Tangent-CFT: An Embedding Model for Mathematical Formulas

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

Shaurya Rohatgi
szr207@psu.edu
Penn State University
State College, Pennsylvania, USA

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

Jian Wu
jwu@cs.odu.edu
Old Dominion University
Norfolk, VA, USA

C. Lee Giles
clg20@psu.edu
Penn State University
State College, Pennsylvania, USA

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

ABSTRACT

When searching for mathematical content, accurate measures of formula similarity can help with tasks such as document ranking, query recommendation, and result set clustering. While there have been many attempts at embedding words and graphs, formula embedding is in its early stages. We introduce a new formula embedding model that we use with two hierarchical representations, (1) Symbol Layout Trees (SLTs) for appearance, and (2) Operator Trees (OPTs) for mathematical content. Following the approach of graph embeddings such as DeepWalk, we generate tuples representing paths between pairs of symbols depth-first, embed tuples using the fastText n-gram embedding model, and then represent an SLT or OPT by its average tuple embedding vector. We then combine SLT and OPT embeddings, leading to state-of-the-art results for the NTCIR-12 formula retrieval task. Our fine-grained holistic vector representations allow us to retrieve many more partially similar formulas than methods using structural matching in trees. Combining our embedding model with structural matching in the Approach0 formula search engine produces state-of-the-art results for both fully and partially relevant results on the NTCIR-12 benchmark. Source code for our system is publicly available.

KEYWORDS

Math Formula Retrieval, Formula Embeddings, Tree Embeddings

ACM Reference Format:
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 *The 2019 ACM SIGIR International Conference on the Theory of Information Retrieval (ICTIR '19), October 2–5, 2019, Santa Clara, CA, USA.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3341981.3344235

1 INTRODUCTION

While many information retrieval and natural language processing tasks benefit from distributed (i.e., dense vector) representations of words, embedding models to produce vector representations for mathematical formulas have not yet been as fully investigated. As with words, mathematical formulas can express similar content in different ways, thus making distributed representations that offer a level of abstraction potentially useful. While there has been a tremendous amount of research on continuous representations for words and paragraphs in Information Retrieval (IR) and Natural Language Processing (NLP), little work has been done on formula embeddings. This is unfortunate, as mathematics has a central role in scientific discourse.

Vector representations for formulas might be applied in a variety of ways. For instance, Mitra and Craswell [15] proposed a query auto-completion method for rare word prefixes using a convolutional latent semantic model. Many formulas are unique, and such a technique might be used for query auto-completion in math search. Formula vectors could also be used for retrieval using vector similarity measures, which we explore in this paper.

Mathematical notation also offers the potential for representing some types of semantic relationships in ways that words alone would lack, and these semantic relationships offer additional scope for the construction of embedding models. Formula representations are inherently hierarchical, either in the form of symbol layout trees (SLTs) that capture the placement and nesting of symbols on writing lines (e.g., in LaTeX), or as operator trees (OPTs), which capture the mathematical semantics of the application of operators to operands (e.g., in Content MathML: see Figure 1).

The primary goal of this work is to study whether embeddings are suitable for information retrieval in tasks involving formulas. For now, we consider only formula embeddings, without using surrounding text. This is natural for tasks including formula auto-completion and isolated formula search, but we are also aiming to obtain powerful embeddings that can be later enhanced through the incorporation of surrounding text [8, 18].

General embedding models such as DeepWalk [16] have been developed for graphs, including trees. These models are typically constructed by first traversing the graph in some way (e.g., breadth-first, depth-first, or random walk) to linearize the graph and then building an embedding model in the usual way. Current graph embedding models generally treat nodes as atomic units, but mathematical formulas exhibit more fine-grained structure (e.g., some

ICTIR '19, October 2–5, 2019, Santa Clara, CA, USA

ACM ISBN 978-1-4503-6881-0/19/10…$15.00
https://doi.org/10.1145/3341981.3344235

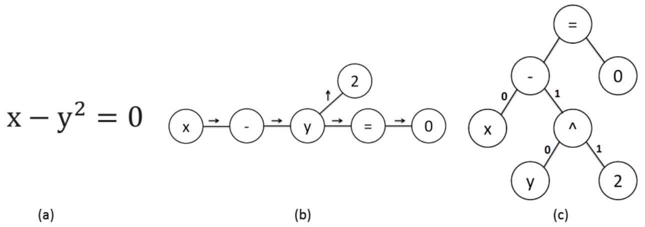

Figure 1: Formula (a) $x - y^2 = 0$ with associated (b) Symbol Layout Tree (SLT), and (c) Operator Tree (OPT). SLTs represent formula appearance by the placement of symbols on writing lines, while OPTs define the mathematical operations represented in expressions.

symbols are variables whereas others are constants) that could be leveraged to produce improved embedding models. Many mathematical formulas are also rare or unique in a collection [12], and basing the vector representation on primitives smaller than nodes can provide more robust representations for unseen and unique formulas.

In this paper, we present an embedding model for mathematical formulas using SLT and OPT representations. We use fastText [2] to produce the distributed representations based on a linearization and tokenization of mathematical formulas. Formula similarity is then computed as the cosine of the average of the distributed representation vectors for the components of an equation. Our proposed method outperforms current methods in partial matching and also has the highest harmonic mean of partial and full relevance measures. We also analyze our embeddings using visualizations that illustrate how formulas are clustered in two-dimensional space. Finally, we show that our formula vectors and a retrieval model based on paths in operator trees (Approach0 [22]) may be combined to improve both partial and full relevance in retrieval results.

The remainder of this paper is organized as follows. We first review related work on mathematical information retrieval and formula embeddings in Section 2. In Section 3, we then present our formula embedding technique. Section 4 presents both quantitative and qualitative results from the evaluation of our embedding model. Finally, in Section 5 we conclude the paper and provide a few remarks about next steps.

2 RELATED WORK

We first introduce how mathematical formulas are represented in documents. We then provide a brief overview of formula recognition models used in mathematical information retrieval (MIR) systems. We describe the fastText embedding model used in our system, and then end with a discussion of existing formula embedding models.

Math Formula Representation. Math encodings are naturally hierarchical, defining formula appearance in symbol layout tree (SLT) encodings such as LaTeX, or formula semantics in operator tree (OPT) encodings such as Content MathML. SLTs represent the placement of symbols on baselines along with their spatial arrangement [20]. Although SLT representations are mostly used online, they can be ambiguous. For instance, a symbol can be a variable in one context, and an operator in another. In contrast, OPTs are mathematically unambiguous, although constructing a valid OPT from an SLT requires a priori knowledge of argument types along with operator syntax, precedence, and associativity (e.g., as represented in an expression grammar). Figure 1 shows the SLT(b) and OPT(c) representation of formula $x - y^2 = 0$.

Nodes. We make use of the SLT and OPT formula representations produced by the Tangent-s formula search engine [4]. Here nodes in SLTs and OPTs represent individual symbols and explicit aggregates such as function arguments in the form: (**Type**!value). More precisely, nodes can be numbers (**N!**n), identifiers such as variable names (**V!**v), text fragments, such as 'lim' and 'such that' (**T!**t), fractions (**F!**), radicals (**R!**), explicitly specified whitespace (**W!**) and finally, matrices, tabular structures, and parenthesized expressions (**M!**f rxc); with f showing fence characters such as parentheses, r the number of rows, and c the number of columns. In our SLTs, operators do not have a type attribute but in our OPTs, commutative and non-commutative operators have type (**U!**) and (**O!**) respectively.

SLT Edges. Considering an object O, in an SLT there are seven types of edge labels representing the spatial relationships between symbols (nodes): **next** ('n') that references the adjacent object appearing to the right of O and on the same line, **within** ('w') references the radicand if O is a root or the first element appearing in row-major order in O if it is a structure represented by M!, **element** ('e') references the next element appearing after O in row-major order inside a structure represented by M!, **above** ('a') references the leftmost object on a higher line starting at the position above O, **below** ('b') references the leftmost object on a lower line starting at the position below O, **pre-above** ('SUP') references the leftmost object of a prescripted superscript of O and **pre-below** ('SUB') references the leftmost object of a prescripted subscript of O.

OPT Edges. Edge labels in OPTs indicate argument position. For commutative operators where order does not affect the result (e.g., '+') argument edges are unlabelled (in practice, all labeled '0'). For non-commutative operators, arguments are indexed from 0 (see Figure 1(c)).

Math Information Retrieval (MIR) Systems. Mathematical information retrieval is motivated by the vast number of technical, scientific, and educational documents containing mathematics that current general-purpose search engines do not index directly [20]. In general search engines such as Google, the user may include a formula in the query using LaTeX or MathML tags, which are treated the same as words. In MIR systems formulas are indexed explicitly, and the search interface provides a formula editor to support the creation of queries containing formulas and text.

For formula retrieval, most previous approaches may be categorized as *text-based* or *tree-based* models [21]. In text-based models, formulas are converted to a string such as LaTeX and then the same search method is used for both formulas and text. For instance, Sojka and Liska [17] proposed the MIaS system based on term frequency-inverse document frequency (TF-IDF) for indexing XHTML documents containing MathML expressions. Kumar et al. [9], in turn, proposed another approach for retrieving formula LaTeX strings using the largest common sub-string between the query formula and each indexed expression, which requires a quadratic algorithm. Text-based approaches lose the hierarchical nature of formulas, and may fail to characterize formula structure well as a result.

In tree-based models, formulas are represented directly as trees, often with sub-trees to support partial matching. Tree-based formula retrieval approaches were found to be more effective than text-based approaches in both the NTCIR-11 [1] and NTCIR-12 [19] formula retrieval benchmarks. Kristianto et al. [7] proposed the MCAT system that encodes path and sibling information from MathML Presentation (SLT-based) and Content (OPT-based) representations, where paths act as the retrieval units. They also make use of a hashing-based formula structure encoding scheme, and textual information at three levels of granularity. Approach0 [22] retrieves formulas using only paths from operator trees generated by parsing LaTeX with a relatively small expression grammar. Candidates are scored based on up to three best-matching subtrees.

Like MCAT, Tangent-s [4] combines retrieval over both SLTs and OPTs. Candidates are first retrieved and scored using tuples representing relative paths between pairs of symbols. The top-k candidates are then aligned with the query to produce formula similarity scores (the Maximum Subtree Similarity, MSS). SLT results and OPT results are then combined via linear regression over alignment metrics from each representation to produce final similarity scores. Recently, Fraser et al. [5] adapted the BM25 text retrieval model to work with an extended set of Tangent tuples capturing formula structure, using these as search terms for ranking. This approach is a hybrid of text-based and tree-based retrieval models.

FastText Embedding Model. Compared to tree-based approaches, we believe an embedding model can learn more abstract formula representations by being less dependent upon *specific* paths and/or sub-trees, and therefore can provide better approximations for retrieval. In particular, we believe that applying an n-gram embedding will be beneficial, as mathematical formulas are often unique [12]. For tasks such as query suggestion, users may insert unseen or even invalid formulas as input queries, making models such as word2vec less appropriate, as they cannot handle unseen formulas. Word2vec considers words as the smallest unit and if any word is not seen in the training collection, there is no embedding for it.

FastText is an n-gram embedding model [2] derived from the word2vec model [14]. In word2vec, individual words are the smallest unit for vector representations, and internal word structure is ignored. FastText generalizes word2vec to accommodate word morphology, providing usable vector representations for rare and unseen words. As illustrated in [2], given a dictionary of n-grams with size G, for a word w that has set of n-grams ζ_w, a vector representation of word w is the sum of the vector representations of its n-grams, and a scoring function which maps pairs of (word, context) will be calculated as:

$$s(w, c) = \sum_{g \in \zeta_w} z_g^{\intercal} v_c .$$

where, z_g is vector representation for the n-gram g and v_c is the vector representation of the word appearing in the context window of word w. FastText represents words as bags of n-grams, with n ranging from 1 to the length of the word. For instance, considering n-grams, $n = [3, 6]$, the word '*system*' (with no boundary symbols) is presented as *sys*, *yst*, *ste*, *tem*, *syst*, *yste*, *stem*, *syste*, *ystem* and *system*. In this model, the minimum and maximum number of n-grams for training is a hyper-parameter. FastText uses the mean of the target word vector and its component n-gram vectors for training the model at each step. The vectors that form the target are updated uniformly based on the calculated error. After generating vectors for n-grams, the vector representation for a word is the sum of its embedded n-gram vectors.

Math Formula Embedding. Unlike development of word embeddings, there have been few attempts to embed formulas. Early research on formula embedding was carried out by Thanda et al. [18] where a variant of the doc2vec algorithm, the distributed bag of words (PV-DBOW) [10] was introduced. They use binary expression trees and assigned each formula a real-valued vector such that formulas with similar structure are close to each other in vector space. Gao et al. [6] introduced embedding for both symbols (symbol2vec) and formulas (formula2vec). Symbol2vec was based on a Continuous Bags-of-Words (CBOW) architecture using negative sampling [13], while formula2vec uses a distributed memory model of paragraph vectors [10]. The proposed method by Krstovski and Blei [8] generates embeddings for both words and equations with a larger context window size for equations than words. They also proposed equation unit embedding, treating equations as sentences where the words are symbols, variables and operators, referred to as a unit.

To the best of our knowledge, previous work has not focused upon embedding isolated formulas, which considers only formula structure and similarities between symbols and operations. Embeddings for individual formulas allow the creation of two separate embeddings, one for formulas, and another for surrounding text. Another shortcoming is that existing formula embeddings are defined at the formula ('word') level, meaning that there is no embedding for unseen formulas. In math search, users may submit queries containing formulas that do not exist in a collection. Based on this property, a 'sub-word' approach for formula embeddings may produce a more robust vector representation.

3 EMBEDDING MODEL

Figure 2 provides an overview of our embedding model, Tangent-CFT (Tangent Combined with fastText). To embed mathematical formulas, we first linearize mathematical formulas and then apply a text embedding model. This is a common approach in embedding techniques. For instance, DeepWalk [16] which is an approach for graph embedding, applies random walks on the graph to get sequence of nodes and then applies word2Vec to get a vector representation for each node. After linearizing formulas, we applied an n-gram embedding model, fastText [2], to embed the formula. Our code is available online.[1]

3.1 Tuple Sequence Generation

Mathematical formulas are usually represented in LaTeX or MathML format. Using Tangent-s [4, 21], these are converted into SLT and OPT encodings, and a depth-first traversal is used to generate a tuple sequence. Tangent-s generates tuples for pairs of symbols and their relative positions using tuples in form of (s1, s2, R, FRP) after traversing the tree depth first. S1 is the ancestor symbol, s2 the descendant symbol, R the edge label sequence from s1 to s2,

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

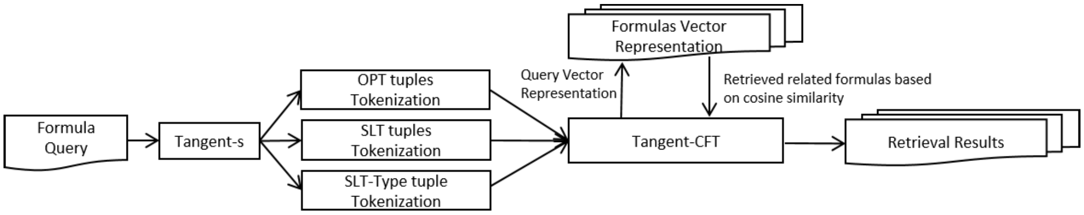
Formulas Vector Representation
Query Vector Representation
Retrieved related formulas based on cosine similarity
Formula Query
Tangent-s
OPT tuples Tokenization
SLT tuples Tokenization
SLT-Type tuple Tokenization
Tangent-CFT
Retrieval Results

Figure 2: Overview of Tangent-CFT. A formula query is converted to an SLT and OPT using Tangent-s. We convert trees to symbol tuple lists using depth-first traversals, and then use fastText to learn tuple embeddings separately for each representation. Formulas are represented by averaging tuple vectors. These formula vectors may be combined across OPT, SLT, and SLT-Type tuple inputs using a simple vector sum.

Table 1: SLT and OPT Tuples for Formula in Figure 1: $x - y^2 \; = 0$

SLT tuples	**OPT tuples (FRP omitted)**
(**V!** x, -, n, -)	(**U!** eq,**O!** minus,0)
(-, **V!** y, n, n)	(**O!** minus, **V!** x, 0)
(**V!** y, **N!** 2, a, nn)	(**V!** x, eob, 0)
(**N!** 2, eob, n, nna)	(**O!** minus, **O!** SUP, 1)
(**V!** y, =, n, nn)	(**O!** SUP, **V!** y, 0)
(=, **N!** 0, n, nnn)	(**V!** y, eob, 0)
(**N!** 0, eob, n, nnnn)	(**O!** SUP, **N!** 2, 0)
	(**N!** 2, eob, 0)
	(**U!** eq, **N!** 0, 1)
	(**N!** 0, eob, 0)

and FRP the full relative path giving the location of s1 on its path from the root of the tree. A parameter w (window size), controls the maximum path length between symbols. In our model, the window size is set to 2. If the End-of-Baseline (EOB) flag is set, the system creates dummy pairs between the last symbol on each baseline and null, to help with matching small expressions. In OPTs, this EOB tuple really represents the end of root-leaf paths, and is generated for all arguments (i.e., at leaves of the OPT). The SLT and OPT tuples for Figure 1 are shown in Table 1.

3.2 Tokenizing Formula Tuples

After converting the formula tree representation to a tuple sequence, each tuple can be considered as a word. To use an n-gram embedding model we need to define our 'words' and 'characters.' In our representation, each character (token) is encoded using a unique identifier.

Tokenizing the tuple elements can provide good insight into the formula structure. For instance, separating node type and value provides more details for formula structure, allowing two formulas sharing the same structure but different variables or constants to obtain a higher similarity score than they would with untokenized tuples (i.e., if tuples are treated as characters).

For example, we checked whether tokenizing numbers using their individual digits vs. complete values would help retrieval, we did a small experiment using four constants as queries using their first 10 decimal places: Pi (π, 3.1415926535), Euler's number (e, 2.7182818284), the golden ratio (1.6180339887), and $\sqrt{2}$ (1.4142135623). For each constant, we observed the number of digits in retrieved formulas matching those to the right of the decimal point in the query. As expected, tokenizing numbers greatly improved retrieval for variations of each constant with differing levels of precision. Without tokenizing digits, only exact matches to the given constant query and seemingly random numbers are retrieved. We extend this approach to matrices - for example, consider three matrices of sizes 2 × 3, 2 × 4, and 5 × 6. If we do not tokenize the dimension numbers separately, the tokenized SLT elements are three unique tokens: ('M!', '[] 2 × 3') , ('M!', '[] 2 × 4') and ('M!', '[] 5 × 6'). In this 'character' representation, no two of these tokens will match. However, the first two matrices in our example seem to be more similar as they have the same number of rows.

Another point to consider is whether to use the Full Relative Path (FRP). This may provide unnecessary information which can be misleading when doing embedding, especially when using the OPT representation, as it is simply a sequence of argument positions for each operation. Therefore, as the input of the embedding model, we used tokenized formula tuples without the FRP.

To provide better insight on how tokenization and enumeration of tokens is performed, we again use our example formula from Figure 1, $x - y^2 = 0$. As shown in Table 1, the first OPT tuple to be encoded is (U!eq,O!minus,0). This tuple can be tokenized into 'U!', 'eq', 'O!', 'minus' and '0'. Each of these tokens gets a unique Id. The second tuple is (O!minus,V!x,0) which is tokenized to 'O!', 'minus', 'V!', 'x' and '0'. Previously seen tokens such as 'U!' will use their previous Id, but new tokens such as 'x' will get a new Id. Note that, token identifiers used for edges and nodes are distinct, to ensure that these two path types produce distinct n-grams.

3.3 Applying fastText to Formulas

After linearizing formulas, we apply the fastText [2] n-gram embedding model to embed the formula. Each encoded tuple is considered as a word and the context window for a tuple *'T'* is defined by nearby tuples in the linearized tuple sequence. The context window size is also a hyper-parameter in the model, which is set to the default value, 5. After the model is trained, each tuple is assigned a d dimensional vector (which is defined before training). The vector representation for formula F with set of n tuples, T_F, is given by:

$$formulaVec\,(F) = \frac{1}{n} \sum_{t \in T_F} tupleVec(t)$$

To compute the the similarity of two formulas, we use the cosine similarity of their vector representations. First the vector representation for each formula in the collection is created and then for a given formula query, based on the trained model, the vector representation is obtained. Then, the similarity between the formula query q with vector V_q and a formula f in the collection with vector V_f is measured as follows:

$$sim\,(q, f) = \frac{V_f \cdot V_q}{\left|V_f\right|\left|V_q\right|}.$$

A higher cosine similarity results in a better retrieval rank.

3.4 Combining SLT and OPT Embeddings

Our embedding models for mathematical formulas are based on their OPT and SLT representations, where the OPT captures the formula semantics, while the SLT focuses on the visual structure of the formula. Therefore, combining the OPT and SLT vector representations may allow us to capture both the semantics and the appearance of formula in a single vector.

In addition to SLT and OPT vector representations, we trained another model that we call SLT-Type from the SLT representation that only leverages the node types and ignores the values. For instance, the tuple (V!x,N!2,a) will be considered as (V,N,a). This might be useful in cases where the name of variables in two formulas are different, but they still have same structure. To achieve the final vector representation of a formula, we sum up the three vector representations: SLT, SLT-Type and OPT.

4 EVALUATION

We now evaluate our formula embeddings using the NTCIR-12 formula browsing task. We then compare our system against state-of-the-art methods, provide an analysis of retrieval errors, and examine the effect of the main embedding parameters. Finally, we present new state-of-the-art results for the NTCIR-12 task using a model that combines the Tangent-CFT formula embeddings with a tree-based formula search engine (Approach0 [22]).

4.1 Dataset and Evaluation Measures

For evaluation, we relied on the NTCIR-12 MathIR Wikipedia Formula Browsing Task [19], which contains over 590,000 mathematical formulas from English Wikipedia. In this task, there are 20 concrete queries (i.e., formulas without wildcards). During the task, two human assessors evaluated the pooled hits from participating systems. Each assessor scores a hit with score 0, 1 or 2 based on relevance (with 2 indicating highly relevant). The final relevance rating is the sum of the two assessor scores, producing a score between 0 and 4. Scores of 3 or 4 are considered fully relevant, scores of 1 or 2 are considered partially relevant, and a score of 0 is considered irrelevant.

To compare our method with previous approaches we used bpref [3] on top-1000 results. For a query with f relevant formulas, bpref locates the first d judged non-relevant formulas, then compares their ranks against the ranks of the f judged relevant formulas pairwise. The harmonic mean of full and partial bpref quantify how well a system can retrieve both fully (F) and partially (P) relevant formulas ($2PF/(P + F)$).

4.2 Retrieval Results

Training. We used grid search to tune the hyper-parameters for our embedding models. To train our models, we used a skip-gram model with 20 negative samples, vector dimension of 300, and context window size of 5. For both SLT and OPT models, we used n-grams of lengths between 3 and 6, and for the SLT model with only node types (SLT-Type), we simply applied word2vec. We found that it is important to include longer n-grams, specifically with size 6, which can improve both partial and full bpref scores. More details on training the embedding models are provided in Section 4.4.

Table 2: NTCIR-12 Results (Avg. bpref@1000)

System	bpref Partial	bpref Full	Harmonic Mean
Tangent-CFT	**0.71**	0.60	**0.65**
Tangent-OPT-FT	0.66	0.60	0.63
Tangent-sLT-FT	0.66	0.58	0.61
Approach0 [22]	0.59	**0.67**	0.63
Tangent-s [4]	0.59	0.64	0.61
MCAT [7]	0.57	0.57	0.57

Table 3: Tangent-CFT Results for Query:
$x - 1 - \frac{1}{2} - \frac{1}{4} - \frac{1}{5} - \frac{1}{6} - \frac{1}{9} - \cdots = 1$

Rank	Formula
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	$\frac{\pi}{4} = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \cdots$

NTCIR-12 Results. Table 2 compares the results of formula retrieval by different systems using partial and full bpref scores along with their harmonic mean. Note that all models in Table 2 support unification of symbols based on predefined types (e.g., for variables and numbers). An example of top-5 Tangent-CFT formula retrieval results is shown in Table 3; this query had the highest harmonic mean bpref over all queries (0.931).

As Table 2 indicates, Tangent-CFT obtains the highest harmonic mean bpref score; achieving the highest partial bpref score, and a full bpref higher than MCAT, but 4-7% lower than Approach0 and Tangent-s. These may be explained by Tangent-CFT using cosine similarity over formula vectors rather than direct comparison of formula trees (SLTs and/or OPTs). In contrast, Approach0 finds matching OPT sub-trees from leaf-root paths, and Tangent-s finds matching SLT and OPT sub-trees from the tuples used to train Tangent-CFT. This allows Approach0 and Tangent-s to identify highly similar formulas more consistently than the more abstract vector representation in Tangent-CFT. However, our vector representation still useful structural information, but in a less strict form, leading to the stronger partially relevant bpref result (12-14% higher than Approach0, Tangent-s, and MCAT).

Comparison of Embeddings vs. Structural Matching. We compared our results with the state-of-the-art tree-based model, Approach0. Consider the formula $O(mn \log m)$ (query number 8 in NTCIR-12). For this query, the top-5 results from Tangent-CFT and Approach0 systems are shown in Table 4. For this particular query, only the partial bpref score of Tangent-CFT is higher than Approach0, and Approach0 does better on full bpref score. Both

Table 4: **Tangent-CFT vs. Approach0 Results for Query:** $O(mn \log m)$

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

systems retrieve the exact match as the first formula. Tangent-CFT retrieves formulas that contain parts of the query formula (with m and n as the variables in all cases), while Approach0 retrieves formulas that are identical to the query after variable/subexpression substitution. Analysis of the top-1000 results for each system shows that formulas judged to be fully relevant such as $O(VE \log V)$ and $O(KN \log N)$ retrieved by Approach0 are not retrieved by Tangent-CFT. On the other hand, Tangent-CFT was able to retrieve formulas such as $O(\log n)$ and $O(mr)$ that were judged to be partially relevant as well as some variants judged to be fully relevant such as $O(\lambda_\delta(n) \log n)$ that Approach0 was not able to retrieve due to differences in the OPT leaf-root path structures.

The NTCIR-12 collection also contains queries such as "$0 \rightarrow G^\wedge \xrightarrow{\pi^\wedge} X^\wedge \xrightarrow{\iota^\wedge} H^\wedge \rightarrow 0$" for which Tangent-CFT did better in both partial and full bpref than Approach0. For that query, among the top-1000 retrieved formulas, Tangent-CFT was able to retrieve formulas such as "$1 \rightarrow K \xrightarrow{i} G \xrightarrow{\pi} H \rightarrow 1$" and "$W \xrightarrow{f} X \xrightarrow{g} Y \xrightarrow{h} Z$" that Approach0 could not, possibly due to constraints in the Approach0 expression grammar. As another example, for the query:

$$\nabla \times \mathbf{B} = \mu_0 \mathbf{J} + \underbrace{\mu_0 \epsilon_0 \frac{\partial}{\partial t} \mathbf{E}}_{\text{Maxwell's term}}$$

Tangent-CFT was able to retrieve two formulas, "$\nabla \times \mathbf{B} = \mu_0 (\mathbf{J} + \epsilon_0 \frac{\partial \mathrm{E}}{\partial t})$" and "$\nabla \times \mathbf{B} - \frac{1}{c^2} \frac{\partial \mathrm{E}}{\partial t} = \mu_0 \mathbf{J}$" that were judged as fully relevant but that were not retrieved by Approach0. Comparing these two formulas with the query formula, the benefit of the n-gram embeddings can be seen, as the two formulas are quite similar to the query, with some of the characters being replaced or eliminated.

In contrast to this, Approach0 was able to achieve better partial and full bpref scores for some queries. Consider the query:

$$P_i^x = \frac{N!}{n_x!(N - n_x)!} p_x^{n_x} (1 - p_x)^{N - n_x}$$

Approach0 was able to retrieve formulas such as $\left(\begin{array}{c} N \\ N_R \end{array} \right) = \frac{N!}{N_R!(N - N_R)!}$ and $f(k) = \binom{n}{k} p^k (1 - p)^{n-k}$ that were judged partially relevant, while Tangent-CFT was not able to do so, perhaps because of differences in the symbols, structure, and operations involved.

Studying the extreme cases, the largest difference when scoring by partial relevance was for the following query formula (where partial bpref scores for Approach0 and Tangent-CFT were 0.21 and 0.73, respectively):

$$N = \left\lfloor 0.5 - \log_2 \left(\frac{\text{Frequency of this item}}{\text{Frequency of most common item}} \right) \right\rfloor$$

For this query, Approach0 was not able to retrieve 24 of the judged (partially or fully) relevant formulas that were retrieved by Tangent-CFT in the top-1000 results, including formulas such as "$L_N = 40 + 10 \log_2(N)$" and "$d = 69 + 12 \log_2 \left(\frac{f}{440 \text{ Hz}} \right).$"

Considering full relevance, the largest difference between the two systems were for the following query (where the full bpref scores for Approach0 and Tangent-CFT were 0.83 and 0.17, respectively):

$$\cos \alpha = -\cos \beta \cos \gamma + \sin \beta \sin \gamma \cosh \frac{a}{k},$$

For this query, Tangent-CFT had all relevant formulas in its result set, however the ranking of hits is weaker than in Approach0. In the top-10 results Tangent-CFT misses 4 fully relevant formulas found in that range by Approach0, including "$\cos A = -\cos B \cos C + \sin B \sin C \cosh a.$" and "$\cos C = -\cos A \cos B + \sin A \sin B \cosh c,$". Therefore, a re-ranking step might further improve the initial results of Tangent-CFT.

SLT vs. OPT Embeddings. Next, we study how SLT and OPT embedding models differ. To compare these embeddings, we provide two examples from our experiments for which one method outperforms the other. For this query:

$$r_{xy} = \frac{\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})}{(n-1) s_x s_y} = \frac{\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n} (x_i - \bar{x})^2 \sum_{i=1}^{n} (y_i - \bar{y})^2}},$$

the SLT model was able to retrieve formulas such as the one below (which was judged to be fully relevant), whereas the OPT model was not able to retrieve this formula, which shares some aspects of the appearance, but in our view not the semantics, of the query formula.

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

On the other hand, OPT was able to do better than SLT on some formulas. For instance, for the query $O(mn \log m)$ (see Table 4) the OPT model was able to retrieve formulas such as $O(nk \log k)$ whereas the SLT model cannot. For that query, the SLT model gave higher rank to formulas such as $O(m + \log n)$ that are more similar in appearance to the formula query.

It should be mentioned that our 'SLT-Type' embedding only improved partial and full bpref scores by 0.01 on average over all queries. The effect of using this embedding should be explored with a larger number of queries, especially queries for which exact matches do not exist in the collection (as they do for all of the NTCIR-12 queries). However, our initial intuition that our SLT-Type embedding might help has some support. For the formula query

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

Tangent-CFT was able to retrieve the following formula (which was judged to be fully relevant), that was not retrieved when using

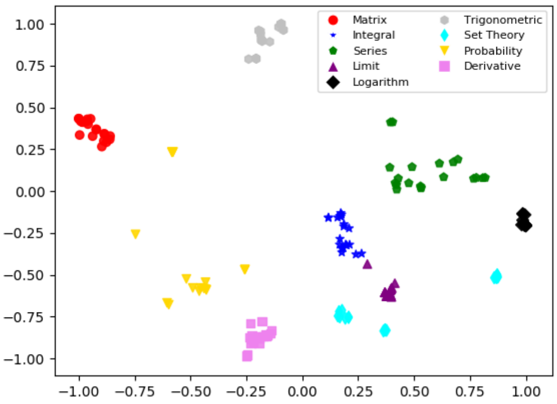

Figure 3: Formula embedding visualization. Each point is a formula in 2-D space obtained using the t-SNE package.

only the SLT and OPT combined model:

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

4.3 Visual Exploration of Embedding Similarities

To better understand how well our model maps related formulas to points that are close in the embedding space, we applied t-Distributed Stochastic Neighbor Embedding (t-SNE) [11] for dimensionality reduction, mapping each of our 300-dimensional embeddings to a two-dimensional space. We selected 180 formulas belonging to 9 categories of mathematical formulas: Matrix, Integral, Summation, Limit, Logarithm, Trigonometric, Set theory, Probability and Derivative. Each category contains 20 different formulas of that type. Providing examples, "$\lim_{x\to c}[f(x)g(x)] = L_1 \times L_2$" is labeled as Limit, while "$\int_{-\infty}^{+\infty} e^{-x^2}\,\mathrm{dx} = \sqrt{\pi}$" is label as Integral. This dataset is publicly available.[2]

Figure 3 shows a visualization of these 180 formulas, with axis labels normalized to be between -1 and 1. As can be seen, most formulas from the same category are mapped to similar positions. The formulas in the Set and Probability categories are spread more widely over the space, compared to other categories. This might be due to the fact that these categories could have been divided into more fine grained categories. For instance, in the Set category, the formulas located on the right side mostly include Logic operations such as $\exists$ or $\forall$, while the formulas on the left side mostly include set operations such as union or intersection. Another point to notice is that the formula "$\lim_{z\to a^+}\int_z^b f(x)\,dx$," which might have been categorized either as Integral or Limit, is the purple triangle near the blue stars.

4.4 Effect of Embedding Parameters

In this section, we examine the effect of model parameters on our bpref evaluation measure. To study the effect of n-gram length we kept other parameters constant (using a skip-gram model with 20 negative samples and a context window size of 5). Table 5 and 6 show the effect of n-gram length on SLT and OPT models, respectively. As these results indicate, our models are relatively insensitive to n-gram length, but increasing n-gram lengths do seem to help a bit for both partial and full bpref scores. This suggests that some aspects of mathematical formula similarity are evidenced in longer sequences.

Table 5: Effect of n-gram size on (Partial, Full) bpref for SLTs. Min length is shown across rows; max length is shown across columns.

	2	3	4	5	6
1	(0.63,0.56)	(0.64,0.56)	(0.65,0.56)	(0.65,0.56)	(0.65,0.56)
2		(0.65,0.56)	(0.65,0.56)	(0.65,0.56)	(0.65,0.57)
3			(0.66,0.57)	(0.65,0.57)	**(0.66,0.58)**
4				(0.63,0.58)	(0.64,0.58)

Table 6: Effect of n-gram size on (Partial, Full) bpref for OPTs. Min length is shown across rows; max length is shown across columns.

	2	3	4	5	6
1	(0.62,0.58)	(0.63,0.58)	(0.63,0.58)	(0.63,0.59)	(0.63,0.59)
2		(0.63,0.58)	(0.63,0.57)	(0.63,0.58)	(0.64,0.59)
3			(0.63,0.57)	(0.63,0.59)	**(0.66,0.60)**
4				(0.59,0.61)	(0.60,0.61)

Table 7: Effect of embedding vector length on (Partial, Full) bpref.

	200	250	300	350
SLT	(0.61,0.55)	(0.65,0.56)	**(0.66,0.58)**	(0.63,0.56)
OPT	(0.63,0.59)	(0.64,0.60)	**(0.66,0.60)**	(0.63,0.55)

Table 8: Effect of architecture (Skip-Gram (SG) and Continuous Bag of Words (CBOW)) vs. training algorithm (Hierarchical Softmax (HS) and Negative Sampling (NS)) on harmonic mean bpref.

	HS	NS(15)	NS(20)	NS(25)
SLT (SG)	0.58	0.60	0.61	0.61
SLT (CBOW)	0.49	0.50	0.51	0.50
OPT (SG)	0.60	0.60	**0.63**	0.62
OPT (CBOW)	0.55	0.56	0.56	0.55

Next, we investigated the effect of vector size. We considered dimensions of length 200, 250, 300 and 350, keeping other parameters fixed. As Table 7 shows, 300 dimensions, a number often used for text, seems like a good choice.

Finally, we studied how different embedding architectures affect our model. We consider skip-gram (SG) and continuous bag of words (CBOW) architectures with different training algorithms, hierarchical softmax (HS) and negative sampling (NS). Table 8 shows the results of these experiments. As can be seen, the skip-gram model achieved better results. The skip-gram model learns to predict their context of a target word, while the CBOW model predicts the target word according to its context. The context is represented as a bag of the words contained in a fixed size window around the target word. In CBOW the vectors from the context words are averaged before predicting the target word, and therefore rare words will be smoothed over a lot of examples with more frequent words. In the skip-gram model there is no averaging of embedding vectors, and the model can learn better representations for rare words when their vectors are not averaged with the other context words in the process of making predictions.

[2] https://github.com/BehroozMansouri/TangentCFT

Table 9: NTCIR-12 Results after Combining Approach0 and Tangent-CFT (TanApp) vs. Individual Retrieval Models.

Retrieval result	Partial bpref	Full bpref	Harm. Mean bpref
TanApp	**0.73**	**0.70**	**0.71**
Tangent-CFT	0.71	0.60	0.65
Approach0	0.59	0.67	0.63
Tangent-s	0.59	0.64	0.61

4.5 Combining Tangent-CFT with Approach0

Relying only on embeddings for formula retrieval may lead to higher partial relevance scores, but perhaps at the cost of lower full relevance scores when compared to operating directly on a formula tree. For example, Approach0 retrieves formulas using leaf-root paths in OPTs, and obtains higher full relevance scores than Tangent-CFT (see Table 2).

To try and leverage the strengths of both the embedding-based and tree-based approaches, we created another model (*TanApp*) that linearly combines retrieval scores from Tangent-CFT and Approach0. Given weight parameter $\alpha \in [0, 1]$, TanApp calculates the score for a given query 'q' as follows:

$$Score_q(f) = \alpha \cdot Tangent\text{-}CFT_q(f) + (1 - \alpha) \cdot Approach0_q(f)$$

We used a grid search over alpha to find the weight that maximizes one of three possible target measures: (1) the average partial bpref, (2) the average full bpref, or (3) the average harmonic mean of full and partial bpref using leave-one-out cross-validation. The first row of Table 9 shows the TanApp results optimized to each measure (i.e., three separate grid-searches optimized for partial, full, and harmonic mean of bpref), along with the corresponding results for each individual system. For comparison, Tangent-s results are also shown. As can be seen, TanApp is the best choice, regardless of the chosen evaluation measure.

5 CONCLUSION

In this paper, we presented Tangent-CFT, an embedding model for mathematical formulas. We use fastText to produce formula embeddings for both symbol layout trees (SLTs) that capture formula structure, and operator trees (OPTs) that capture formula semantics. The embedding procedure converts a tree-based formula representation into a sequence of tuples. Each tuple is treated as a word, with its tokenized elements treated as characters. Tuple 'words' are then embedded using n-grams of varying lengths computed over the tuple and its neighboring tuples in the sequence.

Our Tangent-CFT model combines OPT, SLT, and SLT-Type embeddings to obtain higher partial relevance than state-of-the-art models for the NTCIR-12 formula browsing task. We have also shown that combining results from an embedding model such as Tangent-CFT with results from a structure matching approach (e.g., Approach0) can produce higher partial *and* full relevance scores than previous approaches.

For future work, we plan to extend the existing test collection to include more diverse query formulas, and particularly formulas that are not present as exact matches in the collection. Also, we plan to incorporate text near formulas into our embedding model. So far we have only studied isolated formula retrieval, and we expect that leveraging nearby text would further improve the representation, as was observed in the NTCIR MathIR task [19].

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

REFERENCES

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

[2] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics.*

[3] Chris Buckley and Ellen M Voorhees. 2004. Retrieval evaluation with incomplete information. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval.*

[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.*

[5] Dallas Fraser, Andrew Kane, and Frank Wm Tompa. 2018. Choosing Math Features for BM25 Ranking with Tangent-L. In *Proceedings of the ACM Symposium on Document Engineering 2018.*

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

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

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

[9] P Pavan Kumar, Arun Agarwal, and Chakravarthy Bhagvati. 2012. A structure based approach for mathematical expression retrieval. In *International Workshop on Multi-disciplinary Trends in Artificial Intelligence.*

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

[11] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of Machine Learning Research.*

[12] Behrooz Mansouri, Douglas W. Oard, and Richard Zanibbi. 2019. Characterizing Searches for Mathematical Concepts. In *Joint Conference on Digital Libraries.*

[13] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space.

[14] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems.*

[15] Bhaskar Mitra and Nick Craswell. 2015. Query auto-completion for rare prefixes. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management.*

[16] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.* ACM.

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

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

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

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

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

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