

E-Gen: Leveraging E-Graphs to Improve Continuous Representations of Symbolic Expressions

Hongbo Zheng* Suyuan Wang* Neeraj Gangwar Nickvash Kani

University of Illinois at Urbana-Champaign

{hongboz2, suyuan2, gangwar2, kani}@illinois.edu

Abstract

Vector representations have been pivotal in advancing natural language processing (NLP), with prior research focusing on embedding techniques for mathematical expressions using mathematically equivalent formulations. While effective, these approaches are constrained by the size and diversity of training data. In this work, we address these limitations by introducing E-Gen, a novel e-graph-based dataset generation scheme that synthesizes large and diverse mathematical expression datasets, surpassing prior methods in size and operator variety. Leveraging this dataset, we train embedding models using two strategies: (1) generating mathematically equivalent expressions, and (2) contrastive learning to explicitly group equivalent expressions. We evaluate these embeddings on both in-distribution and out-of-distribution mathematical language processing tasks, comparing them against prior methods. Finally, we demonstrate that our embedding-based approach outperforms state-of-the-art large language models (LLMs) on several tasks, underscoring the necessity of optimizing embedding methods for the mathematical data modality.

1 Introduction

While large language models (LLMs) (Devlin, 2018; Radford, 2018; Radford et al., 2019; Rae et al., 2021; Chowdhery et al., 2023; Achiam et al., 2023) have demonstrated effectiveness in processing natural language, these methods still perform suboptimally with math-based content which plays an important role across numerous domains (Zanibbi et al., 2024; Rohatgi et al., 2019). For instance, even a state-of-the-art LLM like GPT-4V only reaches 49% accuracy in MathVista (Lu et al., 2023), a comprehensive benchmark to evaluate the mathematical reasoning capabilities of LLMs. In another instance, as tested in Frieder et al. (2024),

GPT-4’s performance in computing integration is dominated by specialized embedding-based models (Lample and Charton, 2019; Lample et al., 2022). Therefore, effective approaches to process semantically rich mathematical notation are necessary.

One approach is to compute a semantic representation of mathematical content based on textual context and treat mathematical expressions as a sentence without learning mathematical semantics (Krstovski and Blei, 2018). This has been used in various mathematical language processing (MLP) tasks, such as mathematical information retrieval (Topić et al., 2013; Mansouri et al., 2022a), identifier definition extraction (Pagael and Schubotz, 2014; Hamel et al., 2022; Zou et al., 2024), mathematical reasoning (Geva et al., 2020; Nye et al., 2021) and theorem proving (Wang et al., 2020; Wu et al., 2022). But this approach has two limitations: (1) textual descriptions are lacking in some math content like textbooks or mathematical derivations in academic publications, and (2) don’t really explore relations between symbolic operators from a mathematical perspective.

To address these problems, recent studies have focused on equivalence and mathematical manipulation between expressions to derive the meaning of mathematical expressions independent of context. However, these approaches remain limited, primarily due to the quality of available data. Allamanis et al. (2017) introduces EQNET, a TreeNN-based (Socher et al., 2013) model to group equivalent expressions together, but their datasets are limited to arithmetic and boolean expressions. SEMEMB (Gangwar and Kani, 2023) computes vector-based semantic representations by learning to generate mathematically equivalent expressions but is significantly limited by the SymPy-generated dataset (Meurer et al., 2017). SymPy functions are designed for simplification and directly derive the most simplified expression without intermediate steps, leading to a dataset that lacks enough

*Equal Contribution.

rewrites per expression. Hence, a more efficient mathematical data generation scheme is required.

In this work, we propose E-Gen, a highly scalable and efficient corpus generation scheme based on e-graphs (Willsey et al., 2021). Leveraging a collection of mathematical transformations, E-Gen facilitates the creation of synthetic datasets with large clusters of semantically equivalent expressions. This approach significantly improves upon prior SymPy-based methods, overcoming limitations in the number of rewrites per expression while improving flexibility and scalability. In summary, we highlight the following contributions of E-Gen:

- 1) We introduce E-Gen, a novel scheme for generating a cluster-based mathematical expression dataset, along with a high-diversity mathematical corpus.
- 2) We evaluate two types of embedding models based on seq2seq and contrastive learning respectively, showing improved semantic representation performance over prior works in quantitative and qualitative tests.
- 3) The embedding models are evaluated on two out-of-distribution (OOD) downstream tasks, to demonstrate models' generalizability and robustness.
- 4) Finally, we compare our models with GPT-4o across multiple tasks, demonstrating the effectiveness of embedding-based approaches.

2 Related Work

In mathematical language processing (MLP) (Meadows and Freitas, 2022), semantic representation shows strong potential across various problems. A representative application is mathematical information retrieval (MIR) (Kristianto et al., 2016; Zanibbi et al., 2016a; Mansouri et al., 2022c), where expressions or keywords are ranked based on their relevance to a query. Gao et al. (2017) proposes SYMBOL2VEC, a mathematical symbol representation generation scheme to group LaTeX symbols having similar contexts together. They further extend this approach to a MIR scheme (FORMULA2VEC) to prove its effectiveness. Krstovski and Blei (2018) converts symbolic layout trees (Zanibbi et al., 2016b) of mathematical expressions into token sequences and generates representation based on word2vec. Mansouri et al. (2019) further improves this approach by combining the symbol layout tree and operator tree representations of an expression. Peng et al. (2021)

proposes MathBERT, a pretrained model based on MIR and formula headline generation task (Yuan et al., 2020).

Identifier definition extraction (Kristianto et al., 2014) is another promising application, which aims to align identifiers found in scientific text with their definitions. Popovic et al. (2022) utilizes a transformer-based method to develop an end-to-end joint math entity and relation extraction approach. Jo et al. (2021) provides a mathematical understanding pretrained model by fine-tuning BERT on masked mathematical identifier prediction task. In the theorem proving field, Welleck et al. (2021b,a) generate theorem proof by retrieving relevant references to a given query theorem. However, as discussed in Section 1, MLP studies rely on semantic representation but mostly focus on establishing homomorphism between context and mathematical expressions to construct semantic representation, which is limited by the incapacity to process pure math content and can not really understand mathematical transformations. Even though some recent studies (Allamanis et al., 2017; Meidani et al., 2023) have focused on generating semantic representation based on intrinsic features of mathematical expressions to address this problem, their performance and scope of application are still limited by existing dataset, motivating this work's demonstration of a more effective mathematical corpus generation scheme.

3 Corpus of Equivalent Expressions

3.1 Corpus Generation

To generate a diverse set of mathematical expressions, we manually design templates containing placeholders for arithmetic operators, functional operators, and numerical values. Expressions are systematically instantiated by replacing the placeholders in templates with all possible operators and values, resulting in approximately 5,000 raw expressions in prefix notation. The templates are carefully designed to cover fundamental arithmetic and functional operators while maximizing the potential for mathematical transformations. Each of the resulting raw expressions is processed by E-Gen, which applies around 800 mathematical rules to generate clusters of semantically equivalent expressions.

The core of our E-Gen is *e-graph*, an advanced data structure designed to efficiently manipulate collections of terms under a congruence relation.

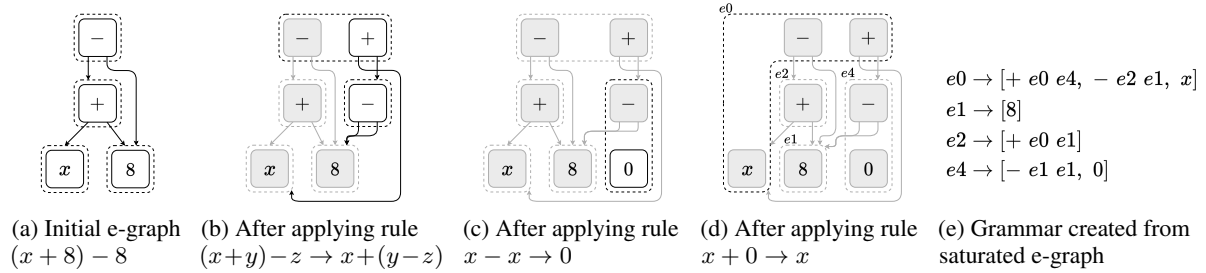


Figure 1: Illustration of e-graph saturation (1a to 1d) and grammar creation (1e). An e-graph consists of e-classes (dashed boxes) containing equivalent e-nodes (solid boxes). Edges connect e-nodes to their child e-classes. Applying mathematical rules to an e-graph adds new e-nodes and edges (1b and 1c), or merges e-classes (1d). Additions and modifications are emphasized in black. In 1e, the saturated e-graph is converted into a context-free grammar, where each grammar is defined using e-class id and the e-nodes with their child e-classes.

An e-graph is composed of e-classes, each containing a set of equivalent e-nodes. An e-node can be linked to one or more child e-classes, depending on the operator’s arity. From a mathematical perspective, child e-classes represent the arguments of their associated e-node, typically corresponding to a mathematical operator. Consequently, any subgraph originating from an e-node within the same e-class represents equivalent expressions.

Figures 1a to 1d show the e-graph saturation process. In the initial e-graph (Figure 1a), the e-node “+” is linked to two child e-classes “x” and “8”, respectively, and the e-node “-” is linked to a subgraph “(x+8)” and a child e-class “8”, together forming the raw expression “(x+8)-8”. In Figure 1b, a new e-node “+” is added to the top e-class after the associative law is introduced. The two e-nodes, “-” and “+” in the top e-class represent subgraphs for expressions “(x+8)-8” and “x+(8-8)” which are equivalent rewrites generated by the associative law. Similarly, transformations such as “(x-x) -> 0” and “x+0 -> x” are embedded into the e-graph in the steps shown in Figures 1c and 1d. The e-graph is iteratively expanded by applying each applicable rule, thereby capturing all possible equivalent expressions.

A context-free grammar is created after e-graph saturation in prefix notation, based on e-classes and the connections of e-nodes in it, as shown in Figure 1e. E-classes will be represented as variable symbols denoted by “e0”, “e1”, “e2”, “e4” in Figure 1d. Variables and numbers comprise the terminal set in the grammar. The production rules are determined by the edges and take the form:

$$e\langle \text{index} \rangle \rightarrow [var/num]$$

$$e\langle \text{index} \rangle \rightarrow [op e\langle \text{index} \rangle e\langle \text{index} \rangle, \dots]$$

where $e\langle \text{index} \rangle$ denotes eclass with corresponding

index, op and var/num denote enodes representing operators and variables/numbers.

We extract equivalent expressions from this grammar using a recursive rewrite algorithm. The expression extraction stage begins at a designated root e-class (“e0” in this case), traversing and replacing e-classes in the grammar with corresponding expansions until the grammar does not contain e-classes anymore. For instance, the second grammar “- e2 e1” of “e0”, “e2” can be replaced by “+ e0 e1” and generate “- + e0 e1 e1”. To avoid excessively long rewrites, a token length limit of 25 and a time limit of 600s are imposed. Eventually, clusters of semantically equivalent expressions for all raw expressions are generated by E-Gen and form a new corpus.

3.2 Corpus Analysis

Category	Operators
Arithmetic	+, -, ×, ÷
Logarithmic/Exponential	pow, abs, sqrt, $\frac{d}{dx}$
Trigonometric	ln, exp (as pow e)
	sin, cos, tan
	csc, sec, cot
Inverse Trigonometric	\sin^{-1} , \cos^{-1} , \tan^{-1}
	\csc^{-1} , \sec^{-1} , \cot^{-1}
Hyperbolic	sinh, cosh, tanh
	csch, sech, coth
Inverse Hyperbolic	\sinh^{-1} , \cosh^{-1} , \tanh^{-1}
	\csch^{-1} , \sech^{-1} , \coth^{-1}

Table 1: Operator coverage of E-Gen Corpus.

The new corpus encompasses a comprehensive set of arithmetic and functional operators as shown in Table 1. Table 2 presents partial examples from two clusters of equivalent expressions generated by E-Gen. The corpus is structured into clusters, where each cluster contains numerous mathematically equivalent expressions derived through a se-

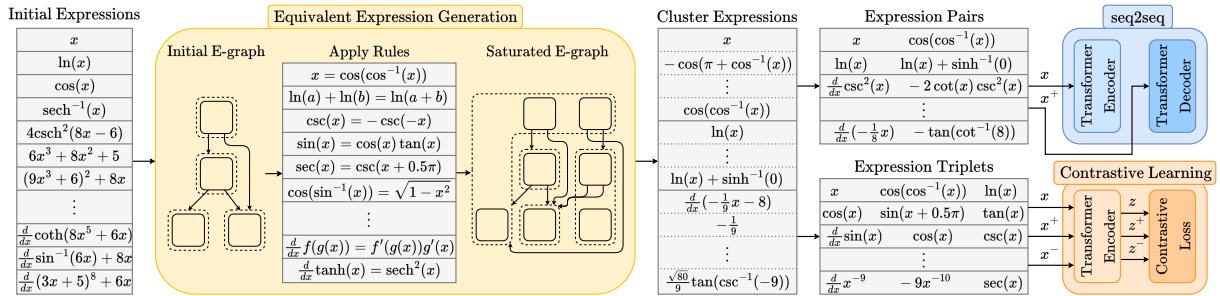


Figure 2: Overview of E-Gen and the cluster-based training framework. Equivalent expressions of each initial expression are generated using an e-graph-based approach, forming clusters of equivalent expressions. The seq2seq model is trained on equivalent expression pairs, while the contrastive learning model is trained on triplets, with each triplet containing a reference expression x , an equivalent positive sample x^+ , and a non-equivalent negative sample x^- . z is the latent space representation of the corresponding input x .

ries of flexible transformations. These transformations span both simple and complex mathematical relationships, significantly enhancing expression diversity and enabling models to learn underlying mathematical rules more efficiently with fewer raw expressions.

Raw expression: $\tanh(3x - (-4)) - 6$
$\tanh(3x + 4) - 6$
$1/\operatorname{coth}(3x + 4) - 6$
$\sinh(3x + 4)\operatorname{sech}(3x + 4) - 6$
$\tanh(3/\operatorname{csc}(\sin^{-1}(x)) + 4) - 6$
$\sinh(3\cos(\cos^{-1}(x)) + 4)/\operatorname{cosh}(3x + 4) - 6$
Raw expression: $\frac{d}{dx} - 2(\ln(x/7))^{-2}$
$4/(x(\ln(x/7))^3)$
$-2/(x/(\ln(x) - \ln(7))^{-3}) \times (-2)$
$-4x^{-1}/\ln(7 \times x^{-1})^3$
$-4/(x(\ln(7) - \ln(x))^3)$
$4\cot(\tan^{-1}(x))(\ln(x/7))^{-3}$

Table 2: Examples of equivalent expressions generated with E-Gen. Expressions listed below each raw expression are part of the equivalent rewrites. Additional examples are in Appendix A.2.

As detailed in Table 3, the E-Gen corpus not only supports a wider range of operators, but also achieves a significantly higher average cluster size of 102, compared to an existing corpora *Equivalent Expressions Dataset* (EED) (Gangwar and Kani, 2023) that is generated using SymPy and limited to a cluster size of 2 (pairs). This substantial increase in cluster size enables models to develop a deeper semantic understanding of the diversity of mathematical transformations.

Training Data. The training dataset is provided in two formats: expression pairs and triplets, corresponding to the two training methodologies described in Section 4. For the seq2seq model, equiva-

Statistic	E-Gen	EED
# Operators	34	24
Average Cluster Size	102	2
Average Sequence Length	15	16
Train Set Size (Pair)	$\sim 55\text{M}$	$\sim 4.66\text{M}$
Train Set Size (Triplet)	$\sim 50\text{M}$	-
Test Set Size	8,077	5,000

Table 3: Dataset Statistics. Corpus comparison between E-Gen and EED which denotes *Equivalent Expressions Dataset* from prior work (Gangwar and Kani, 2023).

lent expression pairs are generated by permuting expressions within each cluster. For contrastive learning, expression triplets are constructed by treating expressions within the same cluster as positive pairs, while randomly sampling expressions from different clusters to serve as negative examples. This process results in a training set comprising 55 million equivalent expression pairs and 50 million expression triplets as shown in Table 3. For the validation set, a subset of clusters is randomly sampled from the corpus, yielding a total of 8,077 expressions. Following Lample and Charton (2019) and Gangwar and Kani (2023), we use the prefix notation to encode the expressions.

4 Methodology

As illustrated in Figure 2, we employ two well-established approaches for learning representations of mathematical expressions: sequence-to-sequence (seq2seq) (Sutskever, 2014; Cho, 2014) equivalent expression generation and contrastive learning (CL) (Wu et al., 2018; Chen et al., 2020), both leveraging the transformer architecture (Vaswani, 2017). These methods aim to capture the underlying semantic relationships between mathematical expressions by generating meaning-

ful embeddings in a high-dimensional latent space.

4.1 seq2seq Expressions Generation

Following prior work (Gangwar and Kani, 2023), we employ the seq2seq framework to learn representations of mathematical expressions by training the model to generate mathematically equivalent expressions. Specifically, given a pair of equivalent expressions, the model takes one as input and is tasked with predicting the other as output. The encoder, during this process, learns to map the input expressions into a latent space where semantically equivalent expressions are clustered together.

4.2 Contrastive Learning

The other promising approach to learning mathematical expression representations is *contrastive learning*, a technique that has gained significant traction in the domain of representation learning. The primary objective is to learn a latent space where semantically equivalent expressions are embedded closer together, while semantically distinct expressions are pushed apart. This is achieved using a contrastive loss function, such as InfoNCE (Oord et al., 2018) or SimCSE (Gao et al., 2021).

In this manuscript, we use a variation of the InfoNCE loss formulated as:

$$\mathcal{L}(f) = \mathbb{E} \left[-\ln \frac{e^{f^\top(\mathbf{x})f(\mathbf{x}^+)}/\tau}{e^{f^\top(\mathbf{x})f(\mathbf{x}^+)}/\tau + e^{f^\top(\mathbf{x})f(\mathbf{x}^-)}/\tau} \right] \quad (1)$$

where f is a transformer encoder $f : \mathcal{X} \rightarrow \mathcal{Z}$ that maps an tokenized input expression $\mathbf{x} \in \mathcal{X}$ to its latent representation $\mathbf{z} \in \mathcal{Z}$. The terms \mathbf{x}^+ and \mathbf{x}^- correspond to positive (equivalent) and negative (non-equivalent) samples respectively, with τ as a temperature hyperparameter.

4.3 Representation Vector

To derive the representation vector from the transformer encoder, an additional step is required to convert the output matrix from the encoder’s last layer $\mathbf{X} \in \mathbb{R}^{S \times D_{\text{model}}}$ to a one-dimensional embedding vector $\mathbf{x} \in \mathbb{R}^{D_{\text{model}}}$ for each expression. Here, S and D_{model} represent the sequence length of the input and the model dimension. We experiment with two common pooling strategies: average pooling and max pooling over the hidden states of the last encoder layer. Our empirical analysis and prior work (Gangwar and Kani, 2023) indicate that max pooling consistently outperforms average pooling for the seq2seq model. In the case of contrastive learning, we train two separate models—one with

average pooling (CL Mean) and one with max pooling (CL Max)—to evaluate the impact of these strategies on embedding quality. In all models, special tokens, such as the start-of-expression and end-of-expression tokens, are excluded from the pooling process.

5 Experiments

5.1 Evaluation Tasks

The models trained on the E-Gen synthetic dataset, are compared against the prior SEMEMB model (Gangwar and Kani, 2023) trained on SymPy-generated corpus, on both in-distribution and out-of-distribution tasks, demonstrating the efficacy of the newly presented methods.

K-Means Clustering. K-Means clustering (Macqueen, 1967) is employed to evaluate the performance of both seq2seq and contrastive learning models. The test set comprises 8,077 expressions in 279 clusters, with cluster sizes ranging from 20 to 40 expressions. Subsequently, K-Means clustering is applied to group the expressions into clusters. A visualization of clusters is shown in Figure 3. To evaluate clustering performance, the predicted cluster labels from the K-Means algorithm are mapped to the corresponding ground truth labels, with accuracy computed as the proportion of expressions correctly assigned to their respective clusters.

Model	Accuracy (%)
seq2seq	96.72
CL Mean	97.61
CL Max	97.30
SEMEMB	37.70

Table 4: K-Means clustering accuracy (%) of seq2seq, CL Mean, and CL Max, compared against prior SEMEMB model.

As shown in Table 4, both the seq2seq and contrastive learning models, trained on the E-Gen cluster-based dataset, demonstrate strong performance in grouping equivalent expressions while separating non-equivalent ones. Notably, the CL models slightly outperform the seq2seq model in this task because they are trained to explicitly group similar expressions together. In contrast, SEMEMB, which is trained on a SymPy-generated corpus, only achieves 37.70% accuracy. The lower performance is attributed to the limited diversity of mathematical transformations in the SymPy-generated

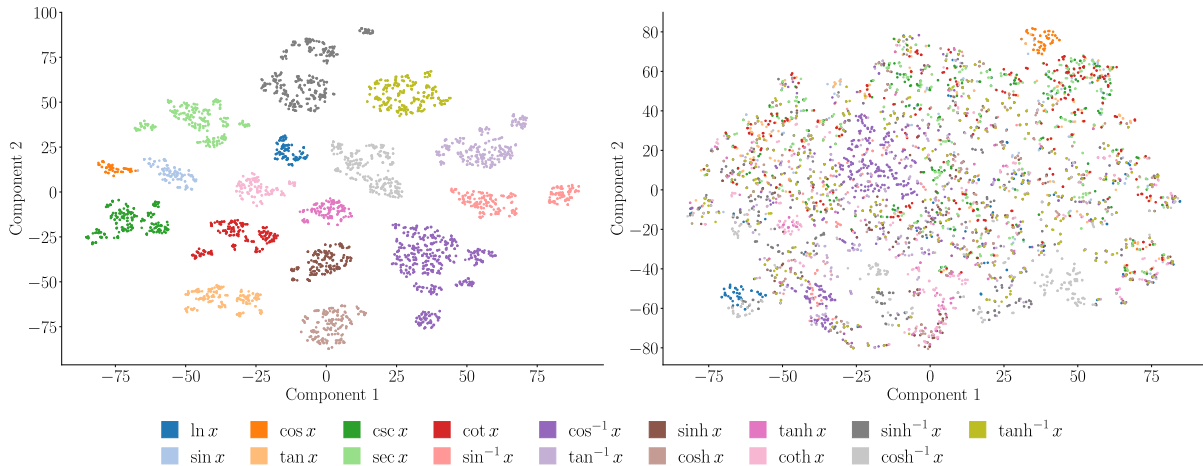


Figure 3: Visualization of representation vectors for 17 different single-operator mathematical expressions and their equivalent forms using our method (left) and SEMEMB (right). t-SNE (Van der Maaten and Hinton, 2008) is applied to reduce the dimensionality of the embeddings from 512 to 2.

corpus which restricts the model’s ability to effectively capture and distinguish small and large syntactic variations among equivalent expressions.

Semantic Understanding Beyond Syntactic Similarity.

To evaluate whether models trained on the E-Gen corpus capture mathematical semantics rather than relying on syntactic similarity, we design an experiment where the model must identify the single semantically equivalent expression among syntactically similar candidates. Specifically, we sample 1,000 query expressions from the test set and, for each query, we introduce one correct answer that is semantically equivalent but syntactically distinct from the query. Additionally, six distractors are generated by making minor syntactic modifications over the query and the correct answer (e.g., altering a single operator or numerical value). These distractors are divided into two groups: three closely resembling the query and three closely resembling the correct answer. The model is tasked with selecting the correct equivalent expression from seven candidates. In the example shown in Table 5, the candidate highlighted in bold represents the correct answer, which is syntactically distinct from the query involving the derivative operation. Candidate 4, 5, 7 are syntactically similar to the correct answer, while candidate 1, 3, 6 shares a similar structure with the query. As shown in Table 6, models trained on the E-Gen corpus significantly outperform SEMEMB, demonstrating the models’ ability to distinguish between semantically equivalent but syntactically different expressions, as well as syntactically similar yet semantically non-equivalent expressions.

Query expression: $\frac{d}{dx} 2x + \sin(x - 4)/6$
$\frac{d}{dx} 3x + \sin(x - 4)/6$
$2 + \cos(x - 4)/6$
$\frac{d}{dx} 2x + \sin(x + 4)/6$
$2 + \cos(x/4)/6$
$2 + \tan(x - 4)/8$
$\frac{d}{dx} 2x + \cos(x - 4)/6$
$9 + \cos(x - 4)/6$

Table 5: Examples of semantic understanding beyond syntactic similarity query, where the correct option is written in red. Six distractors are generated by slightly changing the query expression and the correct option to evaluate model’s capacity in distinguishing syntactically similar but semantically different expressions

Model	Accuracy (%)
seq2seq	76.41
CL Mean	50.36
CL Max	48.19
SemEmb	28.62

Table 6: Semantic understanding beyond syntactic similarity accuracy (%) of seq2seq, CL Mean, and CL Max, compared against prior SEMEMB model.

Mistake Detection. Mistake detection in mathematical derivations is an OOD downstream task, with the test set generated using SymPy to assess the robustness and generalizability of the models.

	x_1	y_1	x_2	\hat{y}_2 (seq2seq)	\hat{y}_2 (SEMEmb)	\hat{y}_2 (GPT-4o)	y_{gt}
1	$\sin(x)$	$-\sin(-x)$	$\cos(x)$	$\cos(-x)$	$-\tan(-x)$	$\cos(-x)$	$\cos(-x)$
2	$\cos(x)$	$\sec(x)$	$\tanh(x)$	$\coth(x)$	$\coth(x)$	$\operatorname{sech}(x)$	$\coth(x)$
3	$\sinh^{-1}(x)$	$\operatorname{csch}^{-1}(1/x)$	$\tanh^{-1}(x)$	$\coth^{-1}(1/x)$	-	$\coth^{-1}(x)$	$\coth^{-1}(1/x)$
4	$\tan(x)$	$\tan(x + \pi)$	$\csc(x)$	$\csc(x + 2\pi)$	$\cot(x + \pi)$	$\csc(x + 2\pi)$	$\csc(x + 2\pi)$
5	$\sin(x)$	$\cos(x - \pi/2)$	$\sec(x)$	$\csc(x + \pi/2)$	$\sec(x - \pi/2)$	$\csc(x - \pi/2)$	$\csc(x + \pi/2)$
6	x	$\ln x$	$\coth(x)$	$\ln \coth(x)$	$\coth \ln(x)$	$\coth^{-1}(x)$	$\ln \coth(x)$
7	x	$\ln x$	$\cos^{-1}(x)$	$\cos^{-1}(\ln x)$	$\cos^{-1}(\ln x)$	$\ln \cos^{-1}(x)$	$\ln \cos^{-1}(x)$
8	x	$\operatorname{csc}^{-1}(x)$	$\operatorname{csch}(x)$	$\operatorname{csc}^{-1}(\operatorname{csc}(x))$	-	$\operatorname{csch}^{-1}(x)$	$\operatorname{csc}^{-1}(\operatorname{csc}(x))$
9	x	$x + 1$	$\tan^{-1}(x)$	$\tan^{-1}(x) \times 1$	$\tan^{-1}(x) + 1$	$\tan^{-1}(x) + 1$	$\tan^{-1}(x) + 1$
10	x	x^3	$\sinh^{-1}(x)$	$\sinh^{-3}(x)$	$\sinh^{-3}(x)$	$\sinh^{-3}(x)$	$\sinh^{-3}(x)$

Table 7: Example comparison of embedding algebra predictions between the seq2seq, SEMEmb, and GPT-4o. The model’s prediction is denoted as \hat{y}_2 , while y_{gt} represents the ground truth. Incorrect predictions are in red. Additional experimental results are provided in Appendix C.3.

Algorithm 1 Threshold Calculation for Mistake Detection

- 1: **input:** set of derivations $\{D_i\}_{i=1}^N$, each containing a sequence of derivation steps $D_i = \{d_k\}_{k=1}^M$
- 2: **output:** threshold value t
- 3: f : transformer encoder
- 4: g : cosine similarity function
- 5: $x \leftarrow []$
- 6: **for all** $D_i \in \{D_1, \dots, D_N\}$ **do**
- 7: $Z_i \leftarrow f(D_i)$
- 8: $c \leftarrow g(Z_i[1:], Z_i[:-1], \text{dim}=-1)$
- 9: $c \leftarrow c \setminus \{c_{\text{mistake}}\}$
- 10: $c_{\min} \leftarrow \min(c)$
- 11: $x \leftarrow x \cup c_{\min}$
- 12: **end for**
- 13: $t \leftarrow \frac{1}{|x|} \sum_{i=1}^{|x|} x_i$
- 14: **Return** t

For this task, step-by-step mathematical derivations are generated for each expression in the E-Gen corpus, and mistakes are introduced into randomly selected steps within these derivations. The goal is to ask the models to identify erroneous steps by classifying transformations between consecutive steps as either "mistake" or "no mistake." To achieve this, a semantic similarity threshold for binary classification is calculated using only derivations from the training set, as described in Algorithm 1. Specifically, cosine similarities are computed between consecutive steps in each derivation, excluding steps with mistakes. The minimum cosine similarity among the correct steps is recorded for each derivation, and the average of these minimum values is used as the threshold. A step is flagged as a mistake if its cosine similarity with the preceding step falls below this threshold.

Examples of mistake detection are shown in Table 8. Mistakes in steps that share a similar structure with the preceding steps, such as those shown in the table, can appear deceptively similar to correct transformations, making them particu-

Derivation	seq2seq	GPT-4o
$-\frac{1}{8}(\sinh(\cosh^{-1}(7 - \frac{x}{8})))^{-1}$		
$-\frac{1}{8}(\sinh(\cosh^{-1}(7 + \frac{x}{8})))^{-1}$		
$-\frac{1}{8}(\sqrt{6 + \frac{x}{8}}\sqrt{8 + \frac{x}{8}})^{-1}$		★
$-\frac{1}{8}(6 + \frac{x}{8})^{-\frac{1}{2}}(8 + \frac{x}{8})^{-\frac{1}{2}}$		
$-\frac{1}{8}(6 + \frac{x}{8})^{-\frac{1}{2}}(8 + (1 - 8)x)^{-\frac{1}{2}}$	★	★
$-\frac{1}{8}(6 + \frac{x}{8})^{-\frac{1}{2}}(8 - 7x)^{-\frac{1}{2}}$		
$5/\csc(\csc^{-1}(1/\ln(\frac{x}{5})))$		
$5/\csc(\csc^{-1}(1/\ln(\frac{1}{5}x)))$		
$5/\operatorname{sec}(\csc^{-1}((\ln(\frac{1}{5}x))^{-1}))$	★	
$5/(1 - (\ln(\frac{1}{5}x))^2)^{-\frac{1}{2}}$		
$5\sqrt{1 - (\ln(\frac{1}{5}x))^2}$		

Table 8: Example comparison of mistake detection in mathematical derivations between seq2seq and GPT-4o. Errors in the derivations are in red. The ★ symbol indicates that the respective model has predicted the step to contain a mistake.

larly challenging to detect. This highlights the importance of leveraging semantic similarity for robust mistake identification. This poses a significant challenge to models’ ability to understand mathematical semantics.

Two examples of mistake detection are shown in Table 8, some steps with minor mistakes often share a similar structure with the preceding steps, such as mistakes steps shown in Table 8, making some incorrect transformations between consecutive steps appear deceptively more similar to correct mathematical transformations that significantly change the expressions’ structure.

The test set has 18,462 derivation steps, including 2,974 steps containing mistakes, with the remainder being error-free. As shown in Table 9, precision, recall, and F1-score are used to evaluate the performance. All three models trained on

	Model	Precision	Recall	F1
no mistake	seq2seq	96.40	94.69	95.54
	CL Mean	97.59	92.52	94.99
	CL Max	97.78	91.93	89.14
	SEMEmb	92.92	83.49	87.95
mistake	seq2seq	74.68	81.61	77.99
	CL Mean	69.33	88.10	77.60
	CL Max	67.96	89.14	77.12
	SEMEmb	44.46	67.50	53.61

Table 9: Mistake detection evaluation results precision, recall, and F1 (%) scores of seq2seq, CL Mean, and CL Max, compared against prior SEMEmb model.

E-Gen corpus demonstrate strong effectiveness in identifying potential mistakes and better generalizability in OOD task, significantly outperforming the SEMEmb approach.

Embedding Algebra. Embedding algebra is a classic task to evaluate if embeddings capture semantic information of an word/token. Techniques like word2vec (Mikolov, 2013) and GloVe (Pennington et al., 2014) exhibit the ability to perform analogy-based reasoning through algebraic operations on their representation vectors, facilitating solutions to analogies such as “Berlin is to Germany as Paris is to France”. By extending this task to mathematical analogies, we test if models can really understand mathematical transformations or just have a shallow understanding of expressions’ structure. For a given triplet of expressions x_1 , y_1 , and x_2 , we compute:

$$f(\hat{y}_2) = -f(x_1) + f(y_1) + f(x_2) \quad (2)$$

where f denotes a function $f : \mathcal{X} \rightarrow \mathcal{Z}$, which maps an expression x to its representation vector z . We then identify the expression whose embedding vector is most similar to $f(\hat{y}_2)$ in terms of cosine similarity, excluding the original expressions x_1 , y_1 , and x_2 from consideration.

For this experiment, 584 analogy examples are manually constructed, and all expressions from the E-Gen corpus are used as the search pool, removing those equivalent to x_2 and y_2 to ensure uniqueness of answer, while adding any necessary expressions to complete the analogy.

As shown in Table 10, the seq2seq model achieves the highest accuracy, indicating its ability to effectively learn underlying mathematical rules and handle basic substitutions. Conversely, SEMEmb exhibit low accuracy on this task and have a tendency to imitate the transformation between x_1

Model	Accuracy (%)
seq2seq	70.38
CL Mean	64.73
CL Max	50.34
SEMEmb	54.85
GPT-4o	39.60

Table 10: Embedding algebra accuracy (%) of seq2seq, CL Mean, and CL Max, compared against prior SEMEmb model and GPT-4o.

and y_1 , leading to “look-alike” predictions rather than a true understanding of the mathematical rules, such as Test 1, 4, 5. Examples are shown in Table 7.

5.2 Comparison with GPT-4o

To further assess the quality of vector representations of symbolic expressions, a comparative analysis is conducted between the seq2seq model, trained on the E-Gen corpus, and the state-of-the-art large language model GPT-4o (Achiam et al., 2023) on several tasks.

Mistake Detection. For mistake detection, a small test set is randomly sampled from the mistake detection test set mentioned above, consists of 322 derivation steps, of which 50 contain mistakes and the remainder are error-free. We first provide GPT-4o a derivation example in the prompt, asking it to identify potential mistakes. If GPT-4o fails to find the potential mistake, we teach it where the mistake appears until it can understand the task and pass the test. Once verified, we evaluate GPT-4o using derivation queries from the test set. As shown in Table 11, our work outperforms GPT-4o across all metrics. GPT-4o demonstrates a recall of 66%, suggesting its capability of detecting a notable portion of errors. However, its low precision indicates a tendency to misclassify correct transformations as mistakes, leading to a high rate of false positives.

	Model	Precision	Recall	F1
no mistake	seq2seq	96.98	94.49	95.72
	CL Mean	97.64	91.18	94.30
	CL Max	97.78	91.93	94.77
	GPT-4o	93.58	90.84	92.19
mistake	seq2seq	73.68	84.00	78.50
	CL Mean	64.71	88.00	74.58
	CL Max	67.96	89.14	77.12
	GPT-4o	56.90	66.00	61.11

Table 11: Mistake detection evaluation results precision, recall, and F1 (%) scores of seq2seq, CL Mean, and CL Max compared with GPT-4o.

Table 8 illustrates examples of GPT-4o’s misjudgments, which commonly occur during complex transformations, particularly those involving function operators or relatively intricate arithmetic calculations. For instance, in Example 1, GPT-4o performs suboptimally with evaluating equivalence when expressions are syntactically different (e.g., $\sinh(\cosh^{-1}(x)) = \sqrt{x+1}\sqrt{x-1}$). In Example 2, GPT-4o fails to identify the operator substitution in step 3, judging it as a valid derivation.

Embedding Algebra. We also conduct the embedding algebra task on GPT-4o, which performs worse than both the seq2seq and CL embedding models, achieving only 39.60% accuracy on the test set as shown in Table 10. Table 7 provides examples comparing the performance of the seq2seq model and GPT-4o on embedding algebra tests. Similar to the mistake detection test, an example query is presented to GPT-4o to verify its understanding and confirm that it produces the correct output before proceeding with the test. While the seq2seq model consistently makes accurate predictions by adhering to mathematical rules, GPT-4o shows a partial understanding of certain mathematical properties, such as correctly predicting the periodicity of “ $\csc(x)$ ”. However, it often imitates the structure of “ y_1 ” without fully considering the underlying mathematical rules. For instance, in Test 5, GPT-4o incorrectly imitates the structure of “ y_1 ”, subtracts “ $\pi/2$ ” from “ x ” in “ $\csc(x)$ ”, while the correct answer should be “ $\csc(x + \pi/2)$ ” which is mathematically equivalent to “ $\csc(x)$ ”. A similar error occurs in Test 8, where GPT-4o predicts “ y_2 ” as “ $\operatorname{csch}^{-1}(x)$ ” by imitating “ $y_1 : \csc^{-1}(x)$ ”, instead of applying the correct mathematical transformation. This comparison highlights the effectiveness of E-Gen corpus in helping models to build up understanding in mathematical rules and transformations.

6 Conclusion

In this work, we enhance semantic representations of symbolic expressions by developing E-Gen, a novel mathematical corpus generation framework based on the e-graph data structure. E-Gen shows strong capability in generation of a scalable cluster-based corpus with high diversity in mathematical transformation. To evaluate the effectiveness of the new corpus, two training approaches based on seq2seq and contrastive learning respectively are implemented to capture equivalence relation be-

tween expressions. We demonstrate the efficacy of these embedding models for a variety of downstream tasks such as: clustering, mathematical information retrieval, mistake detection, and mathematical analogies. Crucially, we show that these semantic representations outperform prior work and GPT-4o in both quantitative and qualitative tests. This work provides an algorithmic foundation for processing symbolic mathematics and the vector-based representations can integrate easily with vector embeddings of other data modalities.

7 Limitations

This work introduces and evaluates the potential of a novel mathematical corpus generation framework to augment mathematical semantic understanding. While the results are promising, there are several areas for improvement. First, extending the dataset to include a larger variety of operators used in published datasets, such as ArXMLiv (Kohlhase et al., 2024) and ARQMath (Mansouri et al., 2022b), would make the vector embeddings more applicable to real world data.

Following that, more efficient grammar enumeration techniques, such as those based on the Earley algorithm (Earley, 1970), will allow us to more easily extend the E-Gen corpus to include expressions with more complex operators and larger arity. In this manuscript, equivalent expressions are generated by enumerating the grammar obtained from the e-graph using a simple recursive function. Additionally, incorporating variable characteristics, such as dimensionality, phase, and/or bounds, into the embeddings would be useful and an avenue yet to be explored in modern literature.

Another limitation is the scarcity of real-world datasets that focus on symbolic mathematics. While tasks like mathematical information retrieval have been explored in competitions such as NTCIR (Zanibbi et al., 2016a) and ARQMath (Mansouri et al., 2022b), these typically involve a mix of symbolic math and natural language in queries and results. Integration of the our embedding methods with natural language vectors is a research area that has yet to be explored.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altschmidt, Sam Altman,

- Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.
- Miltiadis Allamanis, Pankajan Chanthirasegaran, Pushmeet Kohli, and Charles Sutton. 2017. Learning continuous semantic representations of symbolic expressions. In *International Conference on Machine Learning*, pages 80–88. PMLR.
- Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR.
- Kyunghyun Cho. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.
- Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2023. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113.
- Jacob Devlin. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.
- Jay Earley. 1970. An efficient context-free parsing algorithm. *Communications of the ACM*, 13(2):94–102.
- Simon Frieder, Luca Pinchetti, Ryan-Rhys Griffiths, Tommaso Salvatori, Thomas Lukasiewicz, Philipp Petersen, and Julius Berner. 2024. Mathematical capabilities of chatgpt. *Advances in neural information processing systems*, 36.
- Neeraj Gangwar and Nickvash Kani. 2023. Semantic representations of mathematical expressions in a continuous vector space. *Transactions on Machine Learning Research*.
- 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? *arXiv preprint arXiv:1707.05154*.
- Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. Simcse: Simple contrastive learning of sentence embeddings. *arXiv preprint arXiv:2104.08821*.
- Mor Geva, Ankit Gupta, and Jonathan Berant. 2020. Injecting numerical reasoning skills into language models. *arXiv preprint arXiv:2004.04487*.
- Emma Hamel, Hongbo Zheng, and Nickvash Kani. 2022. An evaluation of nlp methods to extract mathematical token descriptors. In *International Conference on Intelligent Computer Mathematics*, pages 329–343. Springer, Springer.
- Hwiyeol Jo, Dongyeop Kang, Andrew Head, and Marti A Hearst. 2021. Modeling mathematical notation semantics in academic papers. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 3102–3115.
- Michael Kohlhase et al. 2024. arxmliv project. <https://kwarc.info/projects/arXMLiv/>. Accessed: 2024-09-17.
- Giovanni Yoko Kristianto, Akiko Aizawa, et al. 2014. Extracting textual descriptions of mathematical expressions in scientific papers. *D-Lib Magazine*, 20(11):9.
- Giovanni Yoko Kristianto, Goran Topic, and Akiko Aizawa. 2016. Mcat math retrieval system for ntcir-12 mathir task. In *NTCIR*.
- Kriste Krstovski and David M Blei. 2018. Equation embeddings. *arXiv preprint arXiv:1803.09123*.
- Guillaume Lample and François Charton. 2019. Deep learning for symbolic mathematics. *arXiv preprint arXiv:1912.01412*.
- Guillaume Lample, Timothee Lacroix, Marie-Anne Lachaux, Aurelien Rodriguez, Amaury Hayat, Thibaut Lavril, Gabriel Ebner, and Xavier Martinet. 2022. Hypertree proof search for neural theorem proving. *Advances in neural information processing systems*, 35:26337–26349.
- I Loshchilov. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.
- Ilya Loshchilov and Frank Hutter. 2016. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*.
- Pan Lu, Hritik Bansal, Tony Xia, Jiacheng Liu, Chunyuan Li, Hannaneh Hajishirzi, Hao Cheng, Kai-Wei Chang, Michel Galley, and Jianfeng Gao. 2023. Mathvista: Evaluating mathematical reasoning of foundation models in visual contexts. *arXiv preprint arXiv:2310.02255*.
- J Macqueen. 1967. Some methods for classification and analysis of multivariate observations. In *Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability/University of California Press*.
- Behrooz Mansouri, Anurag Agarwal, Douglas W Oard, and Richard Zanibbi. 2022a. Advancing math-aware search: the arqmath-3 lab at clef 2022. In *European Conference on Information Retrieval*, pages 408–415. Springer.
- Behrooz Mansouri, Vít Novotný, Anurag Agarwal, Douglas W Oard, and Richard Zanibbi. 2022b. Overview of arqmath-3 (2022): Third clef lab on answer retrieval for questions on math. In *International Conference of the Cross-Language Evaluation Forum for European Languages*, pages 286–310. Springer.

- Behrooz Mansouri, Vít Novotný, Anurag Agarwal, Douglas W Oard, and Richard Zanibbi. 2022c. Third clef lab on answer retrieval for questions on math (working notes version. *Proc. CLEF 2022 (CEUR Working Notes)*).
- Behrooz Mansouri, Shaurya Rohatgi, Douglas W Oard, Jian Wu, C Lee Giles, and Richard Zanibbi. 2019. Tangent-cft: An embedding model for mathematical formulas. In *Proceedings of the 2019 ACM SIGIR international conference on theory of information retrieval*, pages 11–18.
- Jordan Meadows and Andre Freitas. 2022. A survey in mathematical language processing. *arXiv preprint arXiv:2205.15231*.
- Kazem Meidani, Parshin Shojaee, Chandan K Reddy, and Amir Barati Farimani. 2023. Snip: Bridging mathematical symbolic and numeric realms with unified pre-training. *arXiv preprint arXiv:2310.02227*.
- Aaron Meurer, Christopher P. Smith, Mateusz Paprocki, Ondřej Čertík, Sergey B. Kirpichev, Matthew Rocklin, AMiT Kumar, Sergiu Ivanov, Jason K. Moore, Sartaj Singh, Thilina Rathnayake, Sean Vig, Brian E. Granger, Richard P. Muller, Francesco Bonazzi, Harsh Gupta, Shivam Vats, Fredrik Johansson, Fabian Pedregosa, Matthew J. Curry, Andy R. Terrel, Štěpán Roučka, Ashutosh Saboo, Isuru Fernando, Sumith Kulal, Robert Cimrman, and Anthony Scopatz. 2017. *Sympy: symbolic computing in python*. *PeerJ Computer Science*, 3:e103.
- Tomas Mikolov. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.
- Maxwell Nye, Anders Johan Andreassen, Guy Gur-Ari, Henryk Michalewski, Jacob Austin, David Bieber, David Dohan, Aitor Lewkowycz, Maarten Bosma, David Luan, et al. 2021. Show your work: Scratchpads for intermediate computation with language models. *arXiv preprint arXiv:2112.00114*.
- Aaron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*.
- Robert Pagael and Moritz Schubotz. 2014. Mathematical language processing project. *arXiv preprint arXiv:1407.0167*.
- Shuai Peng, Ke Yuan, Liangcai Gao, and Zhi Tang. 2021. Mathbert: A pre-trained model for mathematical formula understanding. *arXiv preprint arXiv:2105.00377*.
- Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.
- Nicholas Popovic, Walter Laurito, and Michael Färber. 2022. Aifb-webscience at semeval-2022 task 12: Relation extraction first—using relation extraction to identify entities. *arXiv preprint arXiv:2203.05325*.
- Alec Radford. 2018. Improving language understanding by generative pre-training.
- Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.
- Jack W Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, et al. 2021. Scaling language models: Methods, analysis & insights from training gopher. *arXiv preprint arXiv:2112.11446*.
- Shaurya Rohatgi, Wei Zhong, Richard Zanibbi, Jian Wu, and C Lee Giles. 2019. Query auto completion for math formula search. *arXiv preprint arXiv:1912.04115*.
- Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.
- I Sutskever. 2014. Sequence to sequence learning with neural networks. *arXiv preprint arXiv:1409.3215*.
- Goran Topić, Giovanni Yoko Kristianto, Minh-Quoc Nghiem, and Akiko Aizawa. 2013. The mcmt math retrieval system for ntcir-10 math track. In *Proceedings of 10th NTCIR Conference, Tokyo, Japan*, pages 680–685.
- Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of machine learning research*, 9(11).
- A Vaswani. 2017. Attention is all you need. *Advances in Neural Information Processing Systems*.
- Qingxiang Wang, Chad Brown, Cezary Kaliszyk, and Josef Urban. 2020. Exploration of neural machine translation in autoformalization of mathematics in mizar. In *Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs*, pages 85–98.
- Sean Welleck, Jiacheng Liu, Ronan Le Bras, Hannaneh Hajishirzi, Yejin Choi, and Kyunghyun Cho. 2021a. Naturalproofs: Mathematical theorem proving in natural language. *arXiv preprint arXiv:2104.01112*.
- Sean Welleck, Jiacheng Liu, Jesse Michael Han, and Yejin Choi. 2021b. Towards grounded natural language proof generation. In *MathAI4Ed Workshop at NeurIPS*.

Max Willsey, Chandrakana Nandi, Yisu Remy Wang, Oliver Flatt, Zachary Tatlock, and Pavel Panchekha. 2021. Egg: Fast and extensible equality saturation. *Proceedings of the ACM on Programming Languages*, 5(POPL):1–29.

Yuhuai Wu, Albert Qiaochu Jiang, Wenda Li, Markus Rabe, Charles Staats, Mateja Jamnik, and Christian Szegedy. 2022. Autoformalization with large language models. *Advances in Neural Information Processing Systems*, 35:32353–32368.

Zhirong Wu, Yuanjun Xiong, Stella X Yu, and Dahua Lin. 2018. Unsupervised feature learning via non-parametric instance discrimination. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3733–3742.

Ke Yuan, Dafang He, Zhuoren Jiang, Liangcai Gao, Zhi Tang, and C Lee Giles. 2020. Automatic generation of headlines for online math questions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 9490–9497.

Richard Zanibbi, Akiko Aizawa, Michael Kohlhase, Iadh Ounis, Goran Topic, and Kenny Davila. 2016a. Ntcir-12 mathir task overview. In *NTCIR*.

Richard Zanibbi, Kenny Davila, Andrew Kane, and Frank Wm Tompa. 2016b. 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*, pages 145–154.

Richard Zanibbi, Behrooz Mansouri, and Anurag Agarwal. 2024. Mathematical information retrieval: Search and question answering. *arXiv preprint arXiv:2408.11646*.

Jiaru Zou, Qing Wang, Pratyush Thakur, and Nickvash Kani. 2024. Stem-pom: Evaluating language models math-symbol reasoning in document parsing. In *The 4th Workshop on Mathematical Reasoning and AI at NeurIPS'24*.

A Cluster-Based Corpus

A.1 Generation Efficiency

Efficiency is a key property of any corpus generation scheme. To evaluate the E-Gen approach, the average time for e-graph saturation and expression rewrite extraction is measured. On average, e-graph saturation process takes 36.83ms, while extraction takes 1.46s per expression. The process is run on an Intel Xeon E5-2666 CPU with a 25-token length limit and a 600s time limit for each raw expression.

A.2 Cluster Example

As discussed in Section 3, the E-Gen framework enables wider coverage of operators, such as derivatives, and the application of a diverse range of mathematical rules. Table 12 shows additional examples of equivalent expression clusters. The key distinction between the E-Gen corpus and prior datasets is that E-Gen consists of clusters of equivalent expressions, rather than simple expression pairs. Each cluster contains a raw expression along with numerous equivalent rewrites. This diverse set of transformations significantly improves the performance of pretrained models in understanding the semantics of mathematical transformations.

B Training Details

Both the seq2seq and contrastive learning models are trained using a transformer architecture with 8 attention heads, 6 layers, a model dimensionality of 512, a feedforward dimension of 2048, and a batch size of 256. All models are optimized using the AdamW optimizer (Loshchilov, 2017) with a learning rate of 10^{-4} and the CosineAnnealingWarmRestarts scheduler (Loshchilov and Hutter, 2016). Training is conducted on a cluster equipped with an AMD EPYC 7763 64-Core Processor and a 46GB L40S GPU. The training time of each model is shown in Table 13.

C Experiments

C.1 K-Means Clustering

Accuracy Calculation. Since all expressions from the same cluster are labeled in the same class, the accuracy for K-Means clustering in Section 5.1 is calculated as follow.

$$acc = \frac{1}{K} \sum_{i=1}^K acc_i \quad (3)$$

Raw expression: $(\sin(2x + 5))^{-8}$
$(\sin(5 - (-2) \sin(\sin^{-1}(x))))^{-8}$ $(\csc(\pi + (-5 - 2x)))^8$ $(\cos(\pi/2 - 2x - 5))^{-8}$ $\left(\frac{\cot(2x+5)}{\cos(2x+5)}\right)^8$ $(\sin(2(x + \pi) + 5))^{-8}$
Raw expression: $\sec^{-1}(7x + 6)/(-7)$
$\cos^{-1}\left(\frac{1}{6-(-7x)}\right)/(-7)$ $(\sec^{-1}(-7x - 6) - \pi) \times \frac{1}{7}$ $\left(\frac{\pi}{2} - \csc^{-1}(7x + 6)\right)/(-7)$ $(-1)(\sec^{-1}(7/\sec(\cos^{-1}(x)) + 6))/7$ $(\cos^{-1}(\cot(\tan^{-1}(7x + 6))))/(-7)$
Raw expression: $(\csc(x/7))^{-3} - (-3)$
$1/\left(\csc\left(\frac{\tan(\tan^{-1}x)}{7}\right)\right)^3 + 3$ $(\tan(\cot^{-1}(\sin(x/7))))^{-3} + 3$ $3 + (1/\cos(\sec^{-1}(\sin(x/7))))^3$ $-1(-3 - (\csc(x/7))^{-3})$ $(\sin(x/7))^3 + 3$ $(\sin(\csc^{-1}(\sin(7^{-1}x))))^{-3} - (-3)$
Raw expression: $\frac{d}{dx}(-4x - 8)^{-4} + 7x$
$(-1) \times (8 - (-4x))^{-5} \times 16 + 7$ $-16 \times (4x + 8)^{-5} + 7$ $7 + (-16) \times (-1/(-4x - 8)^5)$ $7 - (-16) \times (-1) \times (8 - (-4x))^{-5}$ $0 - (-16(-4x - 8)^{-5} - 7)$
Raw expression: $\frac{d}{dx}(-9 \sinh(-7x + 2))$
$63/\operatorname{sech}(-7x + 2)$ $-9/(-1/7 \cosh(7x - 2))$ $63 \cosh(2 - 7x)$ $\frac{63}{2}(\exp(2 - 7x) + \exp(7x - 2))$ $7 \times 9/\operatorname{sech}(7x - 2)$

Table 12: Additional examples of equivalent expressions generated with E-Gen. Expressions listed below each raw expression are all equivalent to it.

where acc_i denotes the accuracy of cluster i , which is computed as follow.

$$acc_i = \frac{1}{|c_i|} \sum_{x_j \in c_i} \mathbb{1}_{\{g(x_j)=c_i\}} \quad (4)$$

where $g(x_j)$ denotes the cluster predicted by K-Means algorithm, and c_i is the ground truth cluster of expression x_j .

C.2 Formula Retrieval

Formula retrieval is implemented as an extra task of semantic understanding. As discussed in Section 1, prior MIR studies rely heavily on contextual cues for semantic representation (Gao et al., 2017; Krstovski and Blei, 2018) rather than really explore mathematical property of expressions, resulting in

Model	Training Time (h:min)
seq2seq	38:04
CL Mean	31:56
CL Max	29:59

Table 13: Training time of seq2seq, CL Mean, and CL Max models.

suboptimal performance, especially in scenarios with limited surrounding text, such as textbook or mathematical derivations. To evaluate the model’s potential in this scenario, we design a test simulating pure formula retrieval based on the E-Gen corpus. Specifically, given a query expression, the top- k most similar expressions are retrieved based on cosine similarity in the continuous embedding space.

Model	$k=5$	$k=10$	$k=15$	$k=20$
seq2seq	99.93	99.82	99.76	99.41
CL Mean	99.85	99.73	99.50	98.94
CL Max	99.78	99.52	99.32	98.82
SemEmb	73.47	60.57	51.95	45.56

Table 14: Mathematical information retrieval accuracy (%) of seq2seq, CL Mean, and CL Max, compared against prior SEMEMB model at different top- k values. Accuracy denotes the proportion of top- k ranked candidates that are semantically equivalent to the query expression.

The test set of E-Gen corpus which contains 8,077 expressions is tested, each expression is used as a query while the remaining expressions are used as candidates. Figure 4 shows an example in which top-4 most similar candidates to $\cos x$ are selected. As Table 14 shows, both seq2seq and contrastive learning models based on E-Gen corpus can effectively retrieve the most semantically relevant expressions from the candidate pool as k increases. Conversely, due to a limited number of rewrites per expression, SEMEMB struggles to identify semantically equivalent expressions when k exceeds 10.

C.3 Embedding Algebra

Table 15 and 16 present additional examples from embedding algebra task. The seq2seq model correctly predicts most answers, outperforming other models. CL Mean slightly outperforms, CL Max, SEMEMB, and GPT-4o. Notably, SEMEMB and GPT-4o use a similar strategy to predict “ y_2 ”. For instance, Tests 6 to 11 evaluate the models’ understanding of function periodicity, where the seq2seq

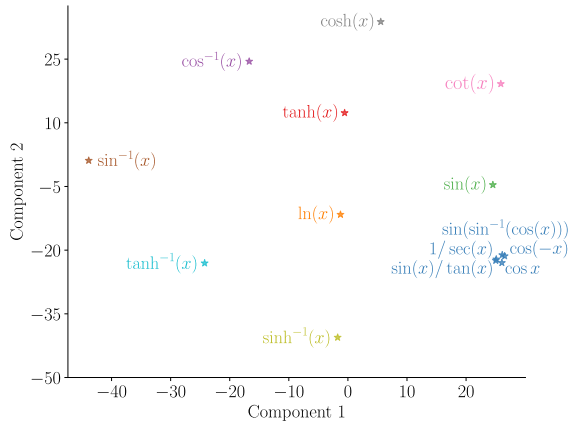


Figure 4: Example of the formula retrieval with seq2seq model. The query expression is $\cos(x)$ in darker blue, and the rest are candidates. Top-4 expressions are retrieved. t-SNE is applied to reduce the dimensionality of the embeddings from 512 to 2.

model trained on the E-Gen cluster-based corpus accurately predicts “ y_2 ” in most cases. In contrast, as discussed in Section 5.2, SEMEMB, similar to GPT-4o, tends to imitate the transformation between “ x_1 ” and “ y_1 ” rather than correctly applying the periodicity of the function. In Test 6, “ $x_1 : \sin(x)$ ” and “ $y_1 : \sin(x + 2\pi)$ ” are equivalent due to the 2π period of “ $\sin(x)$ ”. However, since “ $x_2 : \cot(x)$ ” has a period of π , the correct “ y_2 ” should be “ $\cot(x + \pi)$ ”. The SEMEMB model just simply imitates the transformation from “ x_1 ” to “ y_1 ” by adding “ 2π ” to “ x ” in “ $\cot(x)$ ”, and incorrectly converts “ $\cot(\dots)$ ” into “ $\csc(\dots)$ ”. This comparison highlights that the models trained with E-Gen corpus shows a better understanding in mathematical rules and transformations, rather than relying on superficial "looks-like" predictions.

Question

I will give you mathematical derivation steps, try to find out all possible mistakes in it

- Step 1: $-\frac{1}{8} \left(\sinh \left(\cosh^{-1} \left(7 - \frac{x}{-8} \right) \right) \right)^{-1}$
- Step 2: $-\frac{1}{8} \left(\sinh \left(\cosh^{-1} \left(7 - x \frac{1}{-8} \right) \right) \right)^{-1}$
- Step 3: $-\frac{1}{8} \left(\sinh \left(\cosh^{-1} \left(7 - \frac{1}{-8} x \right) \right) \right)^{-1}$
- Step 4: $-\frac{1}{8} \left(\sinh \left(\cosh^{-1} \left(7 + \frac{x}{8} \right) \right) \right)^{-1}$
- Step 5: $-\frac{1}{8} \left(\sqrt{6 + \frac{x}{8}} \sqrt{8 + \frac{x}{8}} \right)^{-1}$
- Step 6: $-1 \times \frac{1}{8} \left(6 + \frac{x}{8} \right)^{-\frac{1}{2}} \left(8 + \frac{x}{8} \right)^{-\frac{1}{2}}$
- Step 7: $-\frac{1}{8} \left(6 + \frac{x}{8} \right)^{-\frac{1}{2}} (8 + (1 - 8)x)^{-\frac{1}{2}}$
- Step 8: $-\frac{1}{8} \left(6 + \frac{x}{8} \right)^{-\frac{1}{2}} (8 + (1 - 8)x)^{-\frac{1}{2}}$
- Step 9: $-\frac{1}{8} \left(6 + \frac{x}{8} \right)^{-\frac{1}{2}} (8 + (1 + (-8))x)^{-\frac{1}{2}}$
- Step 10: $-\frac{1}{8} \left(6 + \frac{x}{8} \right)^{-\frac{1}{2}} (8 + (-7)x)^{-\frac{1}{2}}$
- Step 11: $-\frac{1}{8} \left(6 + \frac{x}{8} \right)^{-\frac{1}{2}} (8 - 7x)^{-\frac{1}{2}}$

GPT-4o Response

Here's a brief summary of the mistakes:

- Step 5:** The transition from $\sinh(\cosh^{-1} \dots)$ to $\sqrt{6 + \frac{x}{8}}$ and $\sqrt{8 + \frac{x}{8}}$ is incorrect. The substitution does not properly follow the hyperbolic identity. It seems like you're jumping from hyperbolic to standard square root expressions without justification.
 - Step 7:** There's a mistake in the expression $(8) + (1 - 8)x$, which simplifies to $(8) - (7)x$, but the transition introduced incorrect operations leading to errors in simplification.
 - Step 9:** Same issue persists with $(8) + (1 + (-8))x$, simplifying this should have led to $(-7)x$, but it's repeated without any changes.
- The primary issue occurs when transitioning from hyperbolic trigonometric functions in steps 4-5. The correct path should have kept the form involving $\sinh(\cosh^{-1})$ or transformed more systematically into algebraic terms.

Question

I will give you mathematical derivation steps, try to find out all possible mistakes in it

- Step 1: $\frac{5}{\csc(\csc^{-1}(\frac{1}{\ln(\frac{x}{-5})}))}$
- Step 2: $\frac{5}{\csc(\csc^{-1}(\frac{1}{\ln(x \frac{1}{-5})}))}$
- Step 3: $\frac{5}{\csc(\csc^{-1}(\frac{1}{\ln(\frac{1}{-5}x)})}$
- Step 4: $\frac{5}{\sec(\csc^{-1}((\ln(\frac{1}{-5}x))^{-1}))}$
- Step 5: $\frac{5}{1 - ((\ln(\frac{1}{-5}x))^2)^{-\frac{1}{2}}}$
- Step 6: $5\sqrt{1 - (\ln(\frac{-1}{5}x))^2}$

GPT-4o Response

This derivation is correct. Here's a breakdown of why:

- Steps 1 to 3:** You simplified $\frac{5}{\csc(\csc^{-1}(\dots))}$ correctly, transitioning into the logarithmic form involving $\ln(\frac{x}{-5})$. The transformation from division into multiplication by $\frac{1}{\csc}$ is also accurate.
 - Step 4:** The transition from $\csc(\csc^{-1}(\dots))$ to $\sec(\csc^{-1}(\dots))$ is valid because $\csc^{-1}(y) = \frac{1}{\sin(y)}$, and switching to sec correctly follows trigonometric identities here.
 - Step 5:** The step $\sec(\csc^{-1}(\dots)) = \sqrt{1 - (\ln(\dots))^2}$ is a valid simplification using the Pythagorean identity, $\sec(x) = \sqrt{1 - \sin^2(x)}$.
 - Step 6:** Finally, the expression $\sqrt{1 - (\ln(\dots))^2}$ is accurate, representing the correct simplified form of the secant expression.
- The derivation logically follows through to the final result:
 $5\sqrt{1 - (\ln(\frac{-1}{5}x))^2}$
 This matches the correct sequence of operations and transformations.

Figure 5: Two mistake detection conversation examples in GPT-4o prompt. The mistakes in the derivation are in red. In the left one, GPT-4o successfully detects the mistake step, but also reports a false positive mistake step. In the right one, GPT-4o fails to detect the mistake in the derivation at all.

	x_1	y_1	x_2	\hat{y}_2 (seq2seq)	\hat{y}_2 (CL Mean)	y_{gt}
1	$\sin(x)$	$-\sin(-x)$	$\cos(x)$	$\cos(-x)$	$\cos(-x)$	$\cos(-x)$
2	$\sin(x)$	$-\sin(-x)$	$\tan(x)$	$-\tan(-x)$	$-\tan(-x)$	$-\tan(-x)$
3	$\cos(x)$	$\sec(x)$	$\tanh(x)$	$\coth(x)$	$\coth(x)$	$\coth(x)$
4	$\sin^{-1}(x)$	$\csc^{-1}(1/x)$	$\cos^{-1}(x)$	$\sec^{-1}(1/x)$	$\sec^{-1}(1/x)$	$\sec^{-1}(1/x)$
5	$\sinh^{-1}(x)$	$\operatorname{csch}^{-1}(1/x)$	$\tanh^{-1}(x)$	$\coth^{-1}(1/x)$	$\coth^{-1}(1/x)$	$\coth^{-1}(1/x)$
6	$\sin(x)$	$\sin(x + 2\pi)$	$\cot(x)$	$\cot(x + \pi)$	$\cot(x + \pi)$	$\cot(x + \pi)$
7	$\tan(x)$	$\tan(x + \pi)$	$\csc(x)$	$\csc(x + 2\pi)$	$\csc(x + 2\pi)$	$\csc(x + 2\pi)$
8	$-\cos(x)$	$\cos(x + \pi)$	$-\cot(x)$	$\cot(x) \times 1$	$\cos(x)(1/\sin(x))$	$\tan(x + \pi/2)$
9	$\sin(x)$	$\cos(x - \pi/2)$	$\sec(x)$	$\csc(x + \pi/2)$	$\csc(x + \pi/2)$	$\csc(x + \pi/2)$
10	$-\csc(x)$	$\csc(x + \pi)$	$-\sec(x)$	$\sec(x + \pi)$	$\cot(x) \csc(\pi/2 + x)$	$\sec(x + \pi)$
11	$-\cos(x)$	$\cos(x + \pi)$	$-\tan(x)$	$\tan(\pi + x)$	$\sin(-\csc^{-1}(\cot(x)) + \pi)$	$\cot(x + \pi/2)$
12	x	$\ln x$	$\sin(x)$	$\ln \sin(x)$	$\sin(\ln(x))$	$\ln \sin(x)$
13	x	$\ln x$	$\coth(x)$	$\ln \coth(x)$	$\ln \coth(x)$	$\ln \coth(x)$
14	x	$\ln x$	$\cos^{-1}(x)$	$\cos^{-1}(\ln x)$	$\ln \sec^{-1}(x)$	$\ln \cos^{-1}(x)$
15	x	$\sin^{-1}(x)$	$\cos(x)$	$\sin^{-1}(\cos(x))$	$\sin^{-1}(\cos(x))$	$\sin^{-1}(\cos(x))$
16	x	$\csc^{-1}(x)$	$\operatorname{csch}(x)$	$\csc^{-1}(\operatorname{csch}(x))$	$\csc^{-1}(\sinh(x))$	$\csc^{-1}(\csc(x))$
17	x	$x + 1$	$\tan^{-1}(x)$	$\tan^{-1}(x) \times 1$	$\tan^{-1}(x) + 1$	$\tan^{-1}(x) + 1$
18	x	$x - 1$	$\sin(x)$	$\sin(x) - 1$	$\sin(x) - 1$	$\sin(x) - 1$
19	x	$x/1$	$\tan(x)$	$\tan(x)/1$	$(\frac{d}{dx} \ln(x))^1$	$\tan(x)/1$
20	x	x^3	$\operatorname{csch}(x)$	$\operatorname{csch}^3(x)$	$\operatorname{csch}^3(x)$	$\operatorname{csch}^3(x)$
21	x	x^3	$\sinh^{-1}(x)$	$\sinh^{-3}(x)$	$5 + (1/\tan(\cot^{-1}(x)))^3$	$\sinh^{-3}(x)$

Table 15: Additional examples of embedding algebra with seq2seq and CL Mean models. Both models are trained using E-Gen corpus and this evaluation shows the difference between seq2seq and CL Mean models. \hat{y}_2 is the prediction from the corresponding model, and y_{gt} is the ground truth. The expressions in red are wrong predictions. The seq2seq model offers superior performance over the CL Mean model.

	x_1	y_1	x_2	\hat{y}_2 (CL Max)	\hat{y}_2 (SEMEmb)	\hat{y}_2 (GPT-4o)	y_{gt}
1	$\sin(x)$	$-\sin(-x)$	$\cos(x)$	$\cos(-x)$	$-\tan(-x)$	$\cos(-x)$	$\cos(-x)$
2	$\sin(x)$	$-\sin(-x)$	$\tan(x)$	$-\tan(-x)$	$-\tan(-x)$	$-\tan(-x)$	$-\tan(-x)$
3	$\cos(x)$	$\sec(x)$	$\tanh(x)$	$\coth(x)$	$\coth(x)$	$\operatorname{sech}(x)$	$\coth(x)$
4	$\sin^{-1}(x)$	$\csc^{-1}(1/x)$	$\cos^{-1}(x)$	$\sec^{-1}(1/x)$	-	$\sec^{-1}(1/x)$	$\sec^{-1}(1/x)$
5	$\sinh^{-1}(x)$	$\operatorname{csch}^{-1}(1/x)$	$\tanh^{-1}(x)$	$\coth^{-1}(1/x)$	-	$\coth^{-1}(x)$	$\coth^{-1}(1/x)$
6	$\sin(x)$	$\sin(x + 2\pi)$	$\cot(x)$	$\cot(x + \pi)$	$\csc(x + 2\pi)$	$\cot(x + \pi)$	$\cot(x + \pi)$
7	$\tan(x)$	$\tan(x + \pi)$	$\csc(x)$	$\csc(x + 2\pi)$	$\cot(x + \pi)$	$\csc(x + 2\pi)$	$\csc(x + 2\pi)$
8	$-\cos(x)$	$\cos(x + \pi)$	$-\cot(x)$	$\sec(\cos^{-1}(\cot(x)))$	$\sec(x + \pi)$	$\cot(x + \pi)$	$\tan(x + \pi/2)$
9	$\sin(x)$	$\cos(x - \pi/2)$	$\sec(x)$	$\cos(x)$	$\sec(x - \pi/2)$	$\csc(x - \pi/2)$	$\csc(x + \pi/2)$
10	$-\csc(x)$	$\csc(x + \pi)$	$-\sec(x)$	$\csc(\pi/2 + x)$	$\sec(x + \pi)$	$\sec(x + \pi)$	$\sec(x + \pi)$
11	$-\cos(x)$	$\cos(x + \pi)$	$-\tan(x)$	$\sin(\csc^{-1}(\cot(x)))$	$\tan(x + \pi)$	$\tan(x + \pi)$	$\cot(x + \pi/2)$
12	x	$\ln x$	$\sin(x)$	$\ln(\csc(x))$	$\ln \sin(x)$	$\sin^{-1}(x)$	$\ln \sin(x)$
13	x	$\ln x$	$\coth(x)$	$\ln \coth(x)$	$\coth \ln(x)$	$\coth^{-1}(x)$	$\ln \coth(x)$
14	x	$\ln x$	$\cos^{-1}(x)$	$\sec^{-1}(\ln x)$	$\cos^{-1}(\ln(x))$	$\ln \cos^{-1}(x)$	$\ln \cos^{-1}(x)$
15	x	$\sin^{-1}(x)$	$\cos(x)$	$\sin^{-1}(\cos(x))$	$\cos(\sin^{-1}(x))$	$\cos^{-1}(x)$	$\sin^{-1}(\cos(x))$
16	x	$\csc^{-1}(x)$	$\operatorname{csch}(x)$	$\sin^{-1}(\operatorname{csch}(x))$	-	$\operatorname{csch}^{-1}(x)$	$\csc^{-1}(\csc(x))$
17	x	$x + 1$	$\tan^{-1}(x)$	$\tan^{-1}(x)/1$	$\tan^{-1}(x) + 1$	$\tan^{-1}(x) + 1$	$\tan^{-1}(x) + 1$
18	x	$x - 1$	$\sin(x)$	$\sin(x) - 1$	$\sin(x) + 1$	$\sin(x - 1)$	$\sin(x) - 1$
19	x	$x/1$	$\tan(x)$	$\frac{d}{dx} \ln x$	$\tan(x)/1$	$\tan(x)/1$	$\tan(x)/1$
20	x	x^3	$\operatorname{csch}(x)$	$\operatorname{csch}^3(x)$	-	$\operatorname{csch}^3(x)$	$\operatorname{csch}^3(x)$
21	x	x^3	$\sinh^{-1}(x)$	$\operatorname{csch}^{-1}(x)$	$\sinh^{-3}(x)$	$\sinh^{-3}(x)$	$\sinh^{-3}(x)$

Table 16: Additional examples of embedding algebra comparing the embedding model (CL Max) trained on the E-Gen corpus presented in this manuscript with prior work (SEMEmb) and GPT-4o. \hat{y}_2 is the prediction from the corresponding model, and y_{gt} is the ground truth. The expressions in red are wrong predictions.