Searching for Mathematical Formulas Based on Graph Representation Learning

Yujin Song and Xiaoyu Chen(✉)

LMIB-SKLSDE-BDBC, School of Mathematical Sciences, Beihang University, Beijing 100191, China
chenxiaoyu@buaa.edu.cn

Abstract. Significant advances have been witnessed in the area of representation learning. Recently, there have been some attempts of applying representation learning methods on mathematical formula retrieval. We introduce a new formula embedding model based on a kind of graph representation generated from hierarchical representation for mathematical formula. Such a representation characterizes structural features in a compact form by merging the same part of a mathematical formula. Following the approach of graph self-supervised learning, we pre-train Graph Neural Networks at the level of individual nodes to learn local representations and then produce a global representation for the entire graph. In this way, formulas can be embedded into a low-dimensional vector space, which allows efficient nearest neighbor search using cosine similarity. We use 579,628 formulas extracted from Wikipedia Corpus provided by NTCIR-12 Wikipedia Formula Browsing Task to train our model, leading to competitive results for full relevance on the task. Experiments with a preliminary implementation of the embedding model illustrate the feasibility and capability of graph representation learning in capturing structural similarities of mathematical formulas.

Keywords: Math formula search · Graph representation learning · Self-supervised learning · Formula embedding.

1 Introduction

Representation learning, also known as feature learning, has been demonstrated to be effective and powerful in various application fields including computer vision, audio, and natural language processing [3]. Its main objective is to represent data with low-dimensional dense vectors that are able to capture useful features of data through appropriate learning models other than artificial feature engineering. While such kind of distributed representations, taking advantages of excellent robustness and extensibility than symbolic representations of data, have been well studied on words and paragraphs, the question of how to learn continuous vector representations for mathematical formulas which play an important role in dissemination and communication of scientific information has become a compelling line of inquiry.

As for the task of information retrieval involving mathematical formulas, the goal is to learn appropriate formula embedding capable of capturing formula similarity. Considering the measures of formula similarity may vary in different scenarios which heavily affects the performance and evaluation of embedding, in this work we focus on structural similarity in isolated formula search task without using surrounding text, the same setting as in [15]. As formulas are inherently hierarchical and symbols therein have semantic relationships between each other which would be implicit in sequential representations, we wonder whether graph representation learning could enhance the embedding performance since significant advances have been achieved by Graph Neural Networks (GNNs). To this end, we introduce a new formula embedding model based on a kind of graph representation generated from hierarchical representation for mathematical formula. Such a representation characterizes structural features in a compact form by merging the same part of a mathematical formula. Given isolated formulas without meta-information, we leverage the self-supervised learning strategies at node level proposed in [11] to produce node embeddings and then average them to obtain a vector representation for the entire graph of a formula which allows efficient nearest neighbor search using cosine similarity. We explore the effects of different factors on embedding performance, including node feature initialization, the scale of node labels and different graph neural network architectures. Our model [1] achieves competitive results on NTCIR-12 Wikipedia Formula Browsing Task [26].

The remainder of this paper is structured as follows. We first review related work on mathematical formula representations, similarity measures in traditional mathematical information retrieval (MIR) and the new trend of mathematical formula embedding in Section 2. In Section 3, we present the design of graph representation for mathematical formulas and then briefly introduce self-supervised learning tasks and our embedding model based on GNNs in Section 4. We evaluate our model using two metrics for information retrieval in Section 5. Finally, we conclude the paper and prospect the future work in Section 6.

2 Related Work

Math formula representation There are two choices for representing math formula abstractly [27]: one is Symbol Layout Tree (SLT) indicating formula appearance, i.e., symbols' spatial arrangement towards a writing baseline; the other is Operator Tree (OPT) indicating formula semantics such as argument types, operator syntax, and their logical relations. These two abstract representations have their own specific file formats. Those for SLTs include LaTeX, a well-known markup language for typesetting, and Presentation

[1] PyTorch implementation: https://github.com/Franknewchen/MathEmb

MathML, a markup language based on XML for displaying math on the web. SLT representations may be ambiguous as the same symbol may be of different mathematical types in different contexts. To reduce the ambiguity, sTEX [12] and content LaTeX [17] provided semantic annotations for LaTeX documents. Those for OPTs include Content MathML and OpenMath [5,7], both standard XML languages, which could provide semantics to Computer Algebra Systems (CAS) like Maple and Mathematica to perform computations. Various tools allow for conversions between different formats. Schubotz et al. [22] presented nine tools and performed a quantitative evaluation of them on a benchmark dataset. Greiner-Petter et al. [9] introduced the first translation tool for special functions between LaTeX and CAS.

Similarity measure The matching between a query formula and indexed formulas is essential in MIR systems, so how to measure similarity between formulas is a key question. Approaches fall into two main categories, text-based and tree-based [28]. Text-based approaches linearize expression trees to text strings with normalization, such as using canonical orderings for commutative and associative operators, and replacing symbols by their mathematical types, while the hierarchical information of formulas may be lost to some extent due to the linearization. And then text retrieval methods can be applied, such as using term frequency-inverse document frequency (TF-IDF) [17,23]. Kumar et al. [19] proposed an approach to retrieve LaTeX string by matching the largest common substring, which could capture more structural information but required a quadratic algorithm. Tree-based approaches use expression trees directly, aiming at matching complete trees, subtrees or paths traversed in some order [4,6,13,31]. The more common substructures are matched, the more similar two formulas are considered. These approaches are more effective than the text-based according to the results of NTCIR [1,26], but often time-consuming because of the complexity of structure matching. Zhong et al. [30] proposed a rank-safe dynamic pruning algorithm for faster substructure retrieval.

Math formula embedding At an early stage, Thanda et al. [24] explored math formula embedding by using the distributed bag of words (PV-DBOW) model, a variant of doc2vec algorithm. Gao et al. [8] followed the continuous bags-of-words (CBOW), one architecture of the word2vec [16] model with negative sampling and distributed memory model of paragraph vectors (PV-DM) to learn vector representations for math symbols and formulas respectively. Krstovski and Blei [14] proposed a model to embed both equations and their surrounding text based on word embedding by treating an equation appearing in the context of different words as a "singleton word". The joint embedding of text and equation is also adopted by Yasunaga and Lafferty [25]. They applied a Recurrent Neural Network (RNN) to model each equation as a sequence of LaTeX tokens. Pathak et al. [18] conducted symbol-level embedding as in [8], and then created a Formula Entailment (LFE) module based on Long Short Term Memory (LSTM) neural network to recognize entailment between formula pairs. Mansouri et al. [15]

created tuples to represent depth-first paths between pairs of symbols and then embedded tuples using the fastText n-gram embedding model derived from the word2vec model. The above approaches are almost based on word embedding techniques in natural language processing tasks. Their use and effectiveness were explored by Greiner-Petter et al. in five different mathematical scenarios. The results show that math embedding holds much promise for similarity, analogy, and search tasks [10]. To the best of our knowledge, Pfahler et al. [20] fed tree-structured MathML formulas to GNNs for the first time. Aiming at retrieving formulas appearing in the paragraphs or articles related to query formulas rather than exact matches, they designed node-level masking and graph-level contextual similarity self-supervised learning tasks to embed mathematical expressions.

Benchmark for MIR As Aizawa and Kohlhase presented in [2], a major obstacle to MIR research is the lack of readily available large-scale datasets with structured mathematical formulas, carefully designed tasks, and established evaluation methods. Benchmarks published by NTCIR [1,26] and the latest released benchmark ARQMath [29] have promoted the development of MIR.

3 Graph Representation for Mathematical Formula

The formula dataset we work on is obtained from Wikipedia Corpus for NTCIR-12 Wikipedia Formula Browsing Task. The corpus contains 31,839 HTML articles in MathTagArticles directory and 287,850 HTML articles in TextArticles directory. Each formula therein is encoded in three formats: LaTeX, Presentation MathML and Content MathML. As the amount of formulas existing in TextArticles directory only account for 2% of the total number and those formulas are mainly in the form of isolated symbols, we construct a formula dataset by extracting a total number of 579,628 formulas from the articles in MathTagArticles directory and converting each formula from Content MathML into OPT representation by using tools developed in [6].

In order to make full use of the advances of GNNs, we propose a labeled directed acyclic graph (DAG) representation based on the OPT representation by sharing the recurring substructures therein. Such a representation characterizes structural features in a compact form. For example, as shown in Fig. 1, the number "1" and the part "$a - b$", which occur twice in tree representation while only occur once in graph representation respectively. In order to prevent the emergence of multiple edges, especially in the representation of matrices, we do not merge substructures if their parent nodes are the same. Considering the matrix $\begin{bmatrix} x & x \\ y & z \end{bmatrix}$ shown in Fig. 2(a), the node "x" cannot be merged otherwise a multiple edge between node "R" and node "x" will occur in the graph. In the case of its transposed matrix $\begin{bmatrix} x & y \\ x & z \end{bmatrix}$ (Fig. 2(b)), the sharing of node "x" can be conducted and the resulting graph is shown in Fig. 2(c).

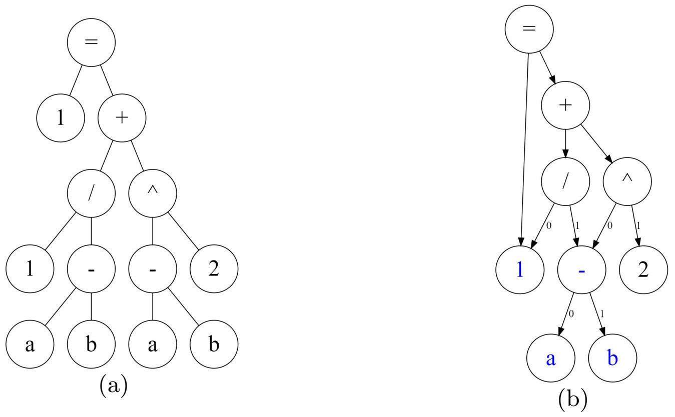

Fig. 1: Comparison of two representations for $\frac{1}{a-b}+(a-b)^2=1$. (a) OPT representation; (b) Graph representation with "1" and "$a-b$" shared (where edge labels are used to declare the argument positions).

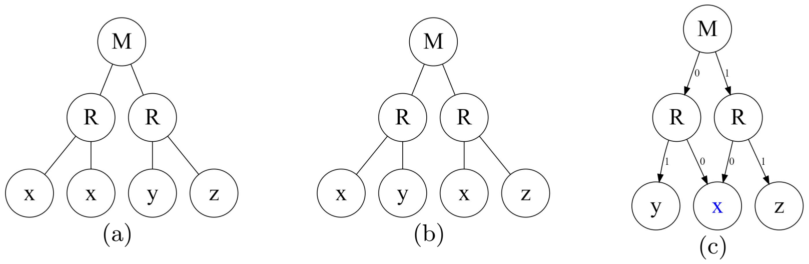

Fig. 2: Representations for matrices. "M" and "R" are abbreviations for "Matrix" and "Row" respectively.

Node Labels We label nodes in the form of "Type!Value" as proposed in [6]. There are 12 types extracted from Content MathML tags in total, including number (denoted as N!), constant (denoted as C!), variable (denoted as V!), function (denoted as F!), object with group structure (denoted as M!) like matrix, text (denoted as T!) like "lim" and "max", commutative operator (denoted as U!), non-commutative operator (denoted as O!), compound operator (denoted as +!) using a subtree to define an operation, whitespace (denoted as W!), unknown type (denoted as -!) and error (denoted as E!), while the value of a node is its corresponding symbols occurring in the formula. For example, the node labels of "a" and "1" in Fig. 1 are represented as "V!a" and "N!1" respectively. In the constructed formula dataset, there are 24,567 labels in the form of "Type!Value" totally, among which about 50% labels only occur once and 41% occur less than fifteen times. Since the amount of labels is so large and 91% of them only occur a few times, we discard the labels whose occurrence frequencies are less than a certain number δ (called *discarding threshold*) and the nodes with these labels are relabeled with a unified artificial symbol "[unif]" in our current setting. We compare the embedding performance when $\delta = 0, 5, 15$ and report the results in Section 5.

Edge Labels Edge labels are used to declare argument positions as in OPT representation [6]. For a commutative operator like "+" whose argument order does not affect the calculation result, argument edges are all labeled with "0". For a non-commutative operator, argument edges are labeled with indices in the argument order starting from 0.

For a mathematical formula, we implement an algorithm of converting its OPT representation into the above graph representation, denoted as a quadruple (V, E, L_V, L_E), where V is the set of nodes, E is the set of directed edges, L_V and L_E are the sets of node labels and edge labels respectively. Accordingly, the formula dataset in graph representation can be denoted as $\mathcal{G} = \{G_1, G_2, \ldots, G_{579628}\}$, where the whole set of node labels is denoted as $\mathcal{L_V} = L_{V_{G_1}} \cup L_{V_{G_2}} \cup \cdots \cup L_{V_{G_{579628}}}$ and $|\mathcal{L_V}| = 24,567$. As for the whole set of edge labels $\mathcal{L_E}$, elements are numbers indicating argument positions, where the minimum element is 0 and the maximum element is less than 68, the maximum degree of the nodes in $\mathcal{G}$. The node with the maximum degree appears in a polynomial with respect to "x" which has 69 terms and the degree of the polynomial is 71.

4 Formula Embedding Model

4.1 Self-supervised Learning Tasks

Since we focus on the isolated formula search task without meta-information, self-supervised learning would suit the task. In this work, we follow the strategies for pre-training GNNs presented in [11] which have enhanced the performances of downstream classification tasks in biology and chemistry fields. We adopt two self-supervised learning tasks therein at node level: (1) Context Prediction is a binary classification task of whether a particular subgraph and a particular context graph belong to the same node. The subgraph is a k-hop neighborhood around a randomly selected center node and the context graph is the surrounding graph structure that is between r_1-hop and r_2-hop ($r_1 < r_2$) from the center node. The neighborhood-context pair to be predicted is obtained using a negative sampling ratio of 1. (2) Attribute Masking is a task of predicting a set of randomly masked node labels using a linear model applied on top of GNNs, aiming to capture the regularities of node labels distributed over different graph structures. The masked nodes are labeled with "[mask]" during the process of training. We train GNNs by adding the losses for both two self-supervised learning tasks.

4.2 Graph Neural Networks

We adopt three architectures, Graph Convolution Network (GCN), Graph SAmple and aggreGatE (GraphSAGE) and Graph Isomorphism Network (GIN), to learn node representations and then produce the entire graph embedding by averaging its node representations. As input features of GNNs, each node and each edge with raw labels are respectively embedded by

$$h_v^{(0)} = \mathrm{EmbNode}\left(i_v\right) \tag{1}$$

$$h_e^{(0)} = \mathrm{EmbEdge}\left(j_e\right) \tag{2}$$

where i_v and j_e denote the index of node v's raw label in $\mathcal{L}_\mathcal{V}$ and the index of edge e's raw label in $\mathcal{L}_\mathcal{E}$ respectively. EmbNode$(\cdot)$ and EmbEdge$(\cdot)$ can be viewed as an initial layer for node features and edge features, mapping indices to randomly generated d-dimensional real vectors. In general, for each node v, GNNs update its representation at the k-th layer by

$$h_v^{(k)} = \mathrm{Combine}^{(k)}\left(h_v^{(k-1)}, \mathrm{Aggregate}^{(k)}\left(\left\{\left(h_v^{(k-1)}, h_u^{(k-1)}, e_{uv}\right) : u \in \mathcal{N}(v)\right\}\right)\right)$$

where e_{uv} is the feature vector of edge between node u and v, and $\mathcal{N}(v)$ is a set neighbors of node v. The representation of node v is iteratively updated by combining its last representation with aggregated information from its neighboring nodes and edges. Combine and Aggregate approaches are different in different network architectures, for details of which refer to [11].

Considering the inherited relationships between labels exiting in formula dataset, we also apply word2vec on the sequentialized form of formulas to extract initial features for nodes and construct the following embedding operation:

$$h_v^{(0)} = \mathrm{W2vNode}\left(l_v\right) \tag{3}$$

where $l_v \in \mathcal{L}_\mathcal{V}$ denotes the raw label of node v. W2vNode$(\cdot)$ first maps each label in $\mathcal{L}_\mathcal{V}$ to a $(d-1)$-dimensional real vector using word2vec. Then each vector is expanded to a d-dimensional vector by filling with 0 at the d-th dimension. For the label "[mask]", the first $d-1$ dimensions are all set to be 0, and the last dimension is set to be 1. For the artificial symbol "[unif]", the initial representation is generated randomly in each training batch to reduce manual bias caused by unifying the discarded labels. In a word, W2vNode$(\cdot)$ performs as a dictionary, providing initial features of fixed d-dimensional vectors for nodes that will be fed to GNNs.

4.3 Hyperparameter Choices

We train the presented model for 100 epochs with Adam optimization, embedding dimension of 300, batch size of 256, learning rate of 0.001, the number of GNN layers of 5, mask rate of 0.15 in Attribute Masking task, and $k = 5$, $r_1 = 2$ and $r_2 = 5$ in Context Prediction task.

5 Experiments and Evaluation

In this section, we evaluate our formula embedding model in different settings relying on 20 concrete queries provided by NTCIR-12 MathIR Wikipedia Formula Browsing Task.

5.1 Evaluation Metric

During the task, each hit from participating systems was evaluated by two human assessors recruited by organizers of NTCIR-12. Each assessor scores a hit with 0, 1 or 2, indicating the degree of relevance from low to high. The agreement between evaluators for this task has the lowest agreement value among all MathIR tasks since some evaluators were very concerned with formula semantics, while others seemed to consider primarily visual similarity when rating hits [26]. The final relevance rating is a score between 0 and 4, i.e. the sum of the two assessor scores. Scores of 3 or 4 are considered fully relevant while scores of 1 or 2 are considered partially relevant and a score of 0 is considered nonrelevant.

We use bpref on top-1000 results and precision@k for $k = 5, 10, 15, 20$ as evaluation metrics. For a query with R judged relevant documents and N judged nonrelevant documents, let r be a relevant document and n be a member of the first R judged nonrelevant documents as retrieved by the system, then

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

which is designed to evaluate IR systems only using judged documents [21]. Precision@$k = \frac{s}{k}$, where s is the number of relevant documents in the top k retrieval results, is an effective evaluation metric to indicate whether the top retrieval results are helpful to users.

5.2 Evaluation Results

We explore the impact of different graph neural network architectures, different initializations for node features and different scales of node labels as shown in Table 1. Firstly, we discard node labels with discarding threshold $\delta = 15$, so that the scale of node labels are reduced from 24,567 to 2,157. Then we initialize node features using EmbNode in which setting GCN performs the best. As a comparison, we adopt W2vNode to initialize node features. This setting only slightly improves the performance of GIN and GCN but impairs the performance of GraphSage by 2%. When the node labels are discarded with $\delta = 5$, the scale node labels is increased from 2,157 to 5,000. Then GCN achieves higher partial and full bpref scores, while both GraphSAGE and GIN perform inconsistent improvements on partial and full bpref scores. In both settings with different discarding thresholds, GCN performs the best, 1% to 5% better than the other two network architectures. This demonstrates that the classic GCN may be a more suitable encoder for formula embedding in characterizing structural features. Finally, to compare the effects of different discarding thresholds and initialization methods, we use GCN to conduct another round of experiment which shows that discarding threshold $\delta = 5$ helps improve the performance of GCN both in partial and full relevance, while using W2vNode does not help.

Table 1: Avg. bpref@1000 of NTCIR-12 results in different settings

Network Architecture	Size of Label Set	Initialization Method	Partial Bpref	Full Bpref
GCN	2,157	EmbNode	**0.5349**	**0.6167**
GraphSAGE	2,157	EmbNode	0.5244	0.5970
GIN	2,157	EmbNode	0.4845	0.5926
GCN	2,157	W2vNode	0.5256	**0.6195**
GraphSAGE	2,157	W2vNode	0.5058	0.5779
GIN	2,157	W2vNode	0.4982	0.6069
GCN	5,000	W2vNode	**0.5408**	**0.6250**
GraphSAGE	5,000	W2vNode	0.4937	0.5814
GIN	5,000	W2vNode	0.5185	0.5822
GCN	5,000	EmbNode	**0.5568**	0.6070
GCN	24,567	EmbNode	0.5278	0.5829
GCN	24,567	W2vNode	0.5309	0.5865

Next we compare our formula embedding model with other models, among which Tangent-CFT [15] is an embedding model, Approach0 [31] is a tree-based model and TanApp, the combination of Tangent-CFT and Approach0, achieves state-of-the-art performance. As Table 2 illustrates, our model achieves a competitive full bpref score. The reason for low partial bpref scores is that the top-1000 results retrieved by our model only hit a few judged formulas for some queries. An example is query #18, for which there are 71 judged formulas, but we only retrieve 14 of them. Another example is query #17 which had the highest harmonic mean bpref score (0.931) over all queries in Tangent-CFT retrieval results. The top-5 results retrieved by our model and Tangent-CFT are shown in Table 3. The three formulas ranked from the second to the fourth in our retrieval results are from the same article as the query is and seem more relevant than those in Tangent-CFT retrieval results. However, they were not judged during the task, leading to a low bpref score. In spite of this, our model can retrieve the exact match as the top-1 formula for each query except for query #1 (ranked the third in the retrieval results). The use of tokenizing formula tuples makes Tangent-CFT easier to retrieve formulas containing more same symbols as queries, which may be judged to be partially relevant, while our model focuses more on relationships between symbols and tends to retrieve formulas with similar structures as queries, which may be judged to be fully relevant.

The precision@k score is just a lower bound as some retrieved results may be not judged. We mainly compare our model with Approach0 considering that

Model	Partial Bpref	Full Bpref
Our Model	0.54	0.63
Tangent-CFT	0.71	0.60
Approach0	0.59	0.67
TanApp	0.73	0.70
Tangent-s [6]	0.59	0.64
MCAT [13]	0.57	0.57

Table 2: Avg. bpref@1000 of NTCIR-12 results of different models

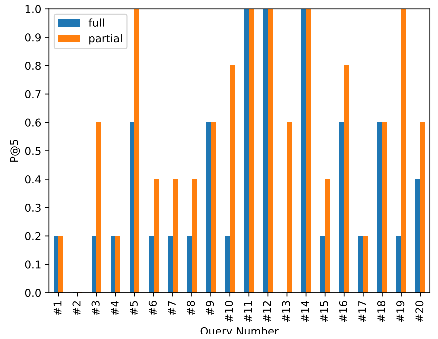

Fig. 3: P@5 for each query of our model

Table 3: Top-5 results for query $x - 1 - \frac{1}{2} - \frac{1}{4} - \frac{1}{5} - \frac{1}{6} - \frac{1}{9} - \cdots = 1$

Model	Rank	Retrieved Results
Our Model	1	$x - 1 - \frac{1}{2} - \frac{1}{4} - \frac{1}{5} - \frac{1}{6} - \frac{1}{9} - \cdots = 1$
	2	$x - 1 = 1 + \frac{1}{2} + \frac{1}{4} + \frac{1}{5} + \frac{1}{6} + \frac{1}{9} + \cdots$
	3	$x - 1 - \frac{1}{2} = 1 + \frac{1}{5} + \frac{1}{6} + \frac{1}{7} + \frac{1}{10} + \frac{1}{11} + \frac{1}{12} + \cdots$
	4	$x - 1 = 1 + \frac{1}{3} + \frac{1}{5} + \frac{1}{6} + \frac{1}{7} + \frac{1}{9} + \frac{1}{10} + \frac{1}{11} + \cdots$
	5	$\frac{1}{1} + \frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \frac{1}{5} + \frac{1}{8} + \cdots = \psi$
Tangent-CFT	1	$x - 1 - \frac{1}{2} - \frac{1}{4} - \frac{1}{5} - \frac{1}{6} - \frac{1}{9} - \cdots = 1$
	2	$1 - \frac{1}{2} - \frac{1}{4} + \frac{1}{3} - \frac{1}{6} - \frac{1}{8} + \frac{1}{5} - \frac{1}{10} - \frac{1}{12} + \cdots$
	3	$1 - \frac{1}{2} - \frac{1}{4} + \frac{1}{8} - \frac{1}{16} + \cdots = \frac{1}{3}$
	4	$\frac{1}{18} = \frac{1}{2} - \frac{1}{3} - \frac{1}{3^2}$
	5	$\frac{\pi}{4} = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \cdots$

all or partly results of other models were judged during NTCIR-12 task and the precision@k score of Tangent-CFT was not provided in [15]. As shown in Table 4, our model achieves competitive scores compared to Approach0 in both full and partial relevances. The reason why the score for query "β" (query #2) is zero is that there are so many unjudged "β" existing in the dataset that our model only takes the unjudged ones as top-5 results. And the zero score for query #13 in full relevance is caused by an error in the public assessment scores. [2] It is worth noting that the ideal precision@k score of full relevance is not 1 for every query, because if there is only one formula judged to be fully relevant among all judged formulas for some query, the ideal score is 0.2. Our model achieves ideal scores for 8 queries in full relevance and 5 queries in partial relevance and

[2] When we carry out experiments, we find an error score in "NTCIR12-MathWiki-13 xxx Mathematical_morphology:24 2.0", a line in the document that contains all judged formulas with relevance scores. The query formula with name "NTCIR12-MathWiki-13" is exactly the 24-th formula in the article "Mathematical_morphology". Therefore, the score should be 4.0 indicating full relevance.

additionally outperforms the average P@5 score of Approach0 for 7 queries in partial relevance.

Remark Because queries #1 and #2 are both isolated symbols represented as single nodes in our graph representation which contain no structural information to learn, we evaluate our model on the other 18 queries and achieve a partial bpref score 0.57 and a full bpref score 0.65 respectively, and higher P@k scores.

Table 4: Avg. P@k of NTCIR-12 results of our model and Approach0

Model	Partially Relevant				Fully Relevant			
	P@5	P@10	P@15	P@20	P@5	P@10	P@15	P@20
Our Model	0.5900	0.4450	0.4100	0.3800	0.3900	0.2500	0.2200	0.1875
Approach0	0.5300	0.4650	0.4100	0.3850	0.4000	0.2900	0.2233	0.1950

In order to intuitively compare our model with other models, we illustrate some specific queries. Our model achieves ideal P@5 score in full relevance for the following queries:

$$O(mn \log m) \qquad \text{(query \#12)}$$

$$\cos\alpha = -\cos\beta\cos\gamma + \sin\beta\sin\gamma\cosh\frac{a}{k}. \qquad \text{(query \#14)}$$

For query #12, the top-5 results retrieved by our model, Tangent-CFT and Approach0 are shown in Table 5. It is obvious that the retrieval results by our model are highly consistent with those by Approach0, i.e., after substituting some single symbols, the retrieval results become the same as the query. As for query #14, in the top-10 results Tangent-CFT missed 4 fully relevant formulas found in that range by Approach0, including $\cos A = -\cos B\cos C + \sin B\sin C\cosh a$ and $\cos C = -\cos A\cos B + \sin A\sin B\cosh c$. Our model not only hits the two formulas, but also hits the query itself as the first one in the top-10 results.

Table 5: Top-5 results for query $O(mn\log m)$

Rank	Our Model	Tangent-CFT	Approach0
1	$O(mn\log m)$	$O(mn\log m)$	$O(mn\log m)$
2	$O(nk\log k)$	$O(m\log n)$	$O(nk\log k)$
3	$O(nk\log(n))$	$O(n\log m)$	$O(KN\log N)$
4	$O(KN\log N)$	$O(n\log m)$	$O(VE\log V)$
5	$O(VE\log V)$	$O(nm)$	$O(n\log n\log\log n)$

Another example is $0 \to G^\wedge \overset{\pi^\wedge}{\to} X^\wedge \overset{\imath^\wedge}{\to} H^\wedge \to 0$ (query #7), for which Tangent-CFT performed better in both partial and full bpref than Approach0.

Among the top-1000 results, Tangent-CFT was able to retrieve formulas such as $1 \to K \xrightarrow{i} G \xrightarrow{\pi} H \to 1$ and $W \to X \xrightarrow{f} Y \xrightarrow{g} Z \xrightarrow{h} 1$ which Approach0 failed to retrieve. Our model performs better than them. The following four formulas

$$0 \to G^\wedge \xrightarrow{\pi^\wedge} X^\wedge \xrightarrow{\imath^\wedge} H^\wedge \to 0,$$

$$0 \to H \xrightarrow{i'} X' \xrightarrow{\pi'} G \to 0,$$

$$1 \to K \xrightarrow{i'} G' \xrightarrow{\pi'} H \to 1,$$

$$0 \to H \xrightarrow{i_H} H \times G \xrightarrow{\pi_G} G \to 0,$$

are in the top-5 results. The second and the fourth formula were not judged, which is the reason why the P@5 scores in full and partial relevances are only 0.2 and 0.4 respectively, but they are actually from the same article as the query is.

Consider the following three queries:

$$\tau_{\text{rms}} = \sqrt{\frac{\int_0^\infty (\tau - \overline{\tau})^2 A_c(\tau) d\tau}{\int_0^\infty A_c(\tau) d\tau}} \qquad \text{(query \#16)}$$

$$P_i^x = \frac{N!}{n_x!(N-n_x)!} p_x^{n_x} (1-p_x)^{N-n_x} \qquad \text{(query \#18)}$$

$$r_{xy} = \frac{\sum\limits_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{(n-1) s_x s_y} = \frac{\sum\limits_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum\limits_{i=1}^n (x_i - \bar{x})^2 \sum\limits_{i=1}^n (y_i - \bar{y})^2}}. \qquad \text{(query \#20)}$$

The three queries have complicated structures containing more operators and symbols. For query #16, our model can retrieve

$$\alpha_{\text{sun}} = \frac{\int_0^\infty \alpha_\lambda I_{\lambda\text{sun}}(\lambda)\, d\lambda}{\int_0^\infty I_{\lambda\text{sun}}(\lambda)\, d\lambda}$$

as Tangent-CFT did, which is ranked the sixth in our results. In addition, formulas such as

$$\frac{1}{\kappa} = \frac{\int_0^\infty (\kappa_{\nu,\text{es}} + \kappa_{\nu,\text{ff}})^{-1} u(\nu, T) d\nu}{\int_0^\infty u(\nu, T) d\nu},$$

$$(\Delta k)^2 = \frac{\int_{-\infty}^\infty (k - k_0)^2 F(k) F^*(k)\, dk}{\int_{-\infty}^\infty F(k) F^*(k)\, dk},$$

which are also judged to be fully relevant, are ranked in top-5 in our results. For query #18, Approach0 could retrieve two formulas judged partially relevant while Tangent-CFT was not able to do so. In our top-5 results, the following two fully relevant formulas can be retrieved.

$$f(p) = \frac{(n+1)!}{s!(n-s)!} p^s (1-p)^{n-s}$$

$$p_n(k) = \frac{n!}{(n-k)!k!} p^k (1-p)^{n-k}$$

For query #20, the formula with the largest number of nodes and the deepest depth, Tangent-CFT could not retrieve the second formula below based on OPT representation, while our model could retrieve it and rank it in top-15. Besides, the first formula below is ranked the second in our results. They are both fully relevant.

$$\mathrm{sim}(d_j, q) = \frac{\mathbf{d_j} \cdot \mathbf{q}}{\|\mathbf{d_j}\| \, \|\mathbf{q}\|} = \frac{\sum_{i=1}^{N} w_{i,j} w_{i,q}}{\sqrt{\sum_{i=1}^{N} w_{i,j}^2} \sqrt{\sum_{i=1}^{N} w_{i,q}^2}}$$

$$\text{similarity} = \cos(\theta) = \frac{\mathrm{A} \cdot \mathrm{B}}{\|\mathrm{A}\| \|\mathrm{B}\|} = \frac{\sum_{\mathrm{i}=1}^{\mathrm{n}} \mathrm{A_i} \times \mathrm{B_i}}{\sqrt{\sum_{\mathrm{i}=1}^{\mathrm{n}} (\mathrm{A_i})^2} \times \sqrt{\sum_{\mathrm{i}=1}^{\mathrm{n}} (\mathrm{B_i})^2}}$$

Our model also performs well on continued fractions and matrices, such as queries #5, #9 and #19. For query #5, the partial bpref score is 1.00, which means that all 35 formulas judged as relevant during the task can be retrieved by our model and no nonrelevant formulas rank above them. Queries #9 and #19 demonstrate the strength of our model in terms of matrix formulas. For example, for query #9

$$\begin{bmatrix} V_1 \\ I_2 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} I_1 \\ V_2 \end{bmatrix},$$

formulas equivalent to the query after substituting variable names could be retrieved in top-10 results, such as

$$\begin{bmatrix} I_1 \\ V_2 \end{bmatrix} = \begin{bmatrix} g_{11} & g_{12} \\ g_{21} & g_{22} \end{bmatrix} \begin{bmatrix} V_1 \\ I_2 \end{bmatrix},$$

$$\begin{bmatrix} a_1 \\ b_1 \end{bmatrix} = \begin{bmatrix} T_{11} & T_{12} \\ T_{21} & T_{22} \end{bmatrix} \begin{bmatrix} b_2 \\ a_2 \end{bmatrix}.$$

Remark For the proposed DAG representation, merging the same substructures would not change the linkage between these substructures and their parent nodes. In other words, for each node in such a DAG, its "children nodes" and "parent nodes" are the same as those in the OPT representation. The first reason why we use DAG rather than OPT is the amount of computation of GNNs can be reduced. For the same subexpressions, the same node embedding will be computed repeatedly in OPT while only once in DAG. If the occurrence times or size of the same subexpression is large in an expression, then DAG will demonstrate its advantage in computing efficiency. The second reason is that the presented self-supervised learning tasks would benefit from such a compact form of representation as recurring substructures are ignored.

The DAG representation preserves structural differences of formulas but only children nodes' messages are aggregated when using GNNs to learn node embeddings. This is the main reason that our model performs more similarly to tree-based models, like Approach0 and Tangent-s. If we use undirected graphs (i.e., add the reverse edges) to train the model, a higher partial bpref score would be achieved. The reason is that node embedding with an undirected graph will aggregate messages not only from children nodes but also from parent nodes. Undirected graphs weaken the "order" information of operations in math expressions and are not capable of preserving the structural differences but have more capabilities in capturing local structures.

6 Conclusion and Future Work

We propose a new mathematical formula embedding model based on graph neural networks in this paper. A kind of graph representation is designed to be generated from hierarchical representation for mathematical formula by merging the same part in the formula. Following the approach of graph self-supervised learning, we represent formulas in distributed dense vectors. The embedding model can be applied in the task of searching for mathematical formulas and achieve competitive full bpref and precision@k scores. The good performance in full relevance indicates a great potential of feature-based graph representation learning in capturing structural information of mathematical formulas. Moreover, our experiments also show that GCN may be a more suitable architecture for this task.

For future work, optimal self-supervised learning strategies at the level of entire graph and evaluation on ARQMath benchmark will be investigated. Considering the embedding effect of undirected graph representation, whether an ensemble of undirected and directed graph embeddings will produce better results is worth studying at a later stage.

Acknowledgements This work has been supported by National Natural Science Foundation of China (Grant No. 61702025) and State Key Laboratory of Software Development Environment.

References

1. Aizawa, A., Kohlhase, M., Ounis, I., Schubotz, M.: NTCIR-11 Math-2 task overview. In: Proceedings of the 11th NTCIR Conference on Evaluation of Information Access Technologies. pp. 88–98. National Institute of Informatics (2014)
2. Aizawa, A., Kohlhase, M.: Mathematical information retrieval. In: Evaluating Information Retrieval and Access Tasks: NTCIR's Legacy of Research Impact. pp. 169–185. Springer (2021)

3. Bengio, Y., Courville, A., Vincent, P.: Representation learning: A review and new perspectives. IEEE Transactions on Pattern Analysis and Machine Intelligence **35**(8), 1798–1828 (2013)
4. Chen, H.: Mathematical formula similarity comparing based on tree structure. In: Proceedings of the 12th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery. IEEE (2016)
5. Davenport, J.H., Kohlhase, M.: Unifying math ontologies: A tale of two standards. In: Proceedings of the 8th International Conference on Mathematical Knowledge Management. pp. 263–278. Springer (2009)
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. Association for Computing Machinery (2017)
7. Dewar, M.: OpenMath: An overview. SIGSAM Bull. **34**(2), 2–5 (2000)
8. 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 **abs/1707.05154** (2017)
9. Greiner-Petter, A., Schubotz, M., Cohl, H.S., Gipp, B.: Semantic preserving bijective mappings for expressions involving special functions between computer algebra systems and document preparation systems. Aslib Journal of Information Management **71**(3), 415–439 (2019)
10. Greiner-Petter, A., Youssef, A., Ruas, T., Miller, B.R., Schubotz, M., Aizawa, A., Gipp, B.: Math-word embedding in math search and semantic extraction. Scientometrics **125**(3), 3017–3046 (2020)
11. Hu, W., Liu, B., Gomes, J., Zitnik, M., Liang, P., Pande, V., Leskovec, J.: Strategies for pre-training graph neural networks. In: Proceedings of the 8th International Conference on Learning Representations (2020)
12. Kohlhase, M.: Using LaTex as a semantic markup format. Mathematics in Computer Science **2**(2), 279–304 (2008)
13. Kristianto, G.Y., Topic, G., Aizawa, A.: MCAT math retrieval system for NTCIR-12 mathir task. In: Proceedings of the 12th NTCIR Conference on Evaluation of Information Access Technologies. pp. 323–330. National Institute of Informatics (2016)
14. Krstovski, K., Blei, D.: Equation embeddings. ArXiv **abs/1803.09123** (2018)
15. 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. Association for Computing Machinery (2019)
16. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. In: Proceedings of the 1st International Conference on Learning Representations. pp. 1–12 (2013)
17. Miller, B.R., Youssef, A.: Technical aspects of the digital library of mathematical functions. Annals of Mathematics and Artificial Intelligence **38**(1), 121–136 (2003)
18. Pathak, A., Pakray, P., Das, R.: LSTM neural network based math information retrieval. In: Proceedings of the 2nd International Conference on Advanced Computational and Communication Paradigms. pp. 1–6 (2019)
19. Pavan Kumar, P., Agarwal, A., Bhagvati, C.: A structure based approach for mathematical expression retrieval. In: Proceedings of the 6th International Conference on Multi-disciplinary Trends in Artificial Intelligence. pp. 23–34. Springer (2012)

20. Pfahler, L., Morik, K.: Semantic search in millions of equations. In: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. pp. 135–143. Association for Computing Machinery (2020)
21. Sakai, T.: Alternatives to bpref. In: Proceedings of the 30th International ACM SIGIR Conference on Research and Development in Information Retrieval. pp. 71–78. Association for Computing Machinery (2007)
22. Schubotz, M., Greiner-Petter, A., Scharpf, P., Meuschke, N., Cohl, H.S., Gipp, B.: Improving the representation and conversion of mathematical formulae by considering their textual context. In: Proceedings of the 18th ACM/IEEE on Joint Conference on Digital Libraries. pp. 233–242. Association for Computing Machinery (2018)
23. Sojka, P., Líška, M.: The art of mathematics retrieval. In: Proceedings of the 11th ACM Symposium on Document Engineering. pp. 57–60. Association for Computing Machinery (2011)
24. Thanda, A., Agarwal, A., Singla, K., Prakash, A., Gupta, A.: A document retrieval system for math queries. In: Proceedings of the 12th NTCIR Conference on Evaluation of Information Access Technologies. pp. 346–353. National Institute of Informatics (2016)
25. Yasunaga, M., Lafferty, J.D.: TopicEq: A joint topic and mathematical equation model for scientific texts. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 33, pp. 7394–7401 (2019)
26. Zanibbi, R., Aizawa, A., Kohlhase, M., Ounis, I., Topic, G., Davila, K.: NTCIR-12 MathIR task overview. In: Proceedings of the 12th NTCIR Conference on Evaluation of Information Access Technologies. pp. 299–308. National Institute of Informatics (2016)
27. Zanibbi, R., Blostein, D.: Recognition and retrieval of mathematical expressions. International Journal of Document Analysis and Recognition **15**(4), 331–357 (2011)
28. Zanibbi, R., Davila, K., Kane, A., Tompa, F.W.: Multi-stage math formula search: Using appearance-based similarity metrics at scale. In: Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval. pp. 145–154. Association for Computing Machinery (2016)
29. Zanibbi, R., Oard, D.W., Agarwal, A., Mansouri, B.: Overview of ARQMath 2020: CLEF Lab on answer retrieval for questions on math. In: Experimental IR Meets Multilinguality, Multimodality, and Interaction. pp. 169–193. Springer (2020)
30. Zhong, W., Rohatgi, S., Wu, J., Giles, C.L., Zanibbi, R.: Accelerating substructure similarity search for formula retrieval. In: Proceedings of the 42nd European Conference on Information Retrieval. pp. 714–727. Springer (2020)
31. Zhong, W., Zanibbi, R.: Structural similarity search for formulas using leaf-root paths in operator subtrees. In: Proceedings of the 41st European Conference on Information Retrieval. pp. 116–129. Springer (2019)