(mn \log m)$" but using different symbols, such as $O(VE \log V)$ or $O(KN \log N)$, are retrieved, indicating that these models effectively handles polynomials, logarithm, and the big-O notation.

Table 6 and Table 7 show the top-5 formulas retrieved from the query with the matrix equation. We arrive at the same findings as in the previous case. First, all the top returns are the same as in the query formula. Moreover, the top-5 returns of the models using SLT and OPT as the layout for graph construction all contain matrices, except the 5th return of GraphCL using SLT. In addition, models can retrieve semantically similar formulas with different symbols.

5 Discussion

In this study, we investigate graph contrastive learning for formula retrieval to address two challenges of mathematical information retrieval: the model needs to

Table 6. The top returns of various models when querying "$\begin{bmatrix} V_1 \\ I_2 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} I_1 \\ V_2 \end{bmatrix}$" (using SLT as the layout for graph construction.)

Rank	InfoGraph	GraphCL	BGRL
1	$\begin{bmatrix} V_1 \\ I_2 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} I_1 \\ V_2 \end{bmatrix}$	$\begin{bmatrix} V_1 \\ I_2 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} I_1 \\ V_2 \end{bmatrix}$	$\begin{bmatrix} V_1 \\ I_2 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} I_1 \\ V_2 \end{bmatrix}$
2	$\begin{bmatrix} V_1 \\ I_1 \end{bmatrix} = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} V_2 \\ -I_2 \end{bmatrix}$	$\begin{bmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \end{bmatrix}$	$\begin{bmatrix} I_1 \\ V_2 \end{bmatrix} = \begin{bmatrix} g_{11} & g_{12} \\ g_{21} & g_{22} \end{bmatrix} \begin{bmatrix} V_1 \\ I_2 \end{bmatrix}$
3	$\begin{bmatrix} I_1 \\ V_2 \end{bmatrix} = \begin{bmatrix} g_{11} & g_{12} \\ g_{21} & g_{22} \end{bmatrix} \begin{bmatrix} V_1 \\ I_2 \end{bmatrix}$	$s_{(3,2,2,1)} = \begin{vmatrix} h_3 & h_4 & h_5 & h_6 \\ h_1 & h_2 & h_3 & h_4 \\ 1 & h_1 & h_2 & h_3 \\ 0 & 0 & 1 & h_1 \end{vmatrix}.$	$\begin{bmatrix} V_1 \\ I_1 \end{bmatrix} = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} V_2 \\ -I_2 \end{bmatrix}$
4	$\begin{bmatrix} V_1 \\ V_2 \end{bmatrix} = \begin{bmatrix} 0 & -r \\ r & 0 \end{bmatrix} \begin{bmatrix} I_1 \\ I_2 \end{bmatrix}$	$\begin{bmatrix} \frac{1}{h_{11}} & \frac{-h_{12}}{h_{11}} \\ \frac{h_{21}}{h_{11}} & \frac{\Delta[\mathbf{h}]}{h_{11}} \end{bmatrix}$	$\begin{bmatrix} V_2 \\ I_2' \end{bmatrix} = \begin{bmatrix} 1 & -R \\ -sC & 1 + sCR \end{bmatrix} \begin{bmatrix} V_1 \\ I_1 \end{bmatrix}$
5	$\begin{bmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \end{bmatrix}$	$h_{11} = \frac{V_1}{I_1}\Big\|_{V_2=0}$	$\begin{bmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \end{bmatrix}$

capture the notation structure and the lack of relevance score between formula pairs. We explore the potential of popular GCL methods, including InfoGraph, GraphCL, and BGRL. We investigate the OPT and SLT graph layouts and their influence on the retrieval results. We observe that the GCL models outperform TangentCFT, a state-of-the-art formula retrieval model. However, TangentCFT is still essential, as our GCL models utilize the node embeddings generated by TangentCFT as the input for these GCL models. We also use case studies to confirm that the methods can handle different formula queries.

To enrich the training instances and further enhance model robustness, future work could explore the generation of new equations as positive training pairs based on equation templates. For example, given a regular expression that generates polynomial equations, each pair of these generated formulas could be a potential positive pair. Another future work could be to improve the GCL data-augmentation process. Current strategies, randomly adding or removing nodes/edges from graphs generated from equations, may not always be ideal. Such adjustments may alter the semantics of the equation and introduce structural inconsistencies. Thus, we are also interested in developing more sophisticated graph-augmentation strategies that preserve the meaning and structure of the equation while increasing the diversity of training data.

References

1. Bojanowski, P., Grave, E., Joulin, A., Mikolov, T.: Enriching word vectors with subword information. Transactions of the association for computational linguistics **5**, 135–146 (2017)

Table 7. The top returns of various models when querying "$\begin{bmatrix}V_1\\I_2\end{bmatrix}=\begin{bmatrix}h_{11}&h_{12}\\h_{21}&h_{22}\end{bmatrix}\begin{bmatrix}I_1\\V_2\end{bmatrix}$" (using OPT as the layout for graph construction.)

Rank	InfoGraph	GraphCL	BGRL
1	$\begin{bmatrix}V_1\\I_2\end{bmatrix}=\begin{bmatrix}h_{11}&h_{12}\\h_{21}&h_{22}\end{bmatrix}\begin{bmatrix}I_1\\V_2\end{bmatrix}$	$\begin{bmatrix}V_1\\I_2\end{bmatrix}=\begin{bmatrix}h_{11}&h_{12}\\h_{21}&h_{22}\end{bmatrix}\begin{bmatrix}I_1\\V_2\end{bmatrix}$	$\begin{bmatrix}V_1\\I_2\end{bmatrix}=\begin{bmatrix}h_{11}&h_{12}\\h_{21}&h_{22}\end{bmatrix}\begin{bmatrix}I_1\\V_2\end{bmatrix}$
2	$\begin{bmatrix}I_1\\V_2\end{bmatrix}=\begin{bmatrix}g_{11}&g_{12}\\g_{21}&g_{22}\end{bmatrix}\begin{bmatrix}V_1\\I_2\end{bmatrix}$	$\binom{I_1}{I_2}=\begin{pmatrix}Y_{11}&Y_{12}\\Y_{21}&Y_{22}\end{pmatrix}\binom{V_1}{V_2}$	$\begin{bmatrix}I_1\\V_2\end{bmatrix}=\begin{bmatrix}g_{11}&g_{12}\\g_{21}&g_{22}\end{bmatrix}\begin{bmatrix}V_1\\I_2\end{bmatrix}$
3	$\begin{bmatrix}V_1\\V_2\end{bmatrix}=\begin{bmatrix}z_{11}&z_{12}\\z_{21}&z_{22}\end{bmatrix}\begin{bmatrix}I_1\\I_2\end{bmatrix}$	$\begin{bmatrix}\frac{1}{h_{11}}&\frac{-h_{12}}{h_{11}}\\\frac{h_{21}}{h_{11}}&\frac{\Delta[\mathbf{h}]}{h_{11}}\end{bmatrix}$	$\begin{bmatrix}I_1\\I_2\end{bmatrix}=\begin{bmatrix}y_{11}&y_{12}\\y_{21}&y_{22}\end{bmatrix}\begin{bmatrix}V_1\\V_2\end{bmatrix}$
4	$\binom{a_1}{b_1}=\begin{pmatrix}T_{11}&T_{12}\\T_{21}&T_{22}\end{pmatrix}\binom{b_2}{a_2}$	$\begin{bmatrix}h_{11}&h_{12}\\h_{21}&h_{22}\end{bmatrix}$	$\begin{bmatrix}K_{11}&K_{12}\\K_{21}&K_{22}\end{bmatrix}\begin{bmatrix}x_1\\x_2\end{bmatrix}=\begin{bmatrix}F_1\\F_2\end{bmatrix}$
5	$\begin{bmatrix}K_{11}&K_{12}\\K_{21}&K_{22}\end{bmatrix}\begin{bmatrix}x_1\\x_2\end{bmatrix}=\begin{bmatrix}F_1\\F_2\end{bmatrix}$	$\begin{bmatrix}\frac{\Delta[\mathbf{h}]}{h_{22}}&\frac{h_{12}}{h_{22}}\\\frac{-h_{21}}{h_{22}}&\frac{1}{h_{22}}\end{bmatrix}$	$\begin{pmatrix}A_1&B_1\\A_2&B_2\end{pmatrix}\binom{x}{y}=\binom{C_1}{C_2}.$

2. Buckley, C., Voorhees, E.M.: Retrieval evaluation with incomplete information. In: Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval. pp. 25–32 (2004)
3. Caragea, C., Wu, J., Ciobanu, A., Williams, K., Fernández-Ramírez, J., Chen, H.H., Wu, Z., Giles, L.: Citeseer x: A scholarly big dataset. In: Advances in Information Retrieval: 36th European Conference on IR Research, ECIR 2014, Amsterdam, The Netherlands, April 13-16, 2014. Proceedings 36. pp. 311–322. Springer (2014)
4. Chen, H.H., Treeratpituk, P., Mitra, P., Giles, C.L.: Csseer: an expert recommendation system based on citeseerx. In: Proceedings of the 13th ACM/IEEE-CS joint conference on Digital libraries. pp. 381–382 (2013)
5. Chen, T., Kornblith, S., Norouzi, M., Hinton, G.: A simple framework for contrastive learning of visual representations. In: International conference on machine learning. pp. 1597–1607. PMLR (2020)
6. Davila, K., Zanibbi, R.: 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. pp. 1165–1168 (2017)
7. Gao, L., Jiang, Z., Yin, Y., Yuan, K., Yan, Z., Tang, Z.: Preliminary exploration of formula embedding for mathematical information retrieval: can mathematical formulae be embedded like a natural language? arXiv preprint arXiv:1707.05154 (2017)
8. Hijikata, Y., Hashimoto, H., Nishida, S.: An investigation of index formats for the search of mathml objects. In: 2007 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology-Workshops. pp. 244–248. IEEE (2007)
9. Hsu, L.Y., Kao, C.H., Jheng, I.S., Chen, H.H.: Toward building an academic search engine understanding the purposes of the matched sentences in an abstract. IEEE Access **9**, 109344–109354 (2021)

10. Kekäläinen, J.: Binary and graded relevance in ir evaluations—comparison of the effects on ranking of ir systems. Information processing & management **41**(5), 1019–1033 (2005)
11. Kristianto, G.Y., Topic, G., Aizawa, A.: Mcat math retrieval system for ntcir-12 mathir task. In: NTCIR (2016)
12. Le, Q., Mikolov, T.: Distributed representations of sentences and documents. In: International conference on machine learning. pp. 1188–1196. PMLR (2014)
13. Liu, T.Y., et al.: Learning to rank for information retrieval. Foundations and Trends® in Information Retrieval **3**(3), 225–331 (2009)
14. Mansouri, B., Rohatgi, S., Oard, D.W., Wu, J., Giles, C.L., Zanibbi, R.: Tangent-cft: An embedding model for mathematical formulas. In: Proceedings of the 2019 ACM SIGIR international conference on theory of information retrieval. pp. 11–18 (2019)
15. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. Advances in neural information processing systems **26** (2013)
16. Peng, S., Yuan, K., Gao, L., Tang, Z.: Mathbert: A pre-trained model for mathematical formula understanding. arXiv preprint arXiv:2105.00377 (2021)
17. Sojka, P., Líška, M.: The art of mathematics retrieval. In: Proceedings of the 11th ACM symposium on Document engineering. pp. 57–60 (2011)
18. Sun, F.Y., Hoffmann, J., Verma, V., Tang, J.: Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization. arXiv preprint arXiv:1908.01000 (2019)
19. Thakoor, S., Tallec, C., Azar, M.G., Azabou, M., Dyer, E.L., Munos, R., Veličković, P., Valko, M.: Large-scale representation learning on graphs via bootstrapping. arXiv preprint arXiv:2102.06514 (2021)
20. Thanda, A., Agarwal, A., Singla, K., Prakash, A., Gupta, A.: A document retrieval system for math queries. In: NTCIR (2016)
21. Wu, J., Williams, K.M., Chen, H.H., Khabsa, M., Caragea, C., Tuarob, S., Ororbia, A.G., Jordan, D., Mitra, P., Giles, C.L.: Citeseerx: Ai in a digital library search engine. AI Magazine **36**(3), 35–48 (2015)
22. Yokoi, K., Aizawa, A.: An approach to similarity search for mathematical expressions using mathml. Towards a Digital Mathematics Library. Grand Bend, Ontario, Canada, July 8-9th, 2009 pp. 27–35 (2009)
23. You, Y., Chen, T., Sui, Y., Chen, T., Wang, Z., Shen, Y.: Graph contrastive learning with augmentations. Advances in neural information processing systems **33**, 5812–5823 (2020)
24. Zanibbi, R., Aizawa, A., Kohlhase, M., Ounis, I., Topic, G., Davila, K.: Ntcir-12 mathir task overview. In: NTCIR (2016)
25. Zhong, W., Fang, H.: Opmes: A similarity search engine for mathematical content. In: Advances in Information Retrieval: 38th European Conference on IR Research, ECIR 2016, Padua, Italy, March 20–23, 2016. Proceedings 38. pp. 849–852. Springer (2016)
26. Zhong, W., Lin, S.C., Yang, J.H., Lin, J.: One blade for one purpose: Advancing math information retrieval using hybrid search. In: Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval (2023)
27. Zhong, W., Zanibbi, R.: Structural similarity search for formulas using leaf-root paths in operator subtrees. In: Advances in Information Retrieval: 41st European Conference on IR Research, ECIR 2019, Cologne, Germany, April 14–18, 2019, Proceedings, Part I 41. pp. 116–129. Springer (2019)