MathAMR+: A Unified Graph Neural Network Framework for Multimodal Mathematical Information Retrieval

by

Jacob Yoon

THESIS

Presented to the Faculty of the Department of Computer Science
Golisano College of Computing and Information Sciences
Rochester Institute of Technology

in Partial Fulfillment
of the Requirements
for the Degree of

Master of Science in Computer Science

Rochester Institute of Technology
May 2026

MathAMR+: A Unified Graph Neural Network Framework for Multimodal Mathematical Information Retrieval

---

APPROVED BY
SUPERVISING COMMITTEE

Dr. Richard Zanibbi, Advisor

Dr. Carlos Rivero, Reader

Dr. Matthew Fluet, Observer

Date

MathAMR+: A Unified Graph Neural Network Framework for Multimodal Mathematical Information Retrieval

Jacob Yoon, M.S.

Rochester Institute of Technology, 2026

Advisor: Dr. Richard Zanibbi

Abstract

Mathematical Information Retrieval (MIR) focuses on developing systems that enable users to search for and retrieve documents containing mathematical content. A key challenge in building effective math-aware retrieval systems lies in jointly modeling the symbolic and operational structure of mathematical expressions together with their surrounding linguistic context. Existing approaches often use linear token sequences, losing structural information, or rely on separate models for text and math, limiting their ability to capture cross-modal patterns and learn contextualized representations.

This thesis proposes MathAMR+, a graph neural network-based retrieval framework that jointly models Abstract Meaning Representation (AMR) graphs, Operator Trees (OPTs), and Symbol Layout Trees (SLTs). Unlike prior work that linearizes similar graph representations for Transformer encoders, the proposed model preserves graph structure and learns contextualized multi-vector representations for fine-grained alignment between textual and mathematical components. It employs a Relational Graph Convolutional Network (RGCN) with hierarchical virtual nodes at the formula, sentence, and document levels to enable structured aggregation and long-range message passing. The model is trained using contrastive learning for dense multi-vector retrieval and evaluated on the ARQMath-3 benchmark.

The central research question is whether explicitly modeling contextualized text-formula interactions through a unified graph representation improves math-aware retrieval compared to prior approaches that either linearize such representations or rely on separate specialized techniques for the two modalities. Initial results suggest that MathAMR+ effectively leverages its joint graph representation to achieve strong math-aware retrieval performance, with experiments indicating that its gains stem from the complementary structure of operator and symbol layout trees, the robustness of relational message passing over hierarchical virtual nodes, and the benefits of fine-grained multi-vector interactions for aligning mathematical expressions with surrounding textual context.

Acknowledgments

Thank you to my advisor, Dr. Richard Zanibbi, for his invaluable guidance, encouragement, and insight throughout this thesis. Working with him has been a deeply enriching experience that has shaped both this work and my growth as a researcher. I would also like to thank my fellow DPRL members, Abhisek Dey, Bryan Amador, Patrick Philippy, and Ricky Saynganthone, for their ideas, advice, and many engaging conversations. I thank my Reader, Dr. Carlos Rivero, for his helpful feedback in refining this work, and my Observer, Dr. Matthew Fluet, for his time in reviewing it. Finally, I thank my family and friends for their unwavering support.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Mathematical Information Retrieval (MIR) focuses on developing systems that enable users to search for and retrieve documents containing mathematical content. A vast amount of mathematical information exists across scientific papers, textbooks, websites, and other repositories, yet accessing and understanding it remains a challenge. Traditional search engines primarily operate on textual data and perform poorly with mathematical expressions, which have unique semantic and structural properties that require specialized techniques. More effective MIR systems could transform how this knowledge is discovered and applied. For example, they can help researchers trace the use of equations and theorems, or assist educators and students in locating relevant proofs, examples, or problem solutions. MIR systems can also support downstream tasks in Natural Language Processing (NLP) and Information Retrieval (IR) in which mathematical knowledge must be interpreted or reasoned over. For instance, retrieval-augmented generation (RAG) allows Large Language Models (LLMs) to query MIR systems for precise mathematical content, improving the accuracy of tasks such as question answering and problem solving [30].

Within MIR, the specific task of *math-aware search* [62] refers to retrieval that leverages both the mathematical notation and its surrounding textual context, enabling more precise retrieval when formulas appear within explanations, proofs, or descriptive passages. This thesis focuses on math-aware search, where accurately modeling and understanding the interplay between formulas and their textual context is essential for effective retrieval.

Mathematical notation encodes meaning through its operational structure, spatial layout of symbols, and context within surrounding text, presenting challenges distinct from those of natural language. To effectively capture these interacting layers of information, MIR systems represent formulas using structured representations such as operator trees (OPTs) and symbol layout trees (SLTs). As shown in Figure 1.1, OPTs capture the hierarchical relationships between mathematical operators and operands, while SLTs encode the two-dimensional spatial arrangement of symbols.

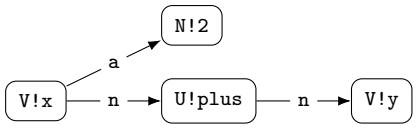
Together, OPTs and SLTs provide complementary information. For instance, the division operator can be visually presented in multiple forms, such as $\frac{a}{b}$, a/b, and $a \div b$. Conversely, the notation $a(b)$ can represent either multiplication or function application depending on whether a denotes a scalar or a function. Utilizing both representations enables retrieval systems to more accurately model the structure and meaning of mathematical expressions.

Formula

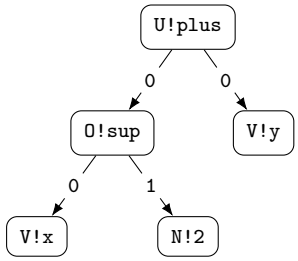
$x^2 + y$

Tree Representations

Symbol Layout Tree (SLT)



Operator Tree (OPT)



Node labels `N!`, `V!`, `O!`, and `U!` indicate symbol types: (N)umbers, (V)ariables, (O)rdered operators, and (U)nordered operators. Edge labels in the SLT denote positional relationships (*e.g.*, `a` for "above" and `n` for "next"), while edge labels in the OPT indicate the order of operands for ordered operations.

Figure 1.1: The Symbol Layout Tree (SLT) and Operator Tree (OPT) representations for the formula $x^2 + y$.

Leveraging these representations, researchers have explored various approaches to math-aware search. Several methods have treated text and mathematical content as separate modalities, combining retrieval scores at a late stage [27,31,37,54,68]. However, this separation prevents the model from learning how formulas and their surrounding text interact to convey meaning. Other methods have attempted to model formulas and text jointly, for example by combining formula tokens with surrounding textual context in a single model input to capture cross-modal relationships [28, 44]. While this represents progress toward joint modeling, these approaches still treat the mathematical and textual content as flat sequences rather than preserving their inherent hierarchical structure.

An interesting direction emerged in 2022 when Mansouri et al. [35] introduced MathAMR, a unified graph representation that connects Abstract Meaning Representation (AMR) graphs of natural language with OPTs for formulas. AMRs are directed, rooted graphs that capture the hierarchical predicate-argument structure of a sentence, representing concepts as nodes and their relationships as edges. By generating an AMR for a passage and substituting each formula with

its corresponding OPT, MathAMR unified the hierarchical structures of natural language and mathematical expressions. However, their approach linearized the graphs into a sequence of tokens via depth-first traversal and encoded them using Sentence-BERT, a Transformer model that can be viewed as a fully connected graph which learns relationships between all token pairs. While this allowed the model to capture contextual dependencies, it lost explicit structural information, forcing relational reasoning to emerge implicitly through attention rather than through the graph's inherent topology. MathAMR also only incorporated operator structure, omitting spatial layout information that can provide useful cues for matching, as equivalent operations may be expressed in multiple visual forms. A representative example is shown in Figure 1.2, visualizing this representation and its resulting sequential form.

1.1 Research Question and Hypothesis

To address these limitations, this thesis presents a novel framework for math-aware search that extends MathAMR by departing from graph linearization in favor of a Graph Neural Network (GNN) that operates directly over the graph structure. The graph representation builds on the unified structure introduced by MathAMR, adapting their formula extraction code and extending the representation by inserting SLTs as sibling subgraphs alongside their corresponding OPTs, encoding complementary spatial layout information alongside semantic operator structure. The AMR backbone captures the predicate-argument structure of the question across sentences, with each formula integrated as a subgraph via `:math` edges and its OPT and SLT attached as siblings via `:opt` and `:slt` edges, respectively. The augmented graph representation, which we call MathAMR+, is visualized in Figure 1.2. This work also constructs and makes publicly available the full ARQMath collection of MathAMR+ graphs, including question-accepted-answer pair labels.

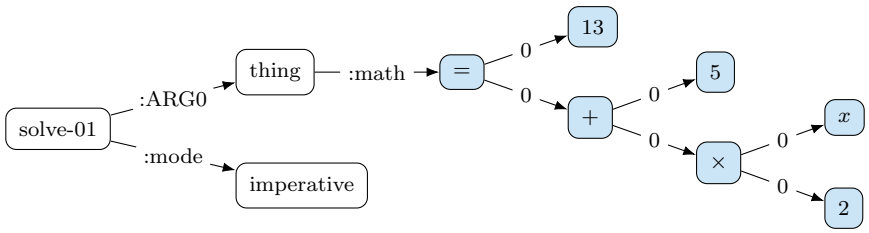
To encode these graphs, the model adapts the Relational Graph Convolutional Network (RGCN) encoder and ColBERT-style [24] MaxSim retrieval mechanism of Amador et al. [4] for the MathAMR+ setting. The encoder represents each graph as a set of nodes connected by typed edges and performs relation-specific message passing, learning separate transformation functions for each edge type to propagate and aggregate information across the heterogeneous structure. This mechanism enables the model to capture patterns encoded across the AMR, OPT, and SLT subgraphs within a unified graph structure. Building on that architecture, this thesis introduces hierarchical virtual nodes at the formula, sentence, and document levels, each connecting bidirectionally to the nodes within its corresponding subgraph (shown as v_{F_1}, v_{S_1}, and v_D in Figure 1.2), enabling efficient global message passing while preserving the graph's inherent topology. The model is trained via contrastive learning on question-accepted-answer pairs, where the set of virtual nodes serve as the aggregate representations over which MaxSim scoring is applied, performing fine-grained alignment

Original Text

"Solve $2x + 5 = 13$."

MathAMR

Graph Representation



Linearized String

```
solve-01 thing math = + times 2 x 5 13 imperative
```

MathAMR+

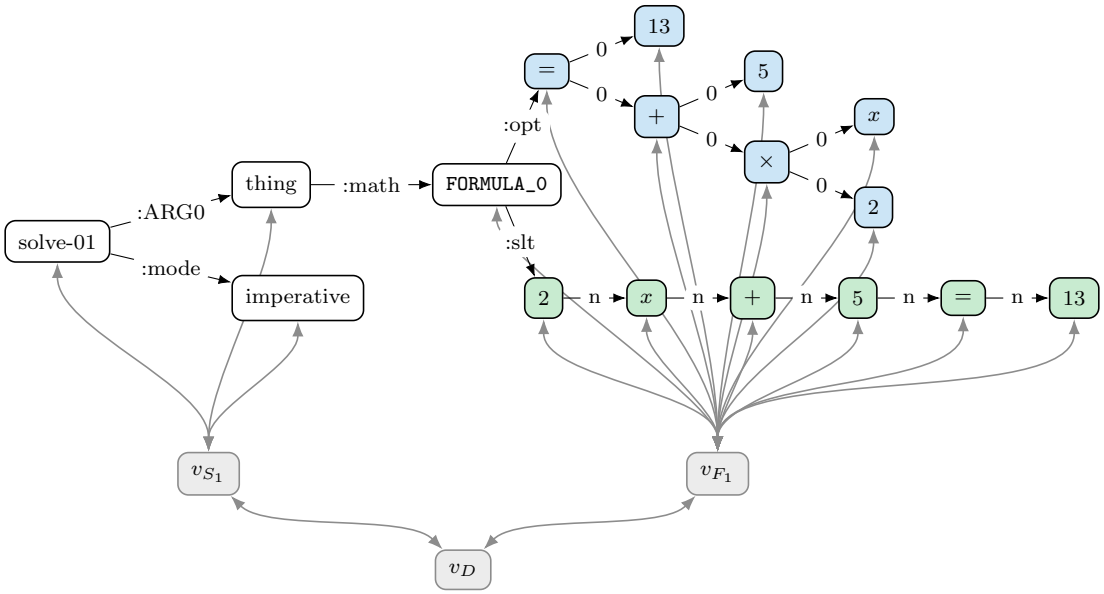


Figure 1.2: MathAMR vs. MathAMR+ representations for "Solve $2x+5 = 13$". Formula, sentence, and document virtual nodes are shown as v_{F_1}, v_{S_1}, and v_D, respectively.

between the structural components of the query and the document.

This thesis investigates the following core research question:

> *By explicitly capturing contextualized text-formula interactions, how effectively does the proposed representation and model architecture perform math-aware search, compared to sequence-based approaches?*

I hypothesize that the explicit preservation of the MathAMR graph topology through relational message passing captures the interplay between formulas and text more effectively than the linearization used in previous approaches. Furthermore, the MaxSim multi-vector interaction mechanism operating over the virtual nodes will enhance retrieval precision by allowing each query virtual node to independently match against its most similar counterpart in the document, rather than averaging over the entire document uniformly.

A systematic investigation of this framework suggests that operator trees and symbol layout trees contribute distinct structural signals that jointly support more effective retrieval. We also observe that the full virtual node hierarchy, combined with ColBERT-style MaxSim scoring, achieves consistent performance gains over using no virtual nodes or a single global aggregate, enabling fine-grained alignment between the structural components of query and document. The results further indicate that relational message passing can integrate heterogeneous graph features in MathAMR+ without being sensitive to the initial state of the hierarchical virtual nodes. Overall, by explicitly modeling and preserving graph topology, the proposed graph-based neural architecture retains the structural relationships between mathematical expressions and their surrounding textual context, resulting in more effective math-aware retrieval on ARQMath-3 than the prior approach that applies linearization and a transformer model.

Chapter 2

Background

This chapter surveys relevant background and related work in Mathematical Information Retrieval (MIR), Abstract Meaning Representations (AMRs), Graph Neural Networks (GNNs), and contrastive learning for dense retrieval, establishing the conceptual and technical foundations for the methods discussed in this thesis.

2.1 Mathematical Information Retrieval

Information Retrieval (IR) studies methods for retrieving information from large collections using queries that express a user's information need. Mathematical Information Retrieval (MIR) extends IR to documents containing formulas and expressions. As discussed in Chapter 1, traditional search engines struggle with mathematical content due to its symbolic structure, spatial layout, surrounding text, and domain context. For example, expressions that share identical symbols can differ semantically, such as $x^2 + y$ versus x^{2+y}, or how $a(b)$ may mean either multiplication or function application. Similarly, the same symbol can convey different meanings based on contextual information, such as how λ may denote a wavelength in physics, an eigenvalue in linear algebra, or a rate parameter in probability. These factors complicate the task, requiring MIR systems to employ specialized methods to manage mathematical information effectively.

An effective MIR search engine supports a number of real-world mathematical search tasks. For example, a researcher might seek relevant papers, theorems, or formulas using a query expressed in mathematical notation. In an educational setting, a student might look for solutions or explanations to a specific problem. In engineering or data science contexts, users may search for derivations or models applicable to their work.

MIR research encompasses several core tasks. These include *formula search*, which retrieves a ranked list of expressions relevant to a query formula; *math-aware search*, which jointly considers expressions and their accompanying text; and *question answering*, where the system produces or

generates a single answer to a mathematical question [65]. This thesis focuses on math-aware search, addressing the challenge of integrating linguistic and mathematical information to produce accurate and contextually grounded retrieval.

2.1.1 Formula Representations

In MIR, mathematical formulas are encoded in a machine-readable format such as LaTeX or Mathematical Markup Language (MathML). LaTeX is a typesetting system designed for producing high-quality mathematical documents, but it provides limited information about the underlying semantics of expressions. For example, the caret symbol "`^`" represents a superscript in LaTeX, but it can mean different operations based on the context. In some cases, it represents exponentiation, as in `x^2` meaning x squared. In other contexts, it could denote an upper limit (`\sum_{i=1}^n` meaning $\sum_{i=1}^{n}$) or label components of vectors and tensors in physics (`v^i` for the i-th component of v). This illustrates the need for MathML, a standard XML-based representation that provides Presentation MathML, which encodes the visual layout of formulas, and Content MathML, which captures the semantic structure of expressions, representing operators, operands, and hierarchical relationships explicitly. Symbol layout trees (SLTs) can be derived from Presentation MathML and LaTeX, capturing the two-dimensional spatial arrangement of symbols, and operator trees (OPTs) can be built from Content MathML, encoding the explicit operators and operands and how subexpressions combine to form a hierarchical structure. Examples of the various formula representations are shown in Figure 2.1.

2.1.2 Test Collections

To benchmark MIR systems, a number of major competitions and test collections have been established, most notably NTCIR and ARQMath.

The NTCIR-10 Math Pilot Task (2013) [2] was the first large-scale benchmark for mathematical retrieval, using 100,000 arXiv papers for formula and document retrieval. NTCIR-11 Math-2 (2014) [3] expanded coverage to 105,120 papers and 8.3 million paragraphs, including a Wikipedia subtask for finer-grained retrieval. NTCIR-12 (2016) [63] refined evaluation methods and emphasized semantic understanding, encouraging hybrid text-formula models.

The ARQMath lab series (ARQMath-1 [66] in 2020, ARQMath-2 [38] in 2021, ARQMath-3 [34] in 2022) shifted focus to Math Stack Exchange (MSE)[1], a community question-answering forum. The underlying MSE collection used across the series contains approximately 2.47 million posts in total: roughly 1.02 million question posts and 1.45 million answer posts, with approximately 28.3 million mathematical formulas distributed across these posts. ARQMath included two main

[1] `https://math.stackexchange.com/`

Formula

$x^2 + y$

LaTeX

```
x^2 + y
```

MathML

Presentation MathML

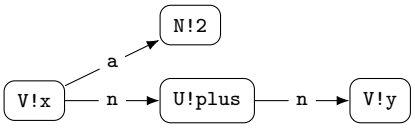
```
<math>
    <msup>
        <mi>x</mi>
        <mn>2</mn>
    </msup>
    <mo>+</mo>
    <mi>y</mi>
</math>
```

Content MathML

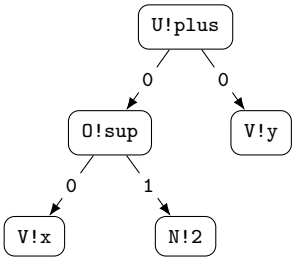
```
<apply>
    <plus/>
    <apply>
        <power/>
        <ci>x</ci>
        <cn>2</cn>
    </apply>
    <ci>y</ci>
</apply>
```

Tree Representations

Symbol Layout Tree (SLT)



Operator Tree (OPT)



Node labels `N!`, `V!`, `O!`, and `U!` indicate symbol types: (N)umbers, (V)ariables, (O)rdered operators, and (U)nordered operators. Edge labels in the SLT denote positional relationships (*e.g.*, `a` for "above" and `n` for "next"), while edge labels in the OPT indicate the order of operands for ordered operations.

Figure 2.1: The LaTeX, Presentation MathML, Content MathML, Symbol Layout Tree (SLT), and Operator Tree (OPT) representations for the formula $x^2 + y$.

tasks: Answer Retrieval for Math Questions (Task 1), and Formula Search (Task 2). ARQMath-3 introduced a pilot Task 3, Open Domain Question Answering, allowing systems to generate free-form answers from the same questions as Task 1. These tasks provide large-scale, real-world benchmarks for both formula-based and answer-based retrieval, with tasks 1 and 2 particularly relevant for this thesis as the benchmarks for evaluating math-aware retrieval models.

ARQMath-3 contains 101 topics (query question posts) and systems are required to retrieve a ranked list of up to 1,000 results per topic. Task 1 topics are full question posts—including their title, body text, and embedded formulas— for which participating systems must rank candidate answer posts from the collection. Task 2 is designed to support both formula-only and contextualized retrieval approaches. Task 2 topics utilize the same set of question posts as Task 1, but with specific formula identifiers marking which formula within that post is the query formula of interest. Systems may exploit only the isolated query formula for purely structural matching, or leverage the surrounding post text and other formulas in the post to perform context-aware retrieval. In either case, the output must be a ranked list of candidate formulas from the collection.

Both NTCIR and ARQMath rely on relevance judgments made by human assessors stored in quantitative relevance assessments (qrels) files[2]. NTCIR assessors used three- to five-level scales, often binarized for evaluation. For ARQMath, the official Task 1 qrels cover 78 of the 101 topics and contain 34,847 judgments across 33,383 unique answer posts, while the official Task 2 qrels cover 76 topics with 11,538 judgments across 9,975 unique formulas. ARQMath used a four-level scale for both formula and answer retrieval. For answer retrieval, a relevance score of 3 ("High") indicates that a post is sufficient to fully answer the question on its own; 2 ("Medium") denotes that the post offers partial progress toward a solution, such as clarifying the question or identifying steps; 1 ("Low") reflects information that may help interpret the question or understand an answer; and 0 ("Not Relevant") is assigned to posts that provide no information relevant to the question, including posts that merely restate the question without providing new information. For Task 2, human assessors viewed the full posts in which each query and candidate formula appears, assessing how useful the formula is in its surrounding context rather than in isolation. This design accommodates a wide spectrum of MIR approaches, from structure-based methods that ignore surrounding text to contextualized models that incorporate textual information.

2.1.3 Evaluation Metrics

We now discuss the set of IR metrics used to evaluate retrieval effectiveness in this thesis, which are consistent with the official ARQMath evaluation framework and the MIR literature more broadly.

Precision at k (P@k) is the proportion of the top k ranked results that are relevant (often

[2]The qrels format follows the TREC specification (`https://trec.nist.gov/data/qrels_eng`).

Formula Search Task Example (Task 2)

Query Formula:

$$\int_a^b f(x)\,dx$$

Relevance 3 (Equivalent to Exact Match): $\int_a^b f(x)\,dx = F(b) - F(a)$
Relevance 2 (Useful but Not as Good as Exact): $\int_0^1 g(t)\,dt$
Relevance 1 (Some Chance of Usefulness): $\int f(x)\,dx$
Relevance 0 (Not Relevant): $\frac{d}{dx} f(x)$

Answer Retrieval Task Example (Task 1)

Query (MSE Question Post):

Title: Proving the limit $(1 + 1/n)^n = e$

Question: How do I prove that $\lim_{n\to\infty} \left(1 + \frac{1}{n}\right)^n = e$?

Tags: calculus, limits, analysis

Relevance 3 (High – Fully Answers the Question):
"You can prove the limit by expanding $(1 + \frac{1}{n})^n$ using the binomial theorem and showing that, as $n \to \infty$, the sum converges to the series $e = \sum_{k=0}^{\infty} \frac{1}{k!}$. Specifically:

$$\left(1 + \frac{1}{n}\right)^n = \sum_{k=0}^{n} \binom{n}{k} \frac{1}{n^k}$$

Each term approaches $\frac{1}{k!}$ as $n \to \infty$.

Relevance 2 (Medium – Provides a Useful Step or Clarification):
"One way to show the limit equals e is to interpret $(1 + \frac{1}{n})^n$ as a compound interest expression and analyze its growth. This gives intuition for why the value approaches e."

Relevance 1 (Low – Context That Might Help Interpretation):
"The number e appears in calculus as the base of the natural logarithm and arises naturally in growth-rate problems."

Relevance 0 (Not Relevant):
"To convert between logarithm bases, use $\log_b(x) = \frac{\ln(x)}{\ln(b)}$."

Figure 2.2: Example queries, results, and relevance ratings for formula search and answer retrieval tasks. Relevance levels follow the ARQMath evaluation scheme [34].

top 3, 5, or 10), capturing immediate user utility. Mean Average Precision (MAP) extends this by computing the mean precision at every rank where a relevant item appears, then averaging across queries; it rewards systems that surface relevant items early and consistently rather than burying them deep in the list. Normalized Discounted Cumulative Gain (nDCG) further extends to graded relevance by weighting each result by its relevance score with a logarithmic rank discount, normalized against the ideal ranking to yield a score in $[0, 1]$.

Because ARQMath qrels cover only a subset of topics and a small fraction of the candidate pool, ARQMath uses prime variants of these metrics, denoted with a prime symbol ($'$), which simply exclude unjudged items from the computation. nDCG' and MAP' are evaluated over the full ranked list of 1,000 retrieved results, making them deep metrics that assess ranking quality throughout the entire submission rather than only at the top. nDCG' is particularly well suited to ARQMath's four-level relevance scale, as it distinguishes the contribution of highly relevant results from partially useful ones at every rank position. MAP' captures a complementary view, measuring how early and consistently relevant items appear across all 1,000 ranks. P'@10 focuses only on the top ten results with relevance ratings, reflecting that users rarely examine further than the first page of results. Together, nDCG', MAP', and P'@10 form the official ARQMath evaluation suite [34].

Binary Preference (Bpref) is additionally reported, which, like the prime metrics, only considers judged items and measures how many relevant items a system ranks before non-relevant ones. Where MAP' averages precision at the ranks of relevant documents, Bpref instead measures, for each relevant document, how many judged non-relevant documents are ranked above it, up to a maximum of $|R|$ (the total number of relevant documents). Bpref focuses more directly on penalizing cases where judged non-relevant documents outrank relevant ones, making it more concerned with relative ordering among judged results.

For binary metrics (P'@10, MAP', and Bpref), ARQMath's four-level relevance scale is thresholded such that relevance scores of 2 or 3 are treated as relevant, while scores of 0 or 1 are treated as non-relevant.

These metrics, shown in Figure 2.3, together capture ranking quality at depth (nDCG', MAP', Bpref) and user-facing utility at the top of the list (P'@10). While a wide array of additional metrics are used in IR research, this concise set of metrics provides a sufficiently comprehensive evaluation of math-aware retrieval performance for the purposes of this thesis, and is consistent with the official ARQMath evaluation framework.

2.2 Evolution of MIR Models

This section traces the progression of MIR models from early text-based systems to modern neural approaches, highlighting how each generation addresses specific challenges in representing and

Binary Relevance Metrics

Precision at rank k	$P@k = \frac{\#\text{ relevant items in top } k}{k}$
Average Precision	$AP = \frac{1}{\lvert R\rvert} \sum_{k \in K_r} P@k$, where R is the set of relevant documents returned and K_r is the set of ranks at which relevant documents appear
Mean Average Precision	$MAP = \frac{1}{\lvert Q\rvert} \sum_{q \in Q} AP(q)$, where Q is the set of queries

Graded Relevance Metrics

Cumulative Gain	$CG@k = \sum_{i=1}^{k} rel_i$, where rel_i is the relevance score of document i
Discounted Cumulative Gain	$DCG@k = \sum_{i=1}^{k} \frac{rel_i}{\log_2(i+1)}$
Ideal DCG	$iDCG@k = DCG@k$ computed on the top k documents sorted in descending order of relevance (the best-possible ranking)
Normalized DCG	$nDCG@k = \frac{DCG@k}{iDCG@k}$

Metrics Accounting Only for Judged Items

Binary Preference	$Bpref = \frac{1}{\lvert R\rvert} \sum_{k \in K_r} \left(1 - \frac{\min(\lvert S\rvert, \lvert R\rvert)}{\lvert R\rvert}\right)$, where S is the set of non-relevant items ranked before k
Prime Metrics (*e.g.*, P'@k, nDCG'@k)	Exclude unjudged items

Figure 2.3: Evaluation metrics used in this thesis, adapted from Zanibbi et al. [65]. Prime variants (nDCG', MAP', P'@10) and Bpref are adopted from the ARQMath evaluation protocol to account for incomplete judgment sets. For binary metrics, ARQMath relevance scores of 2–3 are treated as relevant and 0–1 as non-relevant.

retrieving mathematical information and identifying the limitations that motivate the design choices in this thesis.

2.2.1 Text-Based Models

Early MIR systems represented formulas as linearized strings (*e.g.*, LaTeX) and applied standard sparse retrieval techniques such as TF-IDF and BM25 [29, 40]. While these approaches allowed efficient lookup based on term frequency and document occurrence, surface-level token matching failed to capture structural relationships and semantic equivalences, causing missed matches for minor notational variations.

2.2.2 Tree-Based Sparse Formula Retrieval Models

Recognizing these limitations, researchers moved to structured representations of formulas. Tree-based models encode the hierarchical structure of math expressions and index subtrees to allow for partial matching. MathWebSearch (2012) [26] introduced OPT subexpression indexing with variable substitution, enabling retrieval of structurally equivalent formulas despite notational differences. MCAT (2013) [59] extended this with a part-based framework encoding hierarchical and sibling relationships in both SLTs and OPTs, while integrating formula structure with surrounding text through combined indexing.

The Tangent family further refined tree-based matching through increasingly sophisticated representations. Tangent-3 (2016) [64] introduced a two-stage model using SLT path tuples for candidate retrieval, followed by Maximum Subtree Similarity (MSS) scoring, a dynamic-programming method that identifies the largest common subtrees between query and candidate formulas. Subsequent variants combined SLT and OPT paths (Tangent-S, 2017) [8,34,38,66] and introduced richer indexing features with BM25+ scoring (Tangent-L, 2018) [10].

Approach0 (2019) [69] emerged as a leading tree-based system by focusing on operator tree structure. It represented formulas as OPTs and indexed all leaf-to-root paths, which encode the sequence of operators encountered when traversing from an operand up to the root, capturing which chain of functions is applied to that symbol. For example, in $\sin(x + 1)$, the path from x records that it is first an argument to addition and then to sine, encoding how x is functionally embedded in the expression. These paths were stored in a sparse inverted index mapping each path type to all formulas containing it, enabling efficient candidate retrieval by path intersection. A two-stage process combined this path matching with dynamic programming to identify largest common subtrees (similar to MSS), achieving strong performance on NTCIR-12. More recently, the Leaf-to-Leaf (L2L) system (2025) [60] improved on this by extracting shortest paths between leaf nodes in OPTs and combining structural features with LaTeX similarity and Presentation MathML

symbol matching in an ensemble approach that achieved top performance on ARQMath-3 and NTCIR-12.

> **Note 1:** OPTs and SLTs capture complementary aspects of formula meaning, motivating the use of both in MathAMR+'s unified graph.

While structure-based sparse retrieval models have demonstrated strong performance by capturing hierarchical and symbolic relationships in mathematical expressions, they face a fundamental limitation known as the *vocabulary problem*. Mathematically equivalent or semantically similar expressions with different structural forms may fail to be retrieved, even when they represent the same underlying concept. For example, $\frac{a}{b}$ and $a \cdot b^{-1}$ are algebraically equivalent but have entirely different tree structures. Queries containing tokens absent from the index vocabulary also pose challenges. This motivated a shift toward dense retrieval models.

2.2.3 Dense Retrieval Models

Dense retrieval models aim to alleviate the vocabulary problem by learning continuous vector representations where semantically similar symbols, subexpressions, or formulas are positioned close together regardless of their structural forms, based on their similar usage contexts. Documents are then retrieved and ranked using cosine similarity between query and document vectors, defined as $\cos(q, d) = (q \cdot d)/(\|q\|\|d\|)$, which measures the angle between two vectors and is high when they point in the same direction.

Early work applied word embedding techniques to mathematical content: SMSG5 (2016) [58] re-ranked sparse retrieval results using doc2vec embeddings of linearized MathML, and Gao et al. (2017) [12] proposed symbol2vec and formula2vec, extending Continuous Bag of Words (CBOW) [39] with negative sampling to learn embeddings for mathematical symbols and formula units.

EqEmbs (2018) [28] was the first embedding-based model to explicitly incorporate textual context. Unlike previous approaches that treated formulas in isolation, EqEmbs represented equations as tokens and jointly learned embeddings for words, formulas, and formula units (such as operators or variables). By using large context windows, EqEmbs connected formulas to the surrounding text, enabling embeddings to reflect both symbolic structure and semantic context. EqEmbs was thus among the first to demonstrate that jointly modeling formulas and their surrounding text, rather than treating them in isolation, is a viable and effective approach to mathematical retrieval.

Subsequent work aimed to combine the structural awareness of tree-based methods with the semantic flexibility of dense embeddings. Tangent-CFT (2019) [37] linearized OPT and SLT paths into sequences of tuples, treating each tuple as a word-like unit and applying FastText n-gram embeddings. By averaging tuple embeddings, the model produced formula vectors that captured both

subexpression structure and symbol relationships. Tangent-CFT demonstrated that embeddings could simultaneously encode structure and enable approximate matching, though its best results came from combining these embeddings with Approach0's tree-based structural matching. The results obtained suggested that purely vector-based similarity remained less structurally precise than explicit tree comparison for highly similar formulas, but was more effective at retrieving partially similar expressions that tree-based methods missed entirely.

The potential of graph neural networks (GNN) for mathematical retrieval emerged with Semantic Search (2020) [45], which directly encoded formula graphs using convolutional message passing. By training with masked symbol prediction and contextual similarity objectives, the model learned embeddings reflecting both local operator relationships and global structure. EARN (2021) [1] extended this approach by combining OPTs and SLTs enriched with reverse edges to create undirected graphs, then processing both graph structure and rendered formula images through a ResNet [18] and bi-directional LSTM [16]. A learned linear combination of these structural and visual embeddings provided complementary information, demonstrating how multiple representations of the same formula can improve retrieval effectiveness for dense models.

MathBERT (2021) [44] represented a major step toward jointly modeling formulas and text. Rather than simply concatenating formula and text tokens, MathBERT modified Transformer attention masks to respect OPT connectivity, constraining self-attention to follow formula structure rather than treating all tokens as equally related. Pretraining objectives, including Masked Language Modeling, Context Correspondence Prediction, and Masked Substructure Prediction, allowed the model to learn lexical, contextual, and structural semantics simultaneously. MathBERT's strongest results still came from combining its embeddings with Approach0's structural matching, showing that even sequence-based transformers with structure-aware attention benefit from explicit tree comparison to capture fine-grained structural equivalence.

MathAMR (2022) [35] introduced a fundamentally different approach by unifying formula structure with sentence-level semantic representations. The model generates an Abstract Meaning Representation (AMR) for the sentence containing a formula, then substitutes each formula with its corresponding OPT connected via explicit `:math` edges. Because OPTs encode operator-operand hierarchy, their structure is roughly analogous to the predicate-argument structure in AMR graphs, making this integration conceptually coherent. As shown in Figure 1.2, this creates a unified graph encoding both textual semantics and formula structure. However, MathAMR then linearized these graphs through depth-first traversal before encoding them with Sentence-BERT [48], forcing structural reasoning to emerge implicitly through attention rather than through the graph topology. Moreover, the model only considered sentence-level context, omitting spatial layout information (SLTs) and broader document context.

Recent work in formula retrieval has moved toward explicit graph reasoning with relation-

aware architectures. Amador et al. (2025) [4] applied contrastive learning on Relational Graph Convolutional Networks (RGCNs) over SLTs and OPTs augmented with backward edges, using filtered message passing across edge types to capture operator-level semantics and visual layout simultaneously. Their ColBERT-style multi-vector approach enabled fine-grained query-candidate interactions, achieving competitive performance on both NTCIR-12 and ARQMath-3. However, their model operates purely on formula structure, without any integration of surrounding textual context.

Note 2: MathAMR and RGCN-based formula retrieval are complementary: this thesis applies RGCN encoding and ColBERT-style MaxSim retrieval directly to MathAMR-style unified text-formula graphs.

2.2.4 Math-Aware Search Models

Math-aware search systems must handle both formulas and their surrounding text, creating a challenge in aligning the two modalities of information in a way that supports meaningful retrieval. A survey by Zanibbi et al. [65] identified two main paradigms: *federated search* and *unified search*.

Federated search models process text and formulas with separate uni-modal models before combining retrieval results. Early systems such as MathWebSearch (2014) [17] and MIaS (2018) [55] constructed separate inverted indexes for text and formula tokens, using boolean filtering or rank combination. Later systems explored score-level fusion: MathDowsers (2020) [42] combined Tangent-L formula scores with BM25+ text scores, while MaRec (2023) [13] employed Borda Count voting over KL-divergence and tree-edit distances. More recently, Zhong et al. (2023) [68] combined Approach0's structural matching with ColBERT's contextualized text embeddings, incorporating dense retrieval alongside sparse methods.

However, federated search has a fundamental limitation: by indexing formulas and text separately, these systems lose the contextual relationships between mathematical notation and surrounding language that help disambiguate meaning and establish relevance. Unified search aims to address this by creating joint representations where formulas and text coexist in a shared index. The math search system for the Digital Library of Mathematical Functions (DLMF) [40] was among the first to treat formula tokens (represented as LaTeX) and text tokens identically within a single inverted index, scoring sources with a modified TF-IDF approach. MIaS (2018) [55] also supported this approach by combining canonicalized formula tuples with text in a single index.

The emergence of transformer-based language models created new opportunities for unified dense retrieval. Initial attempts to fine-tune pre-trained models like BERT, ColBERT, and ALBERT on math question-answer pairs showed promise but struggled to capture the structural

relationships inherent in mathematical expressions, as their attention mechanisms treated formula tokens as arbitrary sequences rather than meaningful hierarchical compositions [50]. Analysis showed that transformers failed to recognize variable correspondences between questions and answers, suggesting that standard language model architectures were insufficient for mathematical content without explicit structural encoding [49].

This motivated the development of math-specific architectures. MathPredictor (2021) extended BERT's vocabulary with mathematical tokens to prevent LaTeX command over-segmentation, enabling more coherent formula representations. MathBERT, as discussed earlier, went further by incorporating OPT structure directly into attention and introducing math-specific pretraining objectives. The MABOWDOR system (2023) [67] took a hybrid approach, using PyA0 for formula canonicalization while applying unified dense passage retrieval with DPR, Approach0 for structural matching, and SPLADE for text retrieval. Top results from each component of the hybrid system were combined using convex linear interpolation. MABOWDOR's success on ARQMath answer retrieval demonstrated the effectiveness of integrating more information representations by combining federated and unified components of both formula structure and surrounding text.

Note 3: Unified representations that encode formulas and text jointly can capture cross-modal interactions that federated approaches miss, motivating MathAMR+'s single heterogeneous graph over AMR and formula subgraphs.

Large language models have opened new directions for math-aware search. Recent work has explored using models like GPT-4 and specialized math LLMs such as ToRA to generate answer text from questions, then using these generated answers as retrieval queries [52]. This query transformation helps bridge format gaps between questions and answers in the collection, potentially improving retrieval by making queries more closely match the structure and content of the target documents. Kassaie et al. (2025) [23] took a complementary approach, using LLM responses for query term selection, augmentation, and re-weighting within the MathDowsers BM25 system, then combining multiple reformulation strategies via Reciprocal Rank Fusion (RRF) to achieve consistent gains on ARQMath-3 Task 1 (Answer Retrieval).

2.2.5 Summary

MIR has progressed from simple token-based systems to sophisticated models that integrate symbolic structure and semantic correlation. Text-based methods were fast but lacked structural understanding; tree-based sparse models added structural precision but struggled with notational variation; dense embeddings better captured semantic similarity, though they tend to benefit from combination with explicit structural representations. Two complementary lines of recent work are

particularly relevant to this thesis. MathAMR established a unified graph representation linking formula structure to natural language semantics, but discarded its topology through linearization and was limited to sentence-level context. Amador et al. demonstrated that RGCN encoding with multi-vector MaxSim retrieval is effective for formula graphs, but operated purely on formula structure without any textual context. This thesis proposes applying RGCN-based encoding directly to MathAMR-style unified text-formula graphs, extending both the representational scope of MathAMR and the encoding methodology of Amador et al.

2.3 Abstract Meaning Representation (AMR)

The approach proposed in this thesis builds upon MathAMR, which uses Abstract Meaning Representation (AMR) as its underlying graph representation for modeling natural language. This section provides relevant background on AMR, including its core concepts and representation structure.

Abstract Meaning Representation (AMR) is a graph-based framework that represents the meaning of a sentence as a rooted, directed graph in which nodes encode concepts (events, entities, predicates) and edges encode semantic relations such as roles, modifiers, or logical operators. AMR abstracts away surface word order, morphology, and many syntactic artifacts, producing a normalized structure that captures logical predicate-argument relationships central to the sentence's meaning [33]. For example, the sentence "The student solved the equation" would be expressed as a graph rooted at `solve-01` with outgoing edges linking to `student` as `:ARG0` and `equation` as `:ARG1`.

AMRs have been used in tasks such as summarization, question answering, and semantic similarity because they offer structured and interpretable representations of sentence meaning, providing a form of normalization based on inferred logical semantics rather than surface form. As with many areas of NLP, the introduction of Transformer architectures led to substantial performance gains in AMR-based systems. Today, Transformer models are the most commonly used and generally the most effective approach, though GNNs have also been used in encoder-decoder architectures for AMR parsing and paraphrase generation [56].

Most AMR parsers are trained at the sentence level, producing a separate graph for each sentence without explicitly modeling cross-sentence relationships. To capture document-level coherence, coreference resolution is needed to identify repeated mentions of the same entity and form coreference chains. This can be handled either as a separate pipeline stage using an independent coreference system [41] or jointly within an end-to-end model [11]. This is illustrated in Figure 2.4.

In MathAMRs, embedding OPTs as subgraphs within the semantic structure of AMRs allows mathematical expressions to be interpreted in the context of surrounding language rather than as isolated formulae. Coreference resolution can be extended to link repeated variables and symbol

Original Text

"Alice bought a book. She read it."

AMR Graph Visualization

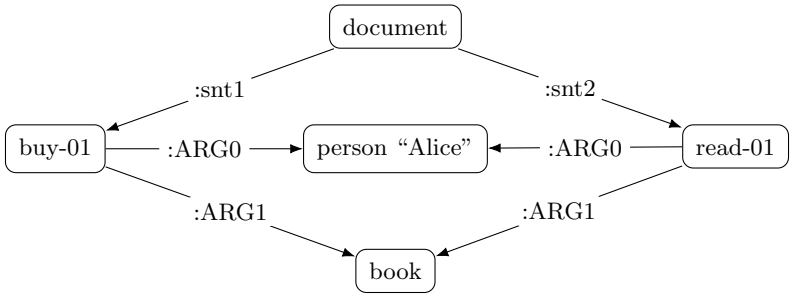


Figure 2.4: Multi-sentence AMR parsing with coreference resolution for the text "Alice bought a book. She read it." Coreference is captured by shared nodes: the `person` node is the `:ARG0` of both `buy-01` and `read-01`, and `book` is the `:ARG1` of both, reflecting that "She" and "it" refer back to Alice and the book respectively.

mentions across both text and formulas.

2.4 Graph Neural Networks

Outside of MIR, graph-structured data appear in many domains, including molecular modeling, social networks, and traffic prediction. Unlike sequences or grids, graphs encode relational structure as edges connecting nodes which cannot be directly processed by standard neural architectures that expect fixed-dimensional inputs. Graph Neural Networks (GNNs) address this limitation by operating over graph topologies directly, where at each layer, a node transforms and accumulates feature vectors from its immediate neighbors (neighborhood aggregation), and this process repeats across multiple layers so that a node's representation eventually reflects information from progressively wider regions of the graph (iterative local computation) [15, 25]. GNNs have been widely adopted for various tasks over graphs, including node classification, link prediction, and graph-level property prediction.

2.4.1 Message Passing

The foundational concept underlying most GNN variants is message passing [15]. Each node i in a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is associated with a feature vector $h_i^{(0)}$, typically derived from input features or learned embeddings. At each layer l, a node collects messages from its neighbors, aggregates them,

and updates its own representation accordingly. This process can be expressed in general form as:

$$m_i^{(l)} = \bigoplus_{j \in \mathcal{N}_i} \phi\left(h_j^{(l-1)}, h_i^{(l-1)}, e_{ji}\right), \tag{2.1}$$

$$h_i^{(l)} = \psi\left(h_i^{(l-1)}, m_i^{(l)}\right), \tag{2.2}$$

where $\mathcal{N}_i$ denotes the set of neighbors of node i, e_{ji} is an optional edge feature vector, ϕ is a message function that computes the contribution of each neighbor, $\bigoplus$ is a permutation-invariant aggregation operator (*e.g.*, sum, mean, or max), and ψ is an update function that combines the aggregated message with the node's current representation. After L layers of message passing, each node's representation $h_i^{(L)}$ encodes information from its L-hop neighborhood (all nodes reachable within L edges). Specific GNN architectures differ in their choices of ϕ, $\bigoplus$, and ψ.

2.4.2 Graph Convolutional Networks

Graph Convolutional Networks (GCNs), introduced by Kipf and Welling [25], generalize the convolutional operation from grid-structured inputs to irregular graph topologies. Rather than aggregating over a two-dimensional spatial window, a GCN layer aggregates over each node's neighborhood as defined by the graph structure. The per-node propagation rule is:

$$h_i^{(l)} = \sigma\left(\sum_{j \in \mathcal{N}_i \cup \{i\}} \frac{1}{\sqrt{\tilde{d}_i\, \tilde{d}_j}}\, W^{(l)}\, h_j^{(l-1)}\right), \tag{2.3}$$

where $\tilde{d}_i = 1 + |\mathcal{N}_i|$ is the degree of node i in the self-loop-augmented graph, $W^{(l)} \in \mathbb{R}^{d^{(l)} \times d^{(l-1)}}$ is a learnable weight matrix, and σ is a nonlinear activation function such as ReLU applied over a vector. The symmetric normalization factor $(\tilde{d}_i\, \tilde{d}_j)^{-1/2}$ ensures that messages are scaled by the degrees of both sender and receiver, preventing high-degree nodes from dominating the aggregation. Self-connections are handled implicitly by including i in the summation. Viewed through the lens of equations 2.1 and 2.2, GCN uses a weighted sum aggregation with degree-based normalization and a linear update with shared weights across all edges.

2.4.3 Relational Graph Convolutional Networks

While effective and computationally efficient, GCNs treat all neighbors equally and operate over a single, homogeneous edge type. Many real-world graphs are heterogeneous, where edges carry distinct typed relations. In SLTs, for instance, edges encode different spatial relationships (e.g., next, above, below). The Relational Graph Convolutional Network (R-GCN), proposed by Schlichtkrull et al. [53], extends GCN to multi-relational graphs by introducing relation-specific transformations.

Given a set of relation types $\mathcal{R}$, the R-GCN update for node i at layer l is:

$$h_i^{(l)} = \sigma\left(W_0^{(l)}\, h_i^{(l-1)} + \sum_{r \in \mathcal{R}} \sum_{j \in \mathcal{N}_i^r} \frac{1}{|\mathcal{N}_i^r|}\, W_r^{(l)}\, h_j^{(l-1)} \right), \tag{2.4}$$

where $\mathcal{N}_i^r$ is the set of neighbors of i connected by relation r, $W_r^{(l)} \in \mathbb{R}^{d^{(l)} \times d^{(l-1)}}$ is a relation-specific weight matrix, and $W_0^{(l)}$ is an explicit self-loop weight matrix. The normalization factor $|\mathcal{N}_i^r|^{-1}$ averages contributions within each relation type, analogous to the degree normalization in GCN.

The key property of R-GCN is that each relation type induces a distinct linear transformation. A message arriving along one type of edge is processed differently from a message arriving along another. This is critical in heterogeneous graphs where conflating edge semantics would obscure meaningful structural distinctions. Since the number of parameters grows linearly with $|\mathcal{R}|$, Schlichtkrull et al. [53] propose two regularization strategies. Basis decomposition ($W_r^{(l)} = \sum_{b=1}^{B} a_{rb}^{(l)}\, V_b^{(l)}$) expresses each relation matrix as a weighted sum of B shared basis matrices $V_b^{(l)}$, so relations share structure and only the scalar coefficients $a_{rb}^{(l)}$ are relation-specific. Block-diagonal decomposition instead constrains each $W_r^{(l)}$ to be block-diagonal, partitioning the feature space into independent subspaces and reducing the per-relation parameter count without imposing cross-relation sharing. Both approaches trade off parameter efficiency against the model's ability to represent independent relation-specific transformations.

Note 4: Relational message passing preserves semantic distinctions across edge types, motivating MathAMR+'s use of an RGCN to jointly encode AMR relations, OPT operator-operand edges, and SLT spatial layout edges within a single model.

2.4.4 Virtual Nodes for Long-Range Message Passing

A well-known limitation of message passing GNNs is that information propagation is bounded by the number of layers: after L layers, each node can only represent information within its L-hop neighborhood (its *receptive field*). Increasing L to widen the receptive field often leads to over-squashing, where the exponentially many nodes at large distances must all route their information through the same fixed-size bottleneck vector, meaning each individual distant node contributes only an exponentially small fraction to the final representation [21, 57].

Virtual nodes [15] address this by introducing auxiliary nodes that connect to every "real" node in the original graph (or a designated subset) and act as information relays. At each message passing layer, a virtual node aggregates information from its connected nodes and broadcasts an updated summary back, reducing the effective distance between any pair of nodes to two hops.

Because each virtual node exchanges exactly one message with each real node, this incurs only $O(|\mathcal{V}|)$ additional cost per layer, in contrast to the $O(|\mathcal{V}|^2)$ cost of full pairwise attention. Formally, the virtual node update is:

$$h_v^{(l)} = \psi_{\text{vn}}\left(h_v^{(l-1)}, \bigoplus_{j\in\mathcal{V}} h_j^{(l-1)}\right), \tag{2.5}$$

where $\mathcal{V}$ denotes the set of all real nodes connected to the virtual node, $\bigoplus$ is a permutation-invariant aggregation, and ψ_{vn} is typically a one-hidden-layer MLP applied to the concatenation of the previous virtual node state and the aggregated summary, with a residual connection adding the input back to the output. The virtual node representation is then injected into each real node $h_i^{(l)}$ after the local message passing step:

$$h_i^{(l)} \leftarrow h_i^{(l)} + \text{MLP}_{\text{vn}}^{(l)}\left(h_v^{(l-1)}\right). \tag{2.6}$$

Virtual nodes are conventionally initialized as zero vectors so they carry no prior information and build their representations entirely through message passing [15]. This ensures the global summary is derived purely from the graph content. Karabulut et al. [21] later explored trainable virtual node embeddings that are learned during training, showing that the initialization strategy can meaningfully affect downstream performance.

Southern et al. [57] provide a theoretical analysis of how virtual nodes reduce over-squashing, showing that the improvement in information mixing depends on the spectrum of the underlying graph (the eigenvalues of its Laplacian matrix, which characterize how quickly information diffuses across the graph's topology). They also highlight a limitation of standard virtual node implementations: all nodes are treated as equally important when forming the global summary, unlike Graph Transformers which can assign different importance scores through attention. When virtual nodes are used with attention-based GNN layers such as a Relational Graph Attention Network (RGAT) [5], the attention mechanism can partially mitigate this limitation by learning to weight neighbor contributions differently during the aggregation step.

Several extensions to the basic virtual node framework have been proposed. Karabulut et al. [21] introduce local virtual nodes placed at high-centrality nodes where over-squashing bottlenecks are most likely, improving connectivity while preserving the original graph structure. Clementi et al. [6] propose RANGE, which replaces the single fixed-size virtual node with multiple attention-based master nodes for richer long-range communication while retaining this linear cost. Pham et al. [46] use virtual nodes for graph classification, demonstrating that virtual node representations can serve as effective graph-level summaries for downstream tasks beyond message passing facilitation.

Note 5: Virtual nodes provide efficient long-range aggregation at $O(|\mathcal{V}|)$ cost per layer. MathAMR+ extends this with a hierarchy of formula-, sentence-, and document-level virtual nodes, producing a multi-vector representation at multiple levels of granularity for MaxSim retrieval.

2.5 Contrastive Learning for Dense Retrieval

Dense retrieval systems learn to map queries and documents into a shared embedding space where relevance is measured by vector similarity. A common training paradigm for learning such embeddings is contrastive learning, which encourages representations of relevant ("positive") pairs to be similar while pushing apart representations of unrelated ("negative") pairs.

2.5.1 Contrastive Objectives

The InfoNCE loss [43], originally proposed for representation learning, has become a standard training objective for dense retrieval. Given a query q, a positive document d^+, and a set of negative documents $\{d_j^-\}$, the loss treats retrieval as a binary classification problem over candidates:

$$\mathcal{L}_{\text{NCE}} = -\log \frac{\exp(\text{sim}(q, d^+)/\tau)}{\exp(\text{sim}(q, d^+)/\tau) + \sum_j \exp(\text{sim}(q, d_j^-)/\tau)}, \tag{2.7}$$

where $\text{sim}(\cdot, \cdot)$ is a similarity function (typically cosine similarity) and τ is a temperature parameter that controls the sharpness of the distribution.

A practical challenge is obtaining sufficient negative examples. Dense Passage Retrieval (DPR) [22] introduced in-batch negatives, where the positive documents of other queries in the same training batch serve as negatives. For a batch of B query–document pairs, this yields $B - 1$ negatives per query without additional computation, as the document embeddings are already computed. The quality of these negatives depends on batch size: larger batches provide harder negatives on average and have been shown to improve retrieval quality [22].

2.5.2 Single-Vector and Multi-Vector Retrieval

DPR and similar systems [22] represent each query and document as a single vector, enabling efficient retrieval via approximate nearest neighbor search. However, compressing an entire document into a single vector can lose fine-grained information.

ColBERT [24] introduced multi-vector retrieval through late interaction. Rather than producing a single embedding, ColBERT represents each query and document as a set of token-level

embeddings and computes relevance using the MaxSim operator. Let $Q \in \mathbb{R}^{|q| \times k}$ and $D \in \mathbb{R}^{|d| \times k}$ be the ℓ_2-normalized token embedding matrices for a query and document. The MaxSim score is:

$$S(Q, D) = \sum_{i=1}^{|q|} \max_{1 \leq j \leq |d|} q_i^\top d_j, \tag{2.8}$$

where q_i and d_j are individual token embeddings. Each query token is matched to its closest document token, and the resulting similarities are summed. This allows fine-grained token-level matching without requiring a fixed alignment between query and document tokens. This preserves token-level detail while remaining more efficient than full cross-attention, as document embeddings can be precomputed and indexed.

The MaxSim paradigm has been extended beyond text. In the context of mathematical retrieval, Amador et al. [4] applied ColBERT-style multi-vector matching to graph neural network embeddings of formula trees, using node-level embeddings rather than token-level embeddings as the vectors in the MaxSim computation. Their results demonstrated that multi-vector retrieval with GNN-produced embeddings can achieve competitive performance on formula retrieval benchmarks.

2.6 Summary

In the current landscape of MIR research, a central challenge lies in building representations that capture the interplay between natural language and mathematical information, and in designing systems that can effectively learn over these representations. The concepts surveyed here, including relational graph encoding, virtual node mechanisms, contrastive learning, and multi-vector retrieval, form the technical foundation for the system design and experimental questions addressed in the following chapter.

Chapter 3

Methodology

This chapter presents the MathAMR+ system in detail, covering graph construction, the hierarchical virtual node architecture, the RGCN-based encoder, the contrastive training procedure, and the retrieval pipeline. The source code is made available online[1].

3.1 Graph Construction

The proposed graph representation builds on MathAMR and combines the inferred logical structure of sentences (natural language meaning) with mathematical structure in a single heterogeneous graph. The pipeline converts raw Math Stack Exchange posts, containing HTML text and math expressions, into directed graphs suitable for GNNs. Applied to the ARQMath collection, this process covers approximately 2.47 million posts (1.02 million questions and 1.45 million answers). It consists of three main steps: extracting formulas and inserting placeholders, parsing the text into AMR, and integrating formula subgraphs into the AMR graph.

Each ARQMath post contains formulas inside `<span class="math-container">` tags, each with a unique identifier. In the extraction step, these spans are located, their positions and IDs are recorded, and the formulas are replaced with placeholders such as `FORMULA_0`, `FORMULA_1`, etc. For question posts, the title is added before the body text to provide more context. This produces clean text for parsing, along with a mapping from placeholders to their original formulas. The formula extraction code is adapted from the original MathAMR implementation[2] [35].

The cleaned text is then parsed using the Transition AMR Parser[3] with the pretrained `doc-sen-conll-amr-seed42` model, producing graphs that capture sentence structure, semantic roles, coreference, and named entities. For long posts, the text is split into sentence-based chunks

[1] `https://gitlab.com/jy9726/mathamr-plus`
[2] `https://github.com/BehroozMansouri/MathAMR`
[3] `https://github.com/IBM/transition-amr-parser`

of up to 500 tokens. Each chunk is parsed separately, and the resulting AMR graphs are merged into a single graph. This is done by adding a root node with edges (`:snt1`, `:snt2`, etc.) pointing to each chunk, while renaming variables to avoid conflicts.

Parsing is parallelized across multiple GPU workers. If parsing fails for a batch, a binary search fallback splits the batch into smaller parts until the problematic input is isolated, allowing the rest to succeed.

After constructing the AMR graph, each `FORMULA_N` placeholder is replaced with a structured math subgraph. For each formula, OPT and SLT representations are retrieved from a pre-built SQLite database; these representations were extracted from MathML using the Tangent-S codebase [8]. These are converted into graph triples and inserted into the AMR graph. Each formula node is connected using two edges: `:opt` (to the OPT root) and `:slt` (to the SLT root), with their full subgraphs added to the graph. If a placeholder is missing from the parsed AMR (e.g., dropped during parsing due to noise), its formula is instead attached directly to the root using a `:math` edge, ensuring no formula information is lost. Each serialized graph record also stores a `formula_map`, mapping placeholder names to their corresponding ARQMath formula identifiers, which is used at Task 2 retrieval time to locate the query formula's virtual node embedding.

Figure 3.1 provides a concrete example of the resulting MathAMR+ graph, demonstrating how logical structure from AMR is enriched with complementary OPT and SLT representations to capture both meaning and mathematical form. The complete set of constructed graphs is serialized in Penman format and released publicly[4] to facilitate reproducibility and future work.

Prior to encoding, we augment each graph with inverse edges for every original edge to facilitate bidirectional message passing. Each inverse relation is given a distinct type by appending a `*` suffix to the original relation name (e.g., `:ARG0` produces `:ARG0*`) [1, 4], effectively doubling the relation set. These inverse edges allow information to flow in both directions during message passing.

Note 6: OPT and SLT subgraphs are inserted into the AMR graph via typed edges, allowing the GNN to leverage complementary formula representations through relational message passing over the graph structure.

3.2 Hierarchical Virtual Nodes

To address the over-squashing problem of GNNs and provide multi-granularity representations for retrieval, MathAMR+ introduces a hierarchy of virtual nodes that are appended to the graph before encoding.

[4] `https://drive.google.com/file/d/1EpgfwvEGl6yoL6gaTXhl0gmzM8NXNulL/view?usp=sharing`

Topic A.346

Greatest lower bound in Q

I have a set $\{r \in \mathbb{Q} \mid r^2 > 2, r > 0\}$. I was wondering why it does not have the greatest lower bound. Isn't $0 \in \mathbb{Q}$ a greatest lower bound for this set in rational numbers?

Full MathAMR+ Graph

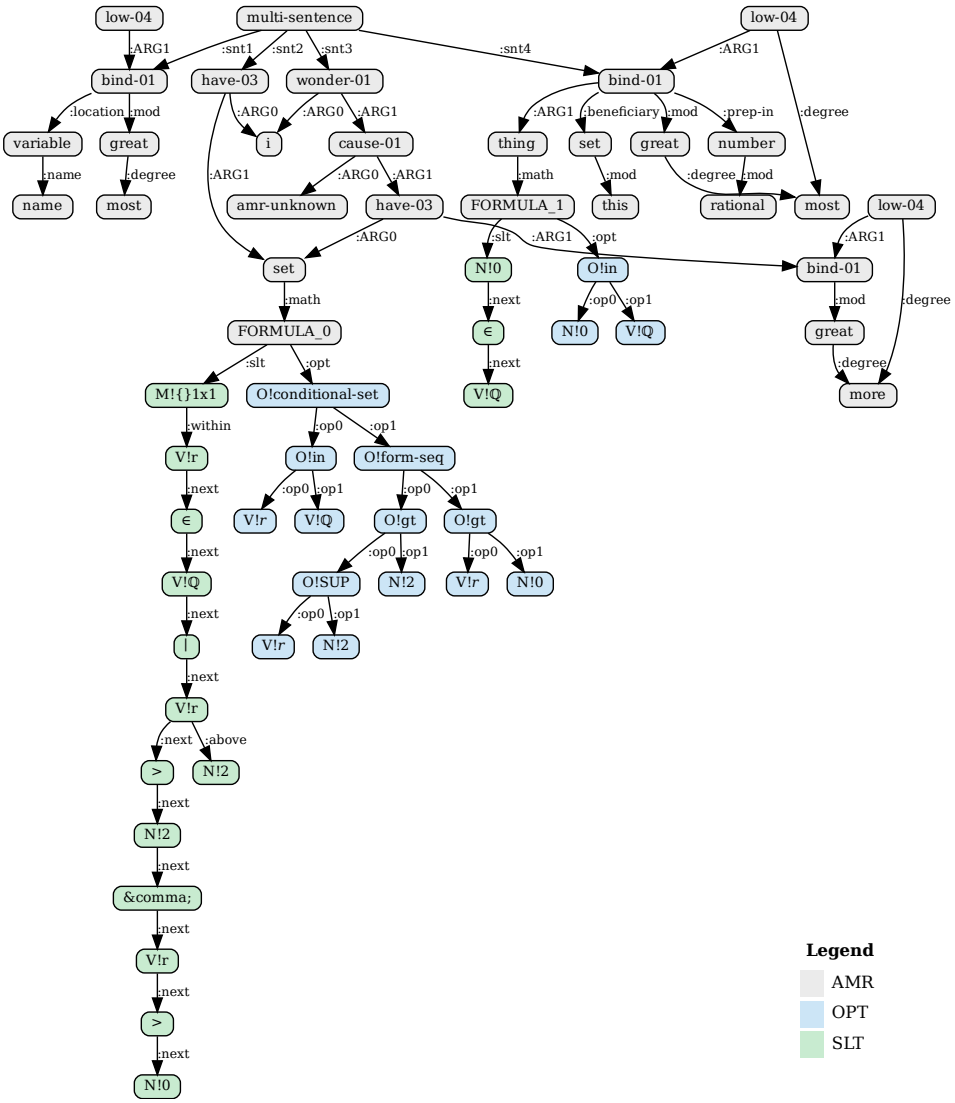


Figure 3.1: MathAMR+ graph for Topic A.346 in the ARQMath-3 test collection. The AMR `multi-sentence` node serves as the root, with sentence subgraphs as children. Each formula is identified by a unique placeholder (*e.g.*, `FORMULA_0`), with its OPT and SLT attached via `:opt` and `:slt` edges. The two formula representations are complementary: for `FORMULA_0` ($\{r \in \mathbb{Q} \mid r^2 > 2, r > 0\}$), the OPT groups the two conditions under a `conditional-set` operator, while the SLT encodes the same formula as a left-to-right symbol sequence within a brace-matrix container.

Formula virtual nodes. For each formula in the document, a `VIRTUAL_FORMULA` node is created and connected via `:virtual_formula` edges (with corresponding inverse edges) to every node belonging to that formula's OPT and SLT subgraphs. This provides a single aggregation point for each formula's structural and layout information.

Sentence virtual nodes. For each sentence in the document, a `VIRTUAL_SENTENCE` node is connected to all AMR concept nodes in that sentence via `:virtual_sentence` edges. This provides a summary of the sentence-level textual semantics.

Document virtual node. A single `VIRTUAL_DOCUMENT` node is connected to all formula and sentence virtual nodes via `:virtual_document` edges, as well as to the root node of the AMR graph. This node serves as a global summary of the entire document, similar to the `[CLS]` token in transformer-based models. The document virtual node is always the last virtual node added to the graph.

This hierarchical structure ensures that, after message passing, each virtual node aggregates information at its designated level of granularity: individual formulas, individual sentences, or the entire document. By first consolidating formula and sentence meanings, the virtual nodes encourage a structured flow of information upward. The document-level virtual node aggregates and produces a global embedding that reflects both mathematical notation and textual information. Together, the set of virtual nodes in a graph forms the document's multi-vector representation used for retrieval. An example of this unified graph representation and virtual nodes setup is shown in Figure 3.2. Section **??** takes this further and investigates the impact of adding additional structural edges to this virtual node hierarchy in various configurations (Virtual Node Connectivity experiment).

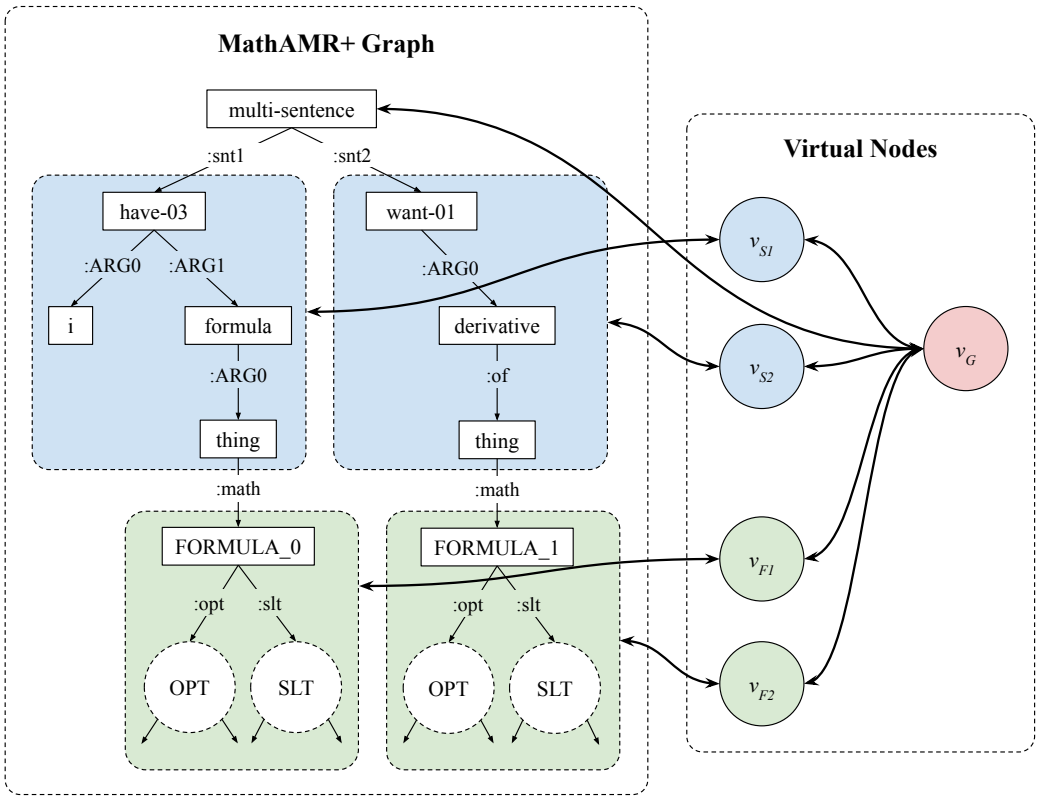
Note 7: Hierarchical virtual nodes provide formula-, sentence-, and document-level aggregation points, composing a multi-vector representation where each vector captures a distinct granularity of the document's content.

Once the virtual node hierarchy is in place, the complete graph is preprocessed offline into a PyTorch Geometric (PyG) [9] `Data` object, comprising a node feature index tensor, an edge index tensor, and an edge attribute tensor. The resulting tensors are serialized as binary blobs and stored in a SQLite database (`preprocessed.db`), keyed by document identifier, supporting lazy per-document loading during training without holding the full corpus in RAM. Different preprocessing configurations (e.g., formula representation mode, virtual node connectivity) can be stored as separate databases without regenerating the source graphs. With these graph tensors in place, the following section describes the model architecture that encodes them into retrieval embeddings.

Original Question

"I have the formula $f(x) = x^2 + 1$. What is the derivative of $f(x)$?"

MathAMR+ Graph with Hierarchical Virtual Nodes

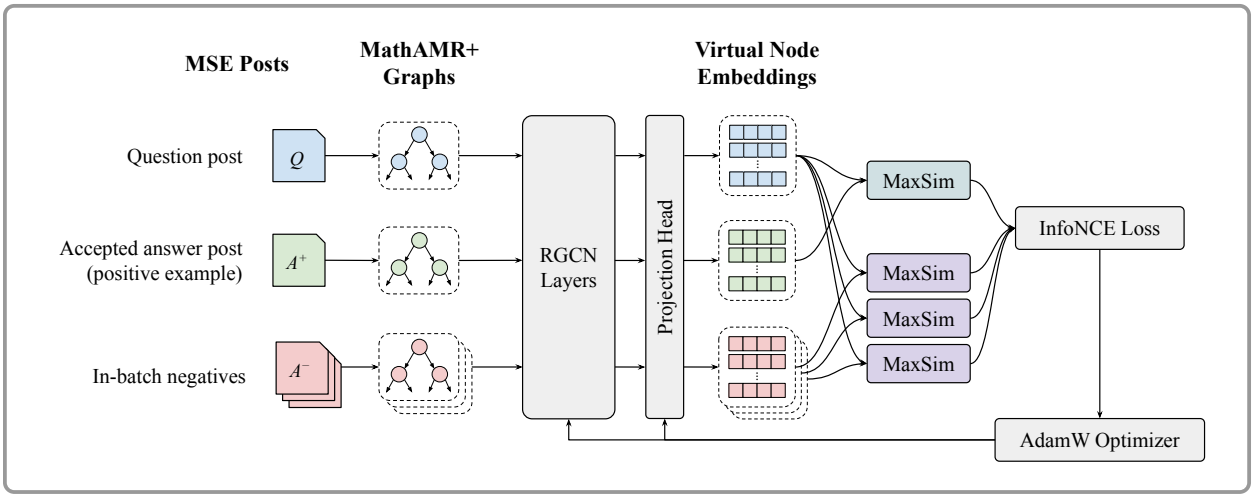


Colored regions with dotted borders delineate sentence and formula subgraphs. Virtual nodes v_{S_1}, v_{S_2} (sentence-level) and v_{F_1}, v_{F_2} (formula-level) connect bidirectionally to all nodes within their respective subgraphs, while the document-level virtual node v_G aggregates information from the document node and sentence and formula virtual nodes. Backward edges are also created for each "real" edge, but are omitted from the diagram for visual simplicity.

Figure 3.2: MathAMR+ representation with hierarchical virtual nodes for a simple mathematical question.

3.3 Model Architecture

This section describes the model architecture. The model takes a MathAMR+ graph as input, encodes it through a stack of RGCN layers, and projects virtual node embeddings into a retrieval space for multi-vector matching. Figure 3.3 provides a visual overview.



Posts are encoded as graphs with virtual nodes, processed through the RGCN layers, and trained using contrastive learning with MaxSim similarity and InfoNCE loss.

Figure 3.3: Overview of the proposed MathAMR+ training pipeline.

3.3.1 Node Feature Initialization

Before being passed through the RGCN, each node in the graph must be mapped to an initial feature vector. A vocabulary is constructed from all unique node types observed across the training corpus, including AMR concepts (e.g., `solve-01`, `person`), OPT operators and operands (e.g., `U!plus`, `V!x`), and SLT layout symbols. Each node type is assigned an index in a learned embedding lookup table of dimension d, initialized with Xavier-uniform initialization. Nodes whose type does not appear in the vocabulary are mapped to a shared `UNK` embedding.

Several preprocessing normalizations are applied to node types before embedding. Node types are normalized using NFKC (Normalization Form Compatibility Composition) to collapse Unicode compatibility variants to their ASCII equivalents. URL-like node types (containing `http` or `www.`) are collapsed to a single `URL` token. AMR concept nodes are Porter-stemmed [47] to reduce residual morphological variation. PropBank frame nodes (e.g., `solve-01`) are already normalized by the parser, so stemming primarily affects plain noun and adjective concepts and non-canonical forms that the parser occasionally produces for rare or ambiguous words.

Numeric nodes are handled specially. In both natural language and mathematical formulas, numbers carry meaning, but many values appear only rarely. Assigning each rare number its own embedding would greatly expand the vocabulary with entries that cannot be learned well. In NLP, digit-level tokenization is often used, which decomposes each number into a sequence of character tokens and is used in sequence-to-sequence models. However, this relies on positional information and doesn't directly suit our graph neural network architecture which relies on single

node embeddings. Wallace et al. [61] show that models with dedicated embeddings for frequent numeric values outperform those that rely on subword decomposition for numeracy.

We instead use a technique where numbers that appear at least $\tau = 5$ times across the training graphs receive their own embeddings. Rare numeric values are mapped to one of three type-specific tokens: integers to `[UNKNUM_INT]`, decimal and scientific-notation values to `[UNKNUM_DEC]`, and rationals to `[UNKNUM_RAT]`. This preserves strong representations for common values like 1, 2, and 3.14, while still encoding useful type information for rare numbers. This normalization applies to both AMR text nodes and formula operand nodes (prefixed `N!`).

Virtual node embeddings are initialized separately from content nodes. Our default strategy initializes each virtual node with a random Xavier-uniform vector as a learned embedding. Alternative initialization strategies, including zero initialization, aggregation-based initialization (mean-pooling over constituent subgraph nodes), and learned projection layers, are investigated in Section 4.3.4.

3.3.2 RGCN Encoder

The core encoder consists of L stacked RGCN layers, as defined in Section 2.4.3. At each layer l, the update for node i is:

$$h_i^{(l)} = h_i^{(l-1)} + \mathrm{GELU}\Big(\mathrm{RGCN}^{(l)}\Big(\mathrm{LayerNorm}\Big(h_i^{(l-1)}\Big), \mathcal{N}_i, \mathcal{R}\Big)\Big), \tag{3.1}$$

where $\mathcal{N}_i$ is the set of neighbors of node i, $\mathcal{R}$ is the set of relation types in the graph, and $\mathrm{RGCN}^{(l)}$ denotes the RGCN convolution at layer l as defined in Equation 2.4. LayerNorm normalizes each node's feature vector to zero mean and unit variance across its d feature dimensions (distinct from ℓ_2 normalization, which would rescale the vector to unit magnitude). GELU is the Gaussian Error Linear Unit activation [19], defined as

$$\mathrm{GELU}(x) = x \cdot \Phi(x), \tag{3.2}$$

where Φ is the cumulative distribution function of the standard normal distribution. It acts as a smooth gating function that scales inputs by their cumulative probability, softly emphasizing larger values. The RGCN convolution operates over these LayerNorm-normalized representations of node i and its neighbors. Each RGCN layer also receives learned edge attribute vectors from a separate edge embedding table, providing additional capacity for distinguishing relation types beyond the relation-specific weight matrices.

The pre-normalization residual design improves optimization stability in deep GNN stacks by ensuring that the residual path preserves an unnormalized identity signal, allowing gradients to propagate without being affected by normalization.

Intuitively, LayerNorm stabilizes node representations before message passing by keeping feature scales consistent across layers. The RGCN then performs relation-aware aggregation over $\mathcal{N}_i$, enabling different edge types to contribute via distinct parameters. GELU provides a smooth gating mechanism that softly weights features based on magnitude, suppressing noise while retaining informative signals. Finally, the residual connection preserves the input representation, allowing each layer to refine rather than overwrite node states and supporting long-range information flow.

3.3.3 Projection Head

After L layers of message passing, virtual node embeddings are extracted from the encoded graph and projected into a lower-dimensional embedding space through a two-layer MLP projection head:

$$z_i = \text{LayerNorm}\Big(W_2\,\text{GELU}\Big(\text{LayerNorm}\Big(W_1\,h_i^{(L)} + b_1\Big)\Big) + b_2\Big)\,, \tag{3.3}$$

where $W_1 \in \mathbb{R}^{d\times d}$, $W_2 \in \mathbb{R}^{d_p\times d}$, b_1, b_2 are shared parameters applied to each virtual node embedding, and d_p is the projection dimension. The projection head serves two purposes: it provides a learned nonlinear mapping from the encoder's hidden space to the embedding space where similarity is computed, and it allows the encoder's hidden dimension d to differ from the embedding dimension d_p. This separation is motivated by the observation that contrastive learning benefits from computing the loss in a lower-dimensional projected space while maintaining a richer internal representation [7].

For a document with n_f formulas in n_s sentences, the final multi-vector representation consists of up to $n_f + n_s + 1$ projected virtual node embeddings: one per formula, one per sentence, and one for the document.

3.4 Training

The model is trained to retrieve relevant answers given a query question post. The training data consists of question–accepted answer pairs extracted from Math Stack Exchange: each question post that has an accepted answer forms a positive pair, and questions without accepted answers are excluded from training. This yields a fairly clean supervision signal, as accepted answers are explicitly marked by the question asker as satisfactory. While single-vector retrieval with graph-level embeddings offers efficiency, multi-vector retrieval has been shown to achieve superior performance by preserving fine-grained representations of document content [4]. This thesis adopts a ColBERT-style multi-vector approach that balances retrieval quality with index scalability by leveraging the virtual node embeddings.

3.4.1 Graph Batching

Training requires batching graphs of varying sizes into a single tensor for efficient GPU computation. Following the standard approach in PyTorch Geometric [9], graphs within a batch are combined into a single disconnected graph by concatenating their node feature matrices, edge index tensors, and edge attribute vectors, with node indices offset to prevent cross-graph edges. Virtual node embeddings are padded to the maximum number of virtual nodes in the batch, and a boolean mask tracks which positions correspond to real virtual nodes versus padding. This mask is used during MaxSim computation and loss calculation to ensure that padded positions do not contribute to similarity scores.

3.4.2 MaxSim Similarity

The model is trained using a contrastive learning objective that encourages questions and their accepted answers to have similar multi-vector representations while pushing apart representations of unrelated question–answer pairs. To compute multi-vector similarity, the MaxSim operation is used.

Let $Q \in \mathbb{R}^{|q| \times k}$ and $D \in \mathbb{R}^{|d| \times k}$ be the ℓ_2-normalized virtual node embedding matrices for a query and document, where k is the hidden dimension. Let $q_i \in \mathbb{R}^k$ and $d_j \in \mathbb{R}^k$ denote the corresponding node embeddings. The MaxSim score is:

$$S(Q, D) = \sum_{i=1}^{|q|} \max_{1 \le j \le |d|} q_i^\top d_j. \tag{3.4}$$

For each query virtual node, the maximum cosine similarity across all document virtual nodes is computed, and these maxima are summed. This allows fine-grained matching: a query formula virtual node may match most strongly with a document formula virtual node, while a query sentence virtual node may align with a document sentence virtual node, without requiring a fixed alignment. Note that $S(Q, D) \neq S(D, Q)$ because the summation runs over the first argument's nodes and the maximization over the second's.

In each training batch of size B, the model processes B question–answer pairs. Given a batch $\{(Q_i, D_i)\}_{i=1}^{B}$, the full $B \times B$ MaxSim similarity matrix S is computed, where entry S_{ij} gives the MaxSim score between query i and document j:

$$S \in \mathbb{R}^{B \times B}, \quad S_{ij} = S(Q_i, D_j) = \sum_{t=1}^{|q_i|} \max_{1 \le s \le |d_j|} (q_{i,t})^\top d_{j,s}. \tag{3.5}$$

3.4.3 InfoNCE Loss

The contrastive loss is based on the InfoNCE objective [43] using in-batch negatives. Given the $B \times B$ MaxSim similarity matrix S, the loss for the forward direction (queries matching to documents) is:

$$\mathcal{L}_{\text{NCE}} = -\frac{1}{B}\sum_{i=1}^{B}\log\frac{\exp(S_{ii}/\tau)}{\sum_{j=1}^{B}\exp(S_{ij}/\tau)}, \tag{3.6}$$

where τ is a temperature parameter that controls the sharpness of the softmax distribution. Lower temperatures produce sharper distributions that focus more heavily on the hardest negatives, while higher temperatures produce softer distributions. The temperature is parameterized as $\tau = \exp(t)$ where $t \in \mathbb{R}$ is a learnable scalar, ensuring $\tau > 0$ while allowing the model to adaptively adjust the contrastive sharpness during training.

Queries and documents have varying numbers of virtual nodes. Without normalization, documents with more formulas and sentences have higher MaxSim scores due to having more vectors to match against. To mitigate this bias, the MaxSim scores are normalized by the number of query virtual nodes:

$$\tilde{S}_{ij} = \frac{S_{ij}}{|q_i|}. \tag{3.7}$$

$$\mathcal{L}_{\text{MaxSim}} = \mathcal{L}_{\text{NCE}}(\tilde{S}). \tag{3.8}$$

3.4.4 Data Augmentation

To improve generalization and prevent overfitting, two stochastic augmentation strategies are applied to all training graphs.

Node masking. Each non-virtual node is independently replaced with a special `MASK` token with probability p_{mask} (default 0.1). Virtual nodes are never masked, as they serve as aggregation points and must remain intact to produce meaningful representations. This encourages the model to learn robust representations that do not depend on any single node's identity.

Edge dropout. Each edge is independently dropped with probability p_{drop} (default 0.1), with the constraint that edges connected to virtual nodes are never dropped. This preserves the virtual node hierarchy while introducing structural noise in the content subgraph, encouraging the model to learn representations that are robust to minor topological variations.

Both augmentation strategies are applied only during training; at inference time, the full unaugmented graph is used.

3.4.5 Optimization

The model is optimized using AdamW [32] with weight decay, and a cosine annealing learning rate schedule that decays from an initial learning rate to a minimum over the course of training. Gradient norms are clipped at a configurable maximum to prevent training instability.

The training pairs are randomly shuffled (with a fixed seed for reproducibility) and split into a training set and a validation set, using a configurable ratio (default 90%/10%). Early stopping monitors validation loss and halts training if no improvement is observed for a configurable number of consecutive epochs. The best checkpoint, as measured by validation loss, is saved for use at inference time. The specific hyperparameter values used are reported in the following chapter in Tables 4.5 and 4.18.

3.5 Indexing and Retrieval

At inference time, the model weights are fixed at the best training checkpoint. The trained model encodes all candidate documents and evaluation queries into multi-vector representations, which are indexed and searched using a Qdrant[5] vector database.

3.5.1 Document Encoding and Indexing

Each document in the retrieval collection is encoded by the trained model to produce its set of virtual node embeddings. These embeddings are stored in a Qdrant vector database using two named vector spaces per document. The first, referred to as `doc`, contains a single vector corresponding to the `VIRTUAL_DOCUMENT` embedding. The second, referred to as `all`, is a multi-vector field containing all virtual node embeddings, including formula-level, sentence-level, and document-level vectors.

For Task 2 (formula retrieval), a separate Qdrant collection is built alongside the document index. Rather than storing one embedding per document, each individual formula virtual node embedding is indexed as its own point, with the parent document identifier and the corresponding formula identifier stored as metadata attached to that point (referred to as a payload in Qdrant). Since each formula in a document has its own `VIRTUAL_FORMULA` node, a corpus with many formulas per document yields many more indexed points than the document-level index. To provide a sense of scale, a random sample of 100,000 ARQMath posts contains on average 8.7 formula virtual nodes, 6.6 sentence virtual nodes, and 1 document virtual node per document, totalling approximately 16.3 virtual nodes per document.

[5] `https://qdrant.tech/`

3.5.2 Retrieval Variants

The retrieval pipeline supports six configurable strategies for Task 1 (document retrieval), all operating over the same Qdrant database. The strategies vary along two independent dimensions: the *retrieval strategy* and the *search precision*.

Retrieval strategy. Three retrieval strategies are supported. *Single-vector* retrieval uses the `VIRTUAL_DOCUMENT` embedding to query the `doc` vector space, ranks candidates by cosine similarity, and returns results without any reranking step. *Doc-VN + MaxSim reranking* uses the `VIRTUAL_DOCUMENT` embedding to retrieve a pool of top-k candidate documents from the `doc` vector space in a first stage, where k controls the size of the candidate pool passed to reranking. Those candidates are then reranked in a second stage by computing exact MaxSim (Equation 3.4) over the full sets of query and candidate virtual node embeddings in the `all` multi-vector space. *ColBERT-style MaxSim* follows the ColBERT two-stage paradigm as implemented by Qdrant's native multi-vector search. In the first stage, each of the $|q|$ query virtual node embeddings independently retrieves candidate documents from the `all` vector space; Qdrant then aggregates the per-query candidate sets and rescores the union using exact MaxSim over the full multi-vector representations of each candidate. Unlike Doc-VN + MaxSim reranking, which uses only the `VIRTUAL_DOCUMENT` embedding as the first-stage filter, ColBERT-style MaxSim uses all query virtual nodes for candidate retrieval, providing broader coverage at the cost of issuing $|q|$ independent nearest neighbor queries.

Search precision. Each retrieval strategy can be run in either *approximate* or *exact* mode. In approximate mode, nearest neighbor queries are executed against Qdrant's HNSW index, where `hnsw_ef` controls the graph traversal breadth, with higher values trading speed for recall. We use `hnsw_ef` = 128, which recovers the large majority of exact results while reducing query time by orders of magnitude over a brute-force scan. For ColBERT-style MaxSim, this traversal is repeated once per query virtual node, so both latency and accumulated recall loss scale with $|q|$. In exact mode, Qdrant performs a brute-force linear scan over all indexed vectors, guaranteeing the true nearest neighbors and serving as the upper bound on retrieval accuracy. For Doc-VN + MaxSim reranking, search precision applies only to the first-stage candidate retrieval; the second-stage MaxSim reranking is always exact.

For Task 2 (formula retrieval), the query is a single formula virtual node embedding extracted from the topic post. Each topic in the ARQMath Task 2 collection specifies a particular formula of interest within a question post; the corresponding `VIRTUAL_FORMULA` embedding for that formula is identified via the formula map stored with the graph and used as the query vector. Section 4.4 discusses experiments that compare these retrieval variants.

Note 8: Storing single-vector and multi-vector embeddings in separate named Qdrant vector spaces enables flexible deployment: the same indexed collection supports efficient single-vector lookup, two-stage Doc-VN + MaxSim reranking, and full ColBERT-style MaxSim search without re-encoding the corpus.

Chapter 4

Graph Collection, Experiments, and Results

This chapter presents the graph processing results, our experimental framework, results, and analysis. We begin by characterizing the MathAMR+ graph corpus produced from the full ARQMath MSE collection, covering collection statistics, parsing outcomes, and graph characteristics. Then, we discuss the experiments that were conducted, covering the motivation and research question, the experimental conditions and hypotheses, the results, and the analysis of findings.

The experiments follow a deliberate progression. The first four investigate data representation and embedding: whether both formula tree types contribute complementary retrieval signals, whether the virtual node hierarchy itself is beneficial compared to simpler aggregation alternatives, how adding additional edges in the virtual node hierarchy affects information flow, and how virtual node initialization affects learned representations. The final two experiments shift focus to retrieval, examining the effectiveness-efficiency trade-off across retrieval strategies and whether an auxiliary training objective can close performance gaps for efficiency-oriented methods. We then benchmark the full system on ARQMath-3 Tasks 1 and 2, applying the configuration informed by the experiments, and close with a summary of key findings.

4.1 Graph Collection

The MathAMR+ pipeline was run over the full ARQMath MSE corpus. AMR parsing was parallelized across 4 NVIDIA A100 GPUs on RIT's Research Computing cluster [51] and took approximately 22 days of wall-clock time. Table 4.1 describes the cluster node used, and Table 4.2 summarizes the resulting collection.

Of 2,482,294 posts processed, 2,432,933 (98.0%) were successfully converted to MathAMR+

Table 4.1: RC cluster node used for graph construction.

Component	**Specification**
CPU	Intel Xeon Platinum 8558P (48 cores / 96 threads)
GPU	4× NVIDIA A100 (40 GB VRAM each)
RAM	256 GB

graphs. The 49,361 failures fell into two categories: 41,404 posts (83.9% of failures) in which all text chunks failed AMR parsing, and 7,946 posts (16.1% of failures) whose parsed AMR produced a disconnected graph, usually from the parser emitting structurally inconsistent output for long or noisy posts. An additional 20 posts were dropped during formula integration due to Penman decode errors, where special characters in formula identifiers (e.g., parentheses or colons) were carried into AMR variable names, producing malformed AMR strings that could not be re-parsed.

Table 4.2: MathAMR+ graph collection statistics over the full ARQMath MSE corpus.

Statistic	**Value**
Posts processed	2,482,294
Successful graphs	2,432,933 (98.0%)
Failed posts	49,361 (2.0%)
All chunks failed to parse	41,404
Disconnected graph	7,946
Penman decode error	20
Question graphs	1,009,372
Answer graphs	1,423,561
Training pairs (question + accepted answer)	545,638
Avg. nodes per graph (all)	156.0
Questions	143.8 nodes, 5.8 formulas
Answers	168.2 nodes, 6.9 formulas
Avg. edges per graph	160.1
Median nodes per graph	101
Vocabulary node types (raw)	411,742
Vocabulary node types (normalized + truncated)	20,000
Vocabulary relation types (raw)	262
Vocabulary relation types (total, including VN + inverse edges)	532

The raw, unprocessed vocabulary covers 411,742 unique node types (AMR concepts, OPT operators and operands, SLT layout symbols) and 262 relation types. After processing (normalization and truncation), we reduce the vocabulary to just 20,000 node types while retaining 99.8% coverage.

Adding 4 virtual node edge types (`:virtual_formula`, `:virtual_sentence`, `:virtual_document`, `:auxiliary_vn`) and inverse counterparts for each edge resulted in a total of 532 relation types used during training.

4.2 Experimental Setup

The primary testbeds are the ARQMath-3 Answer Retrieval and Formula Retrieval tasks (Tasks 1 and 2), which use posts from Math Stack Exchange. As outlined in Section 2.1.3, the metrics reported are nDCG'@1000, MAP'@1000, P'@10, and Bpref@1000.

In addition to retrieval effectiveness, efficiency metrics are recorded for each experiment, including training time per epoch, total model parameter count, GPU memory usage, index size, and query throughput. These measurements are important for assessing the feasibility of the proposed approach in practical deployment scenarios.

Table 4.3 summarizes the scale of the benchmark and corpus. The collection spans over 2.4 million posts containing 28 million mathematical formulas, evaluated against 78 topics for task 1 and 76 topics for task 2. Appendix A provides example posts and formulas at each relevance level for both tasks, concretely illustrating the distinctions between high, medium, low, and no relevance.

Table 4.3: ARQMath-3 benchmark and MathAMR+ corpus statistics.

Statistic	**Value**
MSE Collection	
Total posts	2,466,080
Question posts	1,020,585
Answer posts	1,445,495
Mathematical formulas	28,320,920
Evaluation Topics	
Task 1 topics	78
Task 2 topics	76
Relevance Judgments	
Task 1 total judgments	34,847
Task 1 unique judged answer posts	33,383
Task 2 total judgments	11,538
Task 2 unique judged formulas	9,975
MathAMR+ Corpus	
Training pairs (question + accepted answer)	545,638
Task 1 retrieval index (answer posts)	1,365,254
Task 2 retrieval index (question+answer posts)	2,058,341

All experiments are conducted at reduced scale and a randomly sampled training subset rather than the full corpus due to computational constraints. The full-scale benchmark against prior systems is presented in Section 4.5. Table 4.4 lists the configuration information of the workstation used for training, indexing, and retrieval for the experiments. The hyperparameters for the scaled-down model are listed in Table 4.5, which were chosen based on early runs and the memory constraints of this hardware.

Table 4.4: Workstation used for experiments and benchmarking.

Component	**Specification**
CPU	Intel Core i7-9700KF (8 cores / 8 threads)
GPU	2× NVIDIA GeForce GTX 1080 Ti (11 GB VRAM each)
RAM	32 GB

Table 4.5: Hyperparameters used for experiments.

Category	**Hyperparameter**	**Value**
Model	Hidden dimension	64
	Projection dimension	32
	Number of layers	4
	Number of bases	64
	VN init strategy	Zero learned
Training	Batch size	32
	Number of epochs	7
	Learning rate	1×10^{-4}
	Minimum learning rate	1×10^{-5}
	Weight decay	1×10^{-3}
	Initial temperature	0.05 (learnable)
	Gradient clipping norm	1.0
	Early stopping patience	3 epochs
Augmentation	Node mask probability	0.1
	Edge dropout probability	0.1
Loss	Score normalization	Enabled
Data	Train/validation split	90%/10%
	Formula representations	Both (SLT + OPT)
	Auxiliary virtual node edges	None
Retrieval	Retrieval mode	ColBERT-style MaxSim

Models were trained for 7 epochs, as we have found the model to typically converge by around

epoch 5 to 6. After each epoch, the model is evaluated on a held-out validation set, and the validation loss is used as the primary criterion for monitoring training progress. The training loop tracks the best observed validation loss and counts the number of consecutive epochs without improvement. If the validation loss does not decrease for 3 epochs, training is terminated early.

Each experiment is run with 5 random seeds. For each results table, two views are presented. The primary table reports results for the best seed per strategy, defined as the seed achieving the highest nDCG' on the evaluation set. We report the best seed as it reflects a single well-trained model for each configuration. Statistical significance is assessed on these best-seed results using a two-tailed paired t-test over the 78 per-query scores. Each test compares one condition against the designated baseline on one metric. Because multiple such tests are conducted within a single experiment (one per metric per non-baseline condition), we apply Bonferroni correction, where the significance threshold $\alpha = 0.05$ is divided by the total number of tests m conducted within that experiment, and a comparison is considered significant only if its raw p-value falls below α/m. Conditions that are statistically significantly different from the designated baseline after this correction are marked with an asterisk (*) in the best-seed tables.

A second table then reports the mean ± standard deviation over all 5 seeds for each strategy. This averaged table characterizes variance and training stability but is not used for significance testing.

4.3 Graph Representation and Embedding Experiments

The following three experiments examine how graph representation and embedding initialization choices affect the quality of learned embeddings for retrieval.

4.3.1 Formula Representations

This experiment measures the relative contribution of the two formula tree representations, OPT and SLT, to retrieval effectiveness. As discussed in Chapter 2, combining multiple complementary representations of mathematical content has consistently been shown to improve retrieval quality in prior work. For example, Tangent-S [8] demonstrated this by pairing SLT-based layout paths with OPT-based semantic paths, and Tangent-L [10] further refined the approach by combining structural and lexical features. The intuition in both cases is that OPT and SLT capture different aspects of a formula, and that queries may match on either dimension depending on the nature of the information need.

MathAMR+ integrates both representations into a unified graph, and this experiment directly tests whether this choice is justified. Four conditions are compared: both OPT and SLT (the default

configuration), OPT-only, SLT-only, and no formula representation. In the no-formula condition, mathematical expressions are excluded entirely, leaving only the textual AMR graph.

All parameters other than the formula mode are held fixed at the values listed in Table 4.5. All conditions are trained on 50,000 positive pairs (45,000 train / 5,000 validation) over 5 random seeds, with retrieval evaluated using ColBERT-style MaxSim.

Hypotheses

Using both representations is expected to yield the best performance because they capture complementary aspects of mathematical notation. OPT-only may slightly outperform SLT-only because semantic similarity is expected to be a stronger signal for mathematical relevance than visual similarity; two formulas can appear different in layout but be semantically equivalent. Removing formulas entirely is expected to produce the largest quality drop, confirming that mathematical formula structure is essential for math-aware retrieval and that text-only AMR representations are insufficient for this domain. The gap between the no-formula condition and any formula-enabled condition should be larger than the gap between single-representation and dual-representation modes.

Results and Analysis

Table 4.6 reports best-seed results. Both (OPT+SLT) achieves the highest score on all four metrics, with nDCG' of 0.497, MAP' of 0.150, P'@10 of 0.217, and Bpref of 0.228. OPT-only and SLT-only score close behind (0.496 and 0.495 nDCG' respectively), and the None condition is substantially worse on all metrics. For the three formula-enabled conditions, the improvement over None are statistically significant on all metrics after Bonferroni correction.

Table 4.6: Answer retrieval performance by formula representation mode (best seed per strategy).

Formula Mode	**nDCG'**	**MAP'**	**P'@10**	**Bpref**
Both (OPT+SLT)	**0.497***	**0.150***	**0.217***	**0.228***
OPT-only	0.496*	0.143*	0.199*	0.222*
SLT-only	0.495*	0.147*	0.196*	0.213*
None†	0.404	0.106	0.167	0.183

†Baseline, *$p < 0.05$ (Bonferroni corrected)

Using both formula representations (OPT+SLT) achieves the highest score on all four reported metrics in the averaged results as well (Table 4.7), achieving a mean nDCG' of 0.494 ± 0.002 versus 0.488 ± 0.006 for SLT-only and 0.486 ± 0.007 for OPT-only. The lower variance for Both compared to OPT-only suggests more stable training when combining representations. This aligns with findings

Table 4.7: Answer retrieval performance by formula representation mode (mean ± std over 5 seeds).

Formula Mode	nDCG'	MAP'	P'@10	Bpref
Both (OPT+SLT)	$\mathbf{0.494 \pm 0.002}$	$\mathbf{0.147 \pm 0.004}$	$\mathbf{0.211 \pm 0.010}$	$\mathbf{0.224 \pm 0.004}$
OPT-only	0.486 ± 0.007	0.141 ± 0.004	0.198 ± 0.010	0.217 ± 0.008
SLT-only	0.488 ± 0.006	0.141 ± 0.005	0.197 ± 0.010	0.217 ± 0.005
None	0.392 ± 0.009	0.098 ± 0.005	0.155 ± 0.011	0.177 ± 0.006

from prior symbolic retrieval systems where dual representations consistently outperformed single representations [4, 8, 10], and suggests that the RGCN layers can exploit complementary signals from both tree types simultaneously. OPT-only and SLT-only each recover much of the available signal individually but fall short of the combined representation.

OPT-only and SLT-only achieve nearly identical performance under both evaluations. No pairwise comparison between them is significant. This parity is perhaps surprising given that OPTs encode mathematical semantics more explicitly, but one interpretation is that the RGCN message-passing layers extract similar structural information from either tree type. Another factor is that the LaTeX-to-OPT mapping is approximate, so it does not always capture operational information perfectly.

Removing formula nodes entirely (None) causes a drop of 0.093 nDCG' relative to Both (OPT+SLT) on the best seed, substantially larger than any difference among the formula-enabled conditions. This confirms that formula content is a crucial source of retrieval signal in this setting and that even a single formula representation recovers most of the available gain.

Table 4.8: Efficiency metrics by formula representation mode.

Formula Mode	Time/Epoch (s)	GPU Mem (GB)	Index Size (MB)
Both (OPT + SLT)	472.5 ± 2.8	52.3 ± 3.0	145.6
OPT only	362.4 ± 0.9	41.3 ± 2.8	145.6
SLT only	340.9 ± 1.0	36.3 ± 1.2	145.6
None	262.1 ± 1.7	24.5 ± 1.1	58.3

Table 4.8 reports efficiency metrics, and shows that combining both representations requires additional computational cost, requiring approximately 30% more training time and 44% more GPU memory than SLT-only, and 30% more time and 27% more memory than OPT-only. Among single-representation conditions, SLT-only is both faster and more memory-efficient than OPT-only (340.9s and 36.3 GB vs. 362.4s and 41.3 GB).

This efficiency gap is likely influenced by structural differences between the representations. OPTs tend to contain more nodes than SLTs on average (approximately 16% more in our collection),

as they encode implicit operations explicitly. For example, $3x^2$ is represented with three nodes in an SLT but five in an OPT due to the inclusion of multiplication and exponentiation operators. SLTs are also typically deep with long sequential chains of symbols, whereas OPTs are more highly branching with internal nodes usually having two children. OPT nodes therefore have higher average degree, requiring aggregation over more neighbors during RGCN message passing. These factors slightly increase computational overhead and gradient complexity during training for OPTs.

Given that the performance gains from combining both representations are consistent but modest (approximately 0.005–0.013 across metrics), the overall benefit-to-cost ratio is limited when training at this scale. In resource-constrained settings, SLT-only provides the most favorable trade-off, achieving comparable effectiveness to OPT-only while maintaining the lowest computational cost among formula-enabled conditions.

Example Results

While aggregate metrics provide a useful quantitative comparison, they do not fully capture how OPT and SLT contribute distinct signals during retrieval. To examine this more concretely, we consider retrieval results for Topic A.346. The topic and example retrieved results for each model are shown in Appendix B.

The query asks why the set $\{r \in \mathbb{Q} \mid r^2 > 2, r > 0\}$ has no greatest lower bound in $\mathbb{Q}$, and whether 0 qualifies as one. The set contains only positive rationals whose square exceeds 2, so its infimum is $\sqrt{2}$, which is irrational and absent from $\mathbb{Q}$. Answering correctly requires recognizing both the conditional structure of the set and the incompleteness of $\mathbb{Q}$. As visible in the MathAMR+ graph in Figure 3.1, the OPT representation captures this structure through the `O!conditional-set` operator, which explicitly groups the two constraints $r^2 > 2$ and $r > 0$. The SLT representation encodes the same expression as a flat symbol sequence within a brace-matrix container, preserving its visual layout rather than its logical structure.

Without either representation, the system mainly retrieves general explanations of infimum and related definitions, with limited focus on the specific structure of the query set. SLT alone improves retrieval of documents that match the surface form of the query (for example, sets involving $\mathbb{Q}$ and inequalities), but it can also retrieve results where the inequality condition is changed, which alters the meaning of the set. OPT alone more often retrieves results related to bounds and completeness properties of $\mathbb{Q}$, but it does not always match the exact constraint in the query, and may retrieve related results based on similar supremum or inequality arguments.

The combined representation retrieves results that are related to both the topic and structure of the query in this example. However, it still does not fully reflect finer distinctions in the mathematical setting, such as the fact that the relevant bound is irrational and therefore not contained in $\mathbb{Q}$. Overall, these results suggest that semantic structure (OPT) and visual or symbolic structure

(SLT) provide different types of information that can both be useful for this type of mathematical retrieval task.

Having established that both formula representations contribute meaningfully to retrieval, the following experiments treat the dual-representation configuration as fixed and investigate how other design choices affect model behavior.

4.3.2 Virtual Node Structure

With the inclusion of both OPT and SLT formula representations supported by empirical results, an important next question is whether the hierarchical virtual node structure introduced in Section 3.2 is necessary for learning effective aggregate representations for retrieval, or if simpler aggregation strategies are equally effective. The proposed hierarchy introduces formula, sentence, and document virtual nodes, each aggregating information at different granularities, with the document node capturing the full post. This design is motivated by the multi-granularity nature of mathematical retrieval, where relevance may arise at different levels, and MaxSim over the resulting embeddings enables flexible matching. However, this hierarchy also introduces additional design complexity, raising the question of whether its benefits justify the added overhead.

This experiment compares four conditions. The primary independent variable is the virtual node graph structure, with a second dimension of the training objective and retrieval strategy used (*i.e.*, whether multi-vector MaxSim or single-vector retrieval is used). The *full hierarchy* condition uses the complete structure of formula, sentence, and document virtual nodes, trained with a MaxSim objective and with MaxSim matching over all virtual nodes at retrieval time. The *no virtual nodes* condition removes the hierarchy entirely, using `FORMULA_N` placeholder nodes and AMR sentence root nodes as the MaxSim representative set, with no explicit aggregation nodes in the graph. The *global VN (MaxSim)* condition adds a single global node v_G connected to every node in the graph, and includes v_G alongside the `FORMULA_N` and AMR sentence root nodes as the MaxSim representative set for both training and retrieval. Comparing this condition against no-VN therefore reflects both the structural contribution of v_G to message passing quality and the addition of a global aggregate vector to the retrieval representative set. Finally, the *global VN (single-vector)* condition uses the single v_G in the graph, but trains with a standard single-vector InfoNCE objective on v_G and uses only v_G for retrieval.

A possible fifth condition would be MaxSim over every node in the graph with no representative selection. However, posts in our corpus contain hundreds to thousands of nodes depending on document length and formula density, making per-node indexing and MaxSim computation extremely computationally expensive. This condition is therefore excluded.

All parameters other than the independent variable (virtual node structure) are held fixed at the values listed in Table 4.5. The formula representation mode is fixed at Both (OPT+SLT).

All conditions are trained on 50,000 positive pairs (45,000 train / 5,000 validation) over 5 random seeds.

Hypotheses

The full hierarchy is expected to outperform all alternatives, as it provides structured, multi-granularity representations that MaxSim can exploit selectively. The no-VNs condition is expected to perform worst: without any aggregation node, the model relies entirely on message passing to develop discriminative embeddings in the formula and sentence root nodes, which may be insufficient at this scale. Global VN + MaxSim is expected to improve over no-VNs by giving message passing a global aggregation channel and by adding v_G itself as an additional retrieval representative, providing a document-level summary vector that MaxSim can match against. Global VN (single-vector) is expected to fall below the full hierarchy but above or near no-VNs: it has the same aggregation node as global VN + MaxSim, but collapses all granularity into a single embedding and trains without any MaxSim objective. The gap between global VN + MaxSim and no-VNs reflects the combined benefit of the global aggregation node in message passing and as a retrieval representative, while the gap between the full hierarchy and global VN + MaxSim quantifies the contribution of the structured multi-granularity hierarchy itself.

Results and Analysis

Table 4.9 reports the best-seed retrieval performance for each condition.

Table 4.9: Answer retrieval performance by virtual node structure (best seed per strategy).

VN Structure	Retrieval	nDCG'	MAP'	P'@10	Bpref
Full hierarchy†	MaxSim	0.461	**0.145**	**0.221**	**0.235**
No VNs	MaxSim	0.368*	0.109*	0.183	0.228
Global VN	MaxSim	0.459	0.129	0.199	0.217
Global VN	Single-vector	**0.486**	0.134	0.182	0.229

†Baseline, *$p < 0.05$ vs. baseline (paired t-test, Bonferroni corrected)

The results largely confirm the hypotheses. The full hierarchy achieves the highest MAP', P'@10, and Bpref. Unlike global VN + MaxSim, which uses a global virtual node alongside formula and sentence root nodes as retrieval representatives, the full hierarchy replaces these concrete nodes with a structured set of formula, sentence, and document virtual nodes. Each virtual node is an explicitly dedicated aggregation point at a specific granularity, so MaxSim can align query vectors against representations that are purpose-built for formula-level, sentence-level, or document-level matching rather than against concrete nodes or a single flat aggregate.

The no-VNs condition performs worst on nDCG', MAP', and P'@10. Global VN + MaxSim differs from no-VNs in two ways, adding v_G as a global virtual node connected bidirectionally to every concrete node in the graph and including v_G in the MaxSim representative set. The bidirectional edges make v_G a global communication hub through which all nodes can exchange information during message passing, improving the representations of the formula and sentence root nodes that are also in the representative set. The additional v_G vector in the MaxSim set further provides a document-level summary that no-VNs lacks entirely. The consistent improvement of global VN + MaxSim over no-VNs on nDCG', MAP', and P'@10 reflects this combined benefit, and the gap is statistically significant on nDCG' and MAP' ($p < 0.05$ after Bonferroni correction, best-seed table).

The global VN single-vector condition produces the highest nDCG' across all conditions, a surprising deviation from the hypothesis. nDCG' measures the quality of the overall ranking over the top 1000 retrieved results weighted by graded relevance and position, and the simplicity of the single-vector objective appears to make the model particularly effective at ordering documents globally. However, MAP' and P'@10, which measure the proportion of relevant documents retrieved and the density of relevant documents among the top 10 returned results respectively, remain below the full hierarchy. Collapsing all document structure into a single embedding limits the model's ability to retrieve the full breadth of relevant answers, even while it produces a well-calibrated global ranking. The full hierarchy therefore provides the strongest and most consistent retrieval performance overall.

With the virtual node structure confirmed, a natural follow-on question concerns the edges within that hierarchy. The full hierarchy connects sentence and formula virtual nodes only indirectly, through the document virtual node or through the underlying graph topology. The next experiment tests whether adding auxiliary edges between virtual nodes to enable more direct cross-component information flow improves retrieval.

4.3.3 Virtual Node Connectivity

The default virtual node hierarchy provides no direct connections between sentence and formula virtual nodes. Information flows between them only indirectly, via the document virtual node or through the underlying graph topology. This experiment tests whether inserting auxiliary edges between sentence and formula virtual nodes improves retrieval by enabling more direct cross-component information flow.

Three auxiliary edge configurations are compared against the no-auxiliary-edge baseline. Sequential edges connect adjacent virtual nodes in document order, creating a chain. Circular edges extend the sequential pattern by additionally connecting the last virtual node back to the first. Fully connected edges add bidirectional connections between every pair of sentence and formula

virtual nodes.

The formula representation mode is fixed at Both (OPT+SLT) and the virtual node initialization is fixed at Zero/Embedding. All hyperparameters other than the independent variable (auxiliary virtual node edges) are held fixed at the values listed in Table 4.5. All conditions are trained on 50,000 positive pairs (45,000 train / 5,000 validation) over 5 random seeds, with retrieval evaluated using ColBERT-style MaxSim.

Hypotheses

Sequential auxiliary edges are expected to improve retrieval by enabling discourse-level context flow between adjacent sentences and formulas. Circular edges may offer a slight advantage over sequential edges by providing more robust flow throughout each post. In contrast, fully connected edges are likely to underperform because they obscure the linear discourse structure and introduce quadratic edge count growth, which may dilute the relational convolution mechanism by averaging over a large, unstructured neighborhood. The no-auxiliary-edge baseline may already perform well because the document virtual node provides a global aggregation point, and multiple layers of message passing enable indirect information flow. There is also a risk of representation collapse with many auxiliary edges, where virtual nodes become too similar to each other and lose their ability to capture distinct aspects of the document.

Results and Analysis

Table 4.10 reports best-seed results. *None* achieves the highest scores on all four metrics (0.431 nDCG', 0.131 MAP', 0.212 P'@10, 0.229 Bpref), with the three auxiliary edge configurations performing below this baseline. However, no comparison against *None* reaches statistical significance after Bonferroni correction.

Table 4.10: Answer retrieval performance by auxiliary edge configuration (best seed per strategy). No comparisons are statistically significant vs. the *None* baseline ($p < 0.05$, Bonferroni corrected paired t-test).

Aux. Edges	nDCG'	MAP'	P'@10	Bpref
None[†]	**0.431**	**0.131**	**0.212**	**0.229**
Sequential	0.423	0.121	0.194	0.218
Circular	0.422	0.125	0.205	0.213
Fully Connected	0.420	0.118	0.181	0.214

[†]Baseline

This observation is consistent across seeds (shown in Table 4.11), where every condition that

adds auxiliary edges between virtual nodes falls below *None* on every averaged metric (*None* achieves 0.421 ± 0.008 nDCG' versus 0.409–0.417 for the other configurations).

Table 4.11: Answer retrieval performance by auxiliary edge configuration (mean ± std over 5 seeds).

Aux. Edges	**nDCG'**	**MAP'**	**P'@10**	**Bpref**
None	$\mathbf{0.421 \pm 0.008}$	$\mathbf{0.125 \pm 0.004}$	$\mathbf{0.205 \pm 0.014}$	$\mathbf{0.216 \pm 0.008}$
Sequential	0.409 ± 0.011	0.118 ± 0.004	0.187 ± 0.009	0.204 ± 0.008
Circular	0.409 ± 0.008	0.119 ± 0.004	0.188 ± 0.011	0.205 ± 0.006
Fully Connected	0.417 ± 0.004	0.119 ± 0.004	0.190 ± 0.013	0.215 ± 0.002

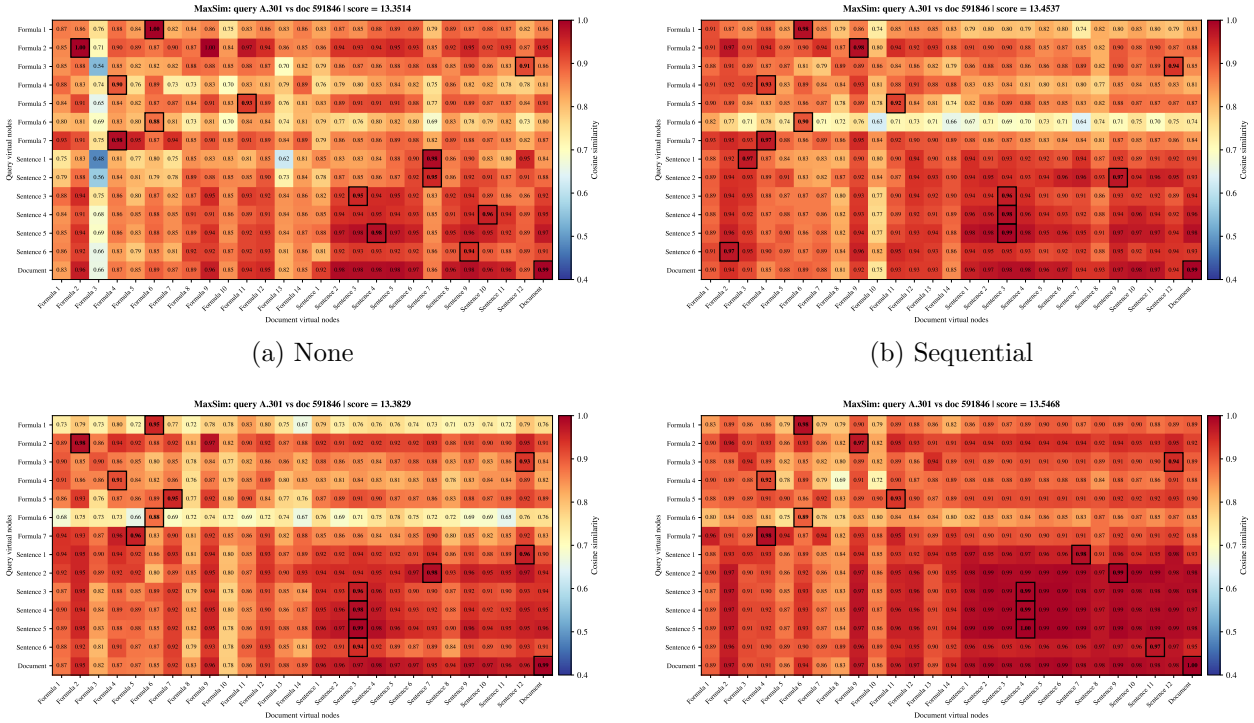
These results refute the primary hypotheses. Neither *Sequential* nor *Circular* auxiliary edges improve over *None*. While we expected that linear connections between adjacent virtual nodes would enable discourse-level context flow, both configurations uniformly underperform. *Sequential* and *Circular* perform nearly identically across all four metrics, providing no evidence that the additional circular edge benefits information flow as hypothesized.

The hypothesis that *Fully Connected* would underperform relative to *Sequential* and *Circular* is refuted. On the best seed, *Fully Connected* (0.420 nDCG') falls below both Sequential (0.423) and Circular (0.422), but the averaged results show *Fully Connected* to have slightly higher average nDCG' than the other auxiliary edge configurations (0.417 ± 0.004).

The results suggest that the document virtual node already serves as an effective global aggregation point, and with 4 layers of message passing, information flow between virtual nodes via the document virtual node and the underlying graph structure may already be sufficient.

Figure 4.1 visualizes heatmaps that show cosine similarity between the virtual nodes of a topic query (vertical axis) and those of a document (horizontal axis). In the *None* heatmap, the score matrix is visually heterogeneous, where different query virtual nodes produce distinctly peaked similarity patterns over different document virtual nodes. The *Sequential* and *Circular* heatmaps are broadly similar but begin to show smoother, less peaked distributions. The *Fully Connected* heatmap is the most visually uniform, with almost no visible discrimination particularly within sentence virtual nodes. They show that when auxiliary edges connect sentence and formula virtual nodes directly, message passing propagates representations across nodes that should remain distinct, causing them to converge toward similar embeddings. This erodes the per-level specificity that multi-vector MaxSim matching depends on, where different virtual nodes should capture different aspects of the document so that query virtual nodes can selectively match against the most relevant parts. This supports the concern raised in the hypotheses regarding representation collapse of virtual nodes caused by auxiliary edges.

Table 4.12 shows that *Sequential* and *Circular* add negligible overhead ($\approx 3\%$ more time and



(a) None

(b) Sequential

(c) Circular

(d) Fully Connected

Figure 4.1: Comparison of MaxSim score distributions (between Topic A.301 and document 591846) across four virtual node connectivity structures. The heatmaps visualize the cosine similarity between query and document VNs. Note how the fully connected structure (d) exhibits less discrimination between virtual nodes, suggesting a degree of representation collapse where nodes become overly similar and lose the ability to capture distinct aspects of the document.

$\approx 2\%$ more memory than *None*), while *Fully Connected* incurs a meaningful cost ($\approx 28\%$ more time and $\approx 24\%$ more memory). Given that auxiliary edges consistently degrade retrieval effectiveness at this scale, the *None* configuration is preferred on both grounds.

Table 4.12: Efficiency metrics by auxiliary edge configuration.

Structure	**Time/Epoch (min)**	**GPU Mem (GB)**
None	10.471 ± 0.011	5.752 ± 0.693
Sequential	10.767 ± 0.011	5.848 ± 0.705
Circular	10.833 ± 0.006	5.852 ± 0.706
Fully connected	14.506 ± 0.013	7.368 ± 0.964

With the virtual node structure and connectivity fixed, the remaining open question on the graph construction side is how virtual node embeddings are seeded before the first message-passing step.

4.3.4 Virtual Node Embedding Initialization

With the formula representation, virtual node structure, and connectivity established, attention turns to how virtual node embeddings are initialized before the first message-passing step. The initial embedding values determine the starting state from which RGCN layers must propagate and refine representations, and this choice may influence both convergence speed and the quality of the final virtual node embeddings.

Two orthogonal dimensions are varied. The first is the initial embedding value, which can be set to zero (all virtual nodes start as zero vectors), random (Xavier-uniform initialization), or aggregation-based (mean-pooling over the constituent nodes of each virtual node's subgraph). The second dimension is whether the initialization includes a learnable component: either treating the vector as a learned embedding, or adding a linear projection layer that transforms the aggregated representation. These two dimensions are crossed to produce the full set of conditions, with the zero initialization and no learned component serving as the baseline.

All parameters other than the virtual node initialization strategy are held fixed at the values listed in Table 4.5. All conditions are trained on 50,000 positive pairs (45,000 train / 5,000 validation) over 5 random seeds, with retrieval evaluated using ColBERT-style MaxSim.

Hypotheses

The hypothesis for the initial value is that aggregation-based initialization may outperform both zero and random initialization because it provides a structurally informed prior that aligns virtual node embeddings with their subgraph content before the first forward pass, reducing the burden on the RGCN layers to propagate information into virtual nodes from scratch. Zero initialization will likely underperform random initialization because it creates a symmetry problem in which all virtual nodes start identically, requiring more training epochs to differentiate. For the learnable component dimension, the hypothesis is that adding a learned embedding or projection will improve retrieval metrics, as it allows the model to learn task-specific offsets or transformations beyond what structural aggregation alone provides.

Results and Analysis

Table 4.13 shows the retrieval performance for each initialization method, and Table 4.14 reports the efficiency metrics.

No initialization strategy produced a statistically significant difference from the Zero / None baseline after Bonferroni correction, so we are only showing our full averaged results with standard deviations. Zero / Embedding leads on mean nDCG' (0.483 ± 0.004) and MAP' (0.145 ± 0.003), with a gap of 0.016 nDCG' over Zero / None. All conditions with a learnable component trend

Table 4.13: Answer retrieval performance by virtual node initialization strategy (mean ± std over 5 seeds). "Typed Proj." denotes a separate learned linear projection per virtual node type. Mean-pooled / Projection reports 4 seeds (one run produced no evaluation output).

Init Value	Learned	nDCG'	MAP'	P'@10	Bpref
Zero[†]	None	0.467 ± 0.003	0.134 ± 0.005	0.199 ± 0.005	0.217 ± 0.005
Zero	Embedding	$\mathbf{0.483 \pm 0.004}$	$\mathbf{0.145 \pm 0.003}$	$\mathbf{0.206 \pm 0.008}$	0.215 ± 0.006
Random	Embedding	0.476 ± 0.007	0.139 ± 0.006	0.200 ± 0.008	0.217 ± 0.005
Mean-pooled	None	0.470 ± 0.008	0.134 ± 0.005	0.201 ± 0.010	0.215 ± 0.006
Mean-pooled	Projection	0.477 ± 0.010	0.138 ± 0.006	0.204 ± 0.002	$\mathbf{0.219 \pm 0.002}$
Mean-pooled	Typed Proj.	0.477 ± 0.005	0.137 ± 0.003	0.204 ± 0.007	0.213 ± 0.007

[†]Baseline

Table 4.14: Efficiency metrics by virtual node initialization strategy.

Init Value	Learned	Time/Epoch (s)	Params
Zero	None	455.184 ± 3.883	7.917M
Zero	Embedding	455.343 ± 3.924	7.917M
Random	Embedding	456.293 ± 1.713	7.917M
Mean-pooled	None	460.883 ± 2.312	7.917M
Mean-pooled	Projection	459.146 ± 1.827	7.921M
Mean-pooled	Typed Proj.	468.075 ± 1.503	7.929M

higher than their corresponding non-learned counterparts, but the differences are small relative to cross-condition variance.

Mean-pooled initialization does not reliably outperform zero initialization, suggesting that providing a structural prior via aggregation offers limited benefit. A possible explanation is that RGCN message-passing layers are sufficiently expressive to propagate subgraph structure into virtual node representations during training, reducing sensitivity to initial values. The Zero / Embedding condition adds no computational overhead and performs well, making it a practical default for the full-scale experiments.

The next experiments shift focus from graph design to retrieval, examining how different retrieval strategies affect the balance of effectiveness and efficiency.

4.4 Retrieval Experiments

The following two experiments examine how different strategies for querying the multi-vector index trade off effectiveness against throughput, and whether an auxiliary training objective can recover quality losses incurred by retrieval modes that utilize the document-level representation.

4.4.1 Retrieval Strategy

This experiment compares retrieval strategies in effectiveness and efficiency. The MathAMR+ model produces a hierarchy of virtual node embeddings at document, sentence, and formula granularities, and the choice of retrieval strategy determines how these embeddings are used at query time, with direct consequences for both retrieval quality and throughput. The three strategies (as defined in Section 3.5.2) are:

1. **ColBERT-style MaxSim.** All query virtual node embeddings independently search the index; in the approximate (HNSW) variant each query virtual node explores up to 128 candidate neighbors during graph traversal, and the union of candidates across all query virtual nodes is scored by MaxSim to return the final top-k results.

2. **Doc-VN + MaxSim reranking.** The document virtual node retrieves a top-k candidate pool from the single-vector Doc-VN index, and those candidates are then reranked by exact MaxSim over the full virtual node sets. We vary candidate size $k \in \{5000, 1000\}$.

3. **Single-vector.** The document virtual node retrieves $k = 1000$ documents with no intermediate expansion or reranking.

Each strategy is evaluated under exact vs. approximate HNSW search.

Control Variables

The model is trained with formula representation Both (OPT+SLT), Zero/Embedding virtual node initialization, and no auxiliary edges between virtual nodes. All remaining training hyperparameters, other than the independent variable (retrieval mode), are as in Table 4.5. Models are trained on 50,000 positive pairs (45,000 train / 5,000 validation) over 5 random seeds. The retrieval index contains 33,771 posts (the judged subset of the ARQMath-3 corpus).

Hypotheses

Exact ColBERT-style MaxSim is expected to achieve the highest retrieval quality. Approximate HNSW ColBERT-style MaxSim should approach the exact upper bound at higher throughput, since HNSW approximation error is typically small. The two exact-vs.-approximate pairs are expected to show small differences, quantifying the approximation penalty. Approximate single-vector is expected to be the weakest strategy because it collapses the multi-granularity hierarchy into a single representation.

Results and Analysis

Table 4.15 shows answer retrieval effectiveness and query throughput on the seed with highest nDCG'. *ColBERT MaxSim* achieves the highest nDCG' (0.454), MAP' (0.136), and P'@10 (0.209). Single-vector retrieval is at the bottom on all effectiveness metrics (0.330 nDCG', 0.083 MAP', 0.142 P'@10, 0.180 Bpref). The averaged results shown in Table 4.16, which include both exact and HNSW variants, show the same pattern across seeds.

Table 4.15: Answer retrieval effectiveness and query throughput by retrieval strategy using approximate (HNSW) search (best seed, seed 42). QPS = queries per second. See Table 4.16 for exact search comparisons and significance tests.

Retrieval Strategy	**nDCG'**	**MAP'**	**P'@10**	**Bpref**	**QPS**
ColBERT MaxSim† (ef = 128)	**0.454**	**0.136**	**0.209**	0.221	90.1
Doc-VN + MaxSim ($k = 5000$)	0.451	0.133	0.203	0.220	83.3*
Doc-VN + MaxSim ($k = 1000$)	0.352*	0.103*	0.191	**0.230**	115.2*
Single-Vector	0.330*	0.083*	0.142*	0.180*	**132.0***

†Baseline, *$p < 0.05$ (Bonferroni corrected)

Table 4.16: Answer retrieval effectiveness and query throughput by retrieval strategy (mean ± std over 5 seeds). QPS = queries per second.

Retrieval Strategy	**nDCG'**	**MAP'**	**P'@10**	**Bpref**	**QPS**
ColBERT MaxSim$^{\bullet}$	$\mathbf{0.452 \pm 0.013}$	$\mathbf{0.132 \pm 0.007}$	0.194 ± 0.012	0.221 ± 0.005	62.9 ± 4.1
ColBERT MaxSim$^{\circ}$	0.440 ± 0.014	0.127 ± 0.007	$\mathbf{0.195 \pm 0.013}$	0.218 ± 0.005	87.2 ± 6.7
Doc-VN + MaxSim$^{\bullet}$ ($k = 5000$)	0.428 ± 0.024	0.124 ± 0.009	0.191 ± 0.017	0.218 ± 0.004	85.7 ± 2.2
Doc-VN + MaxSim$^{\circ}$ ($k = 5000$)	0.428 ± 0.024	0.124 ± 0.009	0.191 ± 0.017	0.218 ± 0.004	83.4 ± 3.4
Doc-VN + MaxSim$^{\bullet}$ ($k = 1000$)	0.318 ± 0.025	0.090 ± 0.009	0.181 ± 0.012	$\mathbf{0.222 \pm 0.009}$	112.0 ± 12.2
Doc-VN + MaxSim$^{\circ}$ ($k = 1000$)	0.318 ± 0.025	0.090 ± 0.009	0.181 ± 0.012	$\mathbf{0.222 \pm 0.009}$	114.9 ± 7.4
Single-Vector$^{\bullet}$	0.299 ± 0.022	0.075 ± 0.008	0.134 ± 0.013	0.165 ± 0.012	131.5 ± 1.9
Single-Vector$^{\circ}$	0.299 ± 0.022	0.075 ± 0.008	0.134 ± 0.013	0.165 ± 0.012	$\mathbf{131.7 \pm 1.4}$

$\bullet$ = Exact search, $\circ$ = HNSW search

The near identical performance of *Doc-VN + MaxSim* at $k = 5{,}000$ compared to *ColBERT MaxSim* can be attributed to differences in candidate pool size. A candidate set of 5,000 posts corresponds to approximately $\sim 15\%$ of the corpus. In contrast, the *ColBERT MaxSim* HNSW configuration uses an "efficiency factor" (ef) value of 128, meaning each query vector retrieves only 128 candidates during the first stage. Although Qdrant does not expose intermediate candidate pool sizes in its API responses, a query with 10 virtual nodes yields a theoretical upper bound of

$10 \times 128 = 1{,}280$ unique candidates posts. The larger candidate pool size also explains the slower query throughput of *Doc-VN + MaxSim* at $k = 5{,}000$ compared to *ColBERT MaxSim* (83.3 vs. 90.1 queries per second).

The averaged results reveal that within the *Doc-VN + MaxSim* and *Single-Vector* conditions, the exact and HNSW variants show identical effectiveness (Table 4.16), indicating that HNSW approximation error is negligible at this index scale (33,771 posts). At the full ARQMath scale of approximately 1M posts, HNSW is expected to yield larger throughput gains.

Reducing k from 5,000 to 1,000 drops nDCG' by 0.099 and P'@10 by 0.12, and raises throughput to 115.2 QPS (+31.9 over $k = 5{,}000$). The sensitivity of nDCG' to k reflects its integration of relevance signal over the full ranked list, making it sensitive to first-stage recall, whereas P'@10 focuses on top-10 results which remains quite high due to the effectiveness of MaxSim reranking.

Single-Vector retrieval achieves the highest throughput at 132.0 QPS (+41.9 over *ColBERT MaxSim*), but performs substantially below all multi-vector configurations: nDCG' of 0.330, MAP' of 0.083, P'@10 of 0.142, and Bpref of 0.180. The drops in all metrics are statistically significant after Bonferroni correction. The overall result confirms that collapsing the multi-granularity hierarchy to a single document-level embedding discards substantial retrieval signal.

These results reveal that while *Doc-VN + MaxSim* at small k and *Single-Vector* retrieval directly improve throughput, they fall substantially below the *ColBERT MaxSim* upper bound. The following experiment investigates whether an auxiliary training objective targeting v_{doc} can narrow this gap for efficiency-focused deployment strategies.

4.4.2 Auxiliary Document-Level Loss

The MaxSim objective provides only indirect supervision for v_{doc}, since it optimizes retrieval through the full set of virtual nodes rather than enforcing that any single vector be effective for retrieval. Thus, v_{doc} may be suboptimal for single-vector or first-stage retrieval use cases. The auxiliary loss is designed to close this gap without degrading MaxSim performance.

We introduce an auxiliary loss that applies an InfoNCE objective directly to the document-level virtual node embeddings, encouraging v_{doc} to function as a strong standalone representation. The auxiliary loss is defined as:

$$\mathcal{L}_{\text{doc}} = -\frac{1}{B}\sum_{i=1}^{B} \log \frac{\exp(R_{ii}/\tau)}{\sum_{j=1}^{B} \exp(R_{ij}/\tau)}, \tag{4.1}$$

where $R_{ij} = (v_{\text{doc}}^{(q_i)})^\top v_{\text{doc}}^{(d_j)}$ denotes the similarity between query and document single-vector embeddings. We combine this with the MaxSim objective $\mathcal{L}_{\text{MaxSim}}$ from equation 3.8 using a linear

combination:

$$\mathcal{L} = \beta\mathcal{L}_{\text{MaxSim}} + (1-\beta)\mathcal{L}_{\text{doc}}, \quad \beta \in [0,1]. \tag{4.2}$$

We evaluate the baseline ($\beta = 1.0$) and four auxiliary settings $\beta \in \{0.7, 0.5, 0.3, 0\}$. All trained models are evaluated under all retrieval strategies, with particular focus on *Doc-VN + MaxSim* and *Single-Vector* retrieval modes, where performance is directly determined by the quality of v_{doc}.

Control Variables

All model and training hyperparameters are fixed at the Table 4.5 values, with formula representation Both (OPT+SLT), Zero/Embedding virtual node initialization, no auxiliary edges, and 50,000 training pairs over 5 random seeds. All retrieval strategies use approximate (HNSW, ef = 128) search. The sole varied factor is the loss weight β, which controls the relative contribution of the MaxSim objective and the auxiliary document-level InfoNCE objective.

Hypotheses

The goal is to improve the efficiency-focused configurations identified above, not to improve peak retrieval accuracy. *ColBERT MaxSim* is expected to remain the most effective strategy regardless of training objective. Training with $\mathcal{L}_{\text{doc}}$ should improve both *Single-Vector* and *Doc-VN + MaxSim* by explicitly optimizing v_{doc} as a dense retrieval representation. The benefit should be largest for single-vector modes, which have no MaxSim reranking to compensate for a weak document embedding. For *Doc-VN + MaxSim*, the improvement may be more limited if first-stage recall is already high. *ColBERT MaxSim* is not expected to benefit and may degrade at low β values if document-level optimization competes with sentence- and formula-level virtual nodes for representational capacity, with the risk increasing as $\beta \to 0$ and $\mathcal{L}_{\text{MaxSim}}$ is removed entirely.

Results and Analysis

Table 4.17 reports the effect of the auxiliary document-level loss across retrieval modes on nDCG', using the best-trained seed per β value. For the *Doc-VN + MaxSim* method, only $k = 1{,}000$ is shown.

The auxiliary document-level loss provides a clear benefit for retrieval modes that utilize v_{doc}. Without any auxiliary loss, *Doc-VN + MaxSim* at $k = 1{,}000$ achieves nDCG' of 0.352, 0.102 below the *ColBERT MaxSim* baseline of 0.454. With $\beta = 0.3$ (70% of training weight on $\mathcal{L}_{\text{doc}}$), *Doc-VN + MaxSim* reaches nDCG' of 0.432, recovering 78% of that deficit. Single-vector nDCG' improves from 0.330 to 0.411 (+0.081). Both gains are statistically significant after Bonferroni correction.

ColBERT MaxSim is largely unaffected by the auxiliary loss when $\beta \geq 0.3$, with nDCG' staying nearly constant across those values, partially refuting the representational competition hypothesis.

Table 4.17: Effect of auxiliary document-level loss on nDCG' and P'@10 across retreival modes (best seed per β). $\beta = 1$ is the no-auxiliary-loss baseline; decreasing β shifts training weight from $\mathcal{L}_{\text{MaxSim}}$ toward $\mathcal{L}_{\text{doc}}$. All retrieval strategies utilize approximate (HNSW) search. Only $k = 1{,}000$ is shown for the *Doc-VN + MaxSim* method.

	ColBERT MaxSim		**Doc-VN + MaxSim**		**Single-Vector**	
Loss weight	**nDCG'**	**P'@10**	**nDCG'**	**P'@10**	**nDCG'**	**P'@10**
$\beta = 1^{\dagger}$	**0.454**	0.209	0.352	0.191	0.330	0.142
$\beta = 0.7$	0.454	**0.210**	0.412*	**0.210**	0.396*	0.165
$\beta = 0.5$	0.452	0.200	0.421*	0.194	0.405*	0.167
$\beta = 0.3$	0.454	**0.210**	0.432*	0.204	0.411*	0.165
$\beta = 0$	0.365*	0.200	**0.437***	0.196	**0.425***	**0.176***

†Baseline, $*p < 0.05$ (Bonferroni corrected)

However, at $\beta = 0$ the MaxSim objective is removed entirely and *ColBERT MaxSim* nDCG' drops significantly to 0.365 (-0.089), confirming that $\mathcal{L}_{\text{MaxSim}}$ is essential for maintaining multi-vector retrieval quality. The representational competition hypothesis therefore holds, but only at the extreme of eliminating the MaxSim loss altogether.

When looking at P'@10, incorporating auxiliary loss seems to have a negligible impact on the precision of the top ten judged results for *ColBERT MaxSim* and *Doc-VN + MaxSim* retrieval modes. With *Single-Vector* retrieval, decreasing β from 1 to 0.5 increases P'@10 from 0.142 to 0.167, which is noticeable but is not shown to be statistically significant. Single-Vector is not able to reach the precision of multi-vector MaxSim, however, as it lags behind both *ColBERT MaxSim* and *Doc-VN + MaxSim* by around 0.045 on P'@10. This suggests that the auxiliary loss primarily improves first-stage recall, and that top-10 precision via MaxSim is largely insensitive to candidate pool size once a basic threshold of first-stage recall is met.

Overall, the auxiliary document-level loss provides a strong practical benefit for deployment-oriented retrieval settings. While $\beta = 0$ achieves the highest nDCG' for *Doc-VN + MaxSim* (0.437) and *Single-Vector* (0.425), it does so at the cost of a significant drop in *ColBERT MaxSim* accuracy. $\beta = 0.3$ offers the best practical tradeoff: it strongly improves document-level representations while leaving *ColBERT MaxSim* unaffected, making it the preferred choice when top-end retrieval quality must be preserved.

The experiments point to a clear configuration: both formula representations, Zero/Embedding virtual node initialization, the default no-auxiliary-edge hierarchy, *ColBERT MaxSim* retrieval, and $\beta = 0.3$ auxiliary loss. With these choices made, the system is trained at full scale and evaluated against prior ARQMath-3 systems.

4.5 Benchmark: ARQMath-3

Based on the learnings from the preceding experiments, we selected the following configuration for our benchmarking run: dual formula representation (OPT + SLT), Zero/Embedding virtual node initialization, the default no-auxiliary-edge hierarchy, and $\beta = 0.3$ auxiliary document loss. This configuration was trained and evaluated on the complete ARQMath-3 corpus, with retrieval performed via ColBERT-style MaxSim. The goal is to assess how MathAMR+ compares to prior ARQMath-3 systems on Tasks 1 and 2, and in particular to the original MathAMR system on which it builds.

Due to time constraints and job scheduling delays on RIT's Research Computing cluster, the final benchmarking runs were performed on the same workstation used for the experiments (Table 4.4). As a result, the benchmarking model was trained at the same scale as the previous experiments. The hyperparameters for the benchmarking run are listed in Table 4.18.

Table 4.18: Hyperparameters used for the benchmarking experiment.

Category	**Hyperparameter**	**Value**
Model	Hidden dimension	64
	Projection dimension	32
	Number of layers	4
	Number of bases	256
	VN init strategy	Zero learned
Training	Batch size	16
	Number of epochs	7
	Learning rate	5×10^{-4}
	Minimum learning rate	1×10^{-5}
	Weight decay	1×10^{-3}
	Initial temperature	0.05 (learnable)
	Gradient clipping norm	1.0
	Early stopping patience	3 epochs
Augmentation	Node mask probability	0.1
	Edge dropout probability	0.1
Loss	Score normalization	Enabled
	Auxiliary doc-level loss β	0.3
Data	Train/validation split	90%/10%
	Formula representations	Both (SLT + OPT)
	Auxiliary virtual node edges	None
Retrieval	Retrieval mode	ColBERT-style MaxSim

The comparison systems for Task 1 are:

- **Approach0 / Struct.+ColBERT (Coco-MAE)** [68]: Combines structural math search (Approach0) with ColBERT-style dense re-ranking using Coco-MAE, a contrastively trained math-aware encoder. The top-performing system at ARQMath-3.
- **Kassaie et al. / RRF+LLM** [23]: Uses LLMs to reformulate queries, and reciprocal rank fusion (RRF) to combine several component approaches. Included as a recent system that exploits modern LLM capabilities.
- **MSM / Ensemble RRF** [14]: A multi-system ensemble using RRF to combine models trained on LaTeX strings and operator trees. Included as a competitive ensemble baseline.
- **MIRMU / MiniLM+RoBERTa** [50]: An ensemble of transformer encoder models (MiniLM and RoBERTa) fine-tuned for math-aware retrieval. Included as a representative transformer-based baseline.
- **Satpute et al. / GPT-4+DPR** [52]: Uses GPT-4 to reformulate queries and Dense Passage Retrieval for document encoding. Included as an example of LLM-augmented retrieval applied to ARQMath.
- **DPRL / QQ-QA-AMR** [36]: Performs search via Sentence-BERT embeddings on linearized MathAMR trees, serving as the primary AMR-based baseline.

The comparison systems for Task 2 are:

- **L2L / All (Unweighted)** [60]: An ensemble system that extracts shortest paths between leaf nodes in OPTs and combines these structural features with LaTeX similarity and Presentation MathML symbol matching. Included as the current state-of-the-art on Task 2.
- **Amador et al. / In+Out (published) and Inner RRF (unofficial)** [4]: A dense model that learns node-level R-GCN embeddings via self-supervised contrastive learning on SLTs and OPTs. It uses ColBERT-style late interaction over a node-level index for retrieval. Included to demonstrate the effectiveness of using GNNs and multi-vector representations. The *Inner RRF* run also reported in the comparison table is an updated unofficial run by the authors and was not published in their paper.
- **Approach0 / Struct.+ColBERT** [68]: The same structural+ColBERT system as in Task 1, applied to formula retrieval. Included as the strongest ARQMath-3 competition result on Task 2.

- **DPRL / TanCFT+MathAMR** [36]: Combines Tangent-CFT formula embeddings with MathAMR graph representations via score fusion. Included as the best run from the MathAMR group.
- **MathDowsers / latex_L8_a040** [20]: A formula retrieval system based on LaTeX symbol tree matching and tuple scoring. Included as a competitive symbolic baseline.
- **DPRL / MathAMR** [36]: The original MathAMR system, which uses Sentence-BERT embeddings on linearized MathAMR trees. Included for direct comparison with MathAMR+.

Tables 4.19 and 4.21 report the results, with efficiency summaries in Tables 4.20 and 4.22 respectively. Appendix C presents example top-3 retrieval results for topics A.301 and B.301, comparing MathAMR+ and the original MathAMR system side by side.

4.5.1 Task 1: Answer Retrieval

Table 4.19 compares MathAMR+ against prior ARQMath-3 systems on Task 1. MathAMR+ achieves nDCG' of 0.334, substantially outperforming *DPRL / QQ-QA-AMR* (nDCG' 0.185) but trailing the other systems. The most important comparison is with QQ-QA-AMR, which also builds on MathAMR representations but linearizes the graph for a Sentence-BERT encoder with 1024-dimensional embeddings; the improvement of 0.149 nDCG' demonstrates that applying RGCN message passing directly to the graph topology with ColBERT-style MaxSim interaction is a more effective strategy. The gap to the remaining systems should be understood in light of a substantial scale difference, as those approaches leverage large language models or large transformer ensembles, while MathAMR+ was trained under the hardware constraints described above. Qualitatively, the difference is visible in Appendix C. For topic A.301, a question about a matrix norm inequality proof, MathAMR+ places a complete proof of the target inequality at rank 2 (relevance 3) with two closely related norm equivalence answers in ranks 1 and 3, whereas the prior MathAMR run retrieves an inner-product inequality at rank 1 (relevance 1) followed by two entirely unrelated results (relevance 0), apparently matching on surface inequality notation without capturing the matrix norm context.

4.5.2 Task 2: Formula Retrieval

Table 4.21 reports Task 2 results. MathAMR+ achieves nDCG' of 0.507, outperforming the official DPRL / MathAMR run (0.316) on all metrics, and on P'@10 (0.588) and Bpref (0.503) also outperforming the unofficial post hoc MathAMR run. On nDCG' and MAP', MathAMR+ falls short of the unofficial run, which was tuned to use only the sentence containing each formula. In the retrieval results shown in Appendix C, topic B.301 shows a particularly clear qualitative contrast

Table 4.19: ARQMath-3 Answer Retrieval (Task 1) benchmarking results.

System	**nDCG'**	**MAP'**	**P'@10**	**Bpref**
Approach0 / Struct.+ColBERT (Coco-MAE) [68]	**0.546**	**0.237**	0.360	0.221
Kassaie et al. / RRF+LLM [23]	0.522	0.195	0.277	–
MSM / Ensemble RRF [14]	0.504	0.157	0.241	0.138
MIRMU / MiniLM+RoBERTa [50]	0.498	0.184	0.267	0.169
Satpute et al. / GPT-4+DPR [52]	0.486	0.219	**0.374**	**0.225**
DPRL / QQ-QA-AMR [36]	0.185	0.040	0.091	–
Ours / MathAMR+	0.334	0.108	0.236	0.163

Table 4.20: Task 1 system efficiency metrics.

Metric	**Value**
Training time	69.2 min/epoch
GPU peak memory	9.75 GB
Model parameters	5,653,808
Index size	2.79 GB
Number of vectors	21,947,328
Query throughput	2.66 queries/sec

between our results and the official submitted MathAMR run. The query formula $\|A\|_2 = \sqrt{\rho(A^T A)}$ is the central object of the surrounding post, and MathAMR+ returns three direct structural matches at ranks 1–3 (all relevance 3), while the baseline retrieves three completely unrelated formulas from complex analysis and trigonometry (all relevance 0), yielding zero across all metrics for this topic.

A gap to the leading systems is visible, but these results are promising given that MathAMR+ is not trained explicitly for formula retrieval. Formula retrieval capability emerges as a by-product of document-level MaxSim training, with formula virtual node embeddings serving as the index. Future work could directly optimize formula embeddings by including formula-level contrastive pairs during training, which may close this gap considerably.

4.6 Summary

This chapter has examined MathAMR+ across five experiments and two benchmark evaluations, progressing from questions of data representation and embedding to questions of retrieval strategy and system-level performance. We now summarize our key findings.

Table 4.21: ARQMath-3 Formula Retrieval (Task 2) benchmarking results.

System	nDCG'	MAP'	P'@10	Bpref
L2L / All (Unweighted) [60]	**0.849**	**0.620**	0.680	–
Amador et al. / Inner RRF† [4]	0.773	0.554	0.632	–
Approach0 / Struct.+ColBERT [68]	0.720	0.568	**0.688**	**0.560**
Amador et al. / In+Out [4]	0.701	0.505	0.597	–
DPRL / TanCFT+MathAMR [36]	0.681	0.471	0.617	–
MathDowsers / latex_L8_a040 [20]	0.640	0.451	0.549	0.443
DPRL / MathAMR†[1] [36]	0.579	0.367	0.549	–
DPRL / MathAMR[2] [35]	0.316	0.160	0.253	—
Ours / MathAMR+	0.507	0.333	0.588	0.503

† *Unofficial run by the authors*

Table 4.22: Task 2 system efficiency metrics.

Metric	Value
Number of indexed formulas	21,515,119
Query throughput	2.19 queries/sec

Formula representations. Removing formula representations entirely causes a drop of approximately 0.093 nDCG' on the best seed (Table 4.6), far larger than any other difference observed across experiments. This establishes that structured formula content is an essential retrieval signal in this domain, and that text-only AMR graphs are insufficient. All three formula-enabled conditions are significantly superior to None on nDCG' and MAP' after Bonferroni correction. Among formula-enabled conditions, using both OPT and SLT representations achieves the highest nDCG' on both the best seed (0.497) and the averaged results (0.494 ± 0.002), slightly ahead of OPT-only (0.496 and 0.486 ± 0.007) and SLT-only (0.495 and 0.488 ± 0.006). However, OPT-only and SLT-only achieve near identical performance, suggesting that the RGCN message-passing layers are able to extract complementary signals from the two representations, but similar levels of information with either one alone.

Virtual node structure. The full hierarchy of formula, sentence, and document virtual nodes achieves the most consistent retrieval performance overall. The no-VN condition performs worst on nDCG', MAP', and P'@10, with the global VN + MaxSim condition improving over no-VNs significantly on nDCG' and MAP' ($p < 0.05$ after Bonferroni correction). Global VN single-vector produces the highest nDCG' of all conditions, but lower MAP' and P'@10, suggesting the single-vector objective produces a well-calibrated global ranking at the cost of breadth. The full hierar-

chy provides the best balance, with purpose-built aggregation nodes at each granularity enabling MaxSim to selectively match at formula, sentence, or document level.

Virtual node connectivity. Adding auxiliary edges between same-level virtual nodes consistently degrades retrieval at the scale tested. On the best seed (Table 4.10), None outperforms all three connected configurations on every metric (0.431 vs. 0.420–0.423 nDCG'). The averaged results (Table 4.11) show the same pattern. None of the degradations reach statistical significance after Bonferroni correction, however, reflecting consistent but modest effects. The representation collapse visible in Figure 4.1 provides qualitative evidence for the mechanism, where auxiliary edges cause virtual node representations to converge, reducing the per-level specificity that multi-vector MaxSim depends on.

Virtual node initialization. No initialization strategy produces a statistically significant difference from the *Zero / None* baseline. Table 4.13 shows that *Zero / Embedding* leads on averaged nDCG' (0.483 ± 0.004) with the closest p-values to significance ($p = 0.064$ nDCG', $p = 0.067$ MAP'). The implication is that RGCN message passing broadly compensates for different starting states. Given these experiments were conducted at reduced scale, it remains an open question whether initialization sensitivity would emerge at full scale.

Retrieval strategy and auxiliary loss. The retrieval experiments reveal a clear hierarchy. *ColBERT MaxSim* achieves the best results for nDCG' (0.454), MAP' (0.136), and P'@10 (0.209), as shown in Table 4.15. *Doc-VN + MaxSim* at k =1,000 performs worse with nDCG' of 0.352, and *Single-Vector* retrieval substantially underperforms all multi-vector strategies (nDCG' 0.330) with statistical significance on nDCG', MAP', and Bpref. The auxiliary document-level loss largely closes the gap for efficiency-oriented modes: on the best seed, two-stage retrieval at k = 1,000 improves from nDCG' 0.352 to 0.432 at $\beta = 0.3$ (+0.079, statistically significant), and single-vector improves from 0.330 to 0.411 (+0.081, statistically significant) (Table 4.17). ColBERT MaxSim is not affected for $\beta \geq 0.3$, but drops significantly to nDCG' 0.365 at $\beta = 0$ when $\mathcal{L}_{\text{MaxSim}}$ is removed entirely. This makes $\beta = 0.3$ the preferred setting, improving efficiency-oriented modes without sacrificing peak retrieval quality.

Benchmark. On Task 1, MathAMR+ outperforms *DPRL / QQ-QA-AMR* by 0.149 nDCG', confirming that graph-based relational learning with a VN hierarchy and ColBERT-style MaxSim is more effective than linearizing MathAMR for a Sentence-BERT encoder. The gap to systems that employ LLMs or large transformer ensembles reflects the hardware-constrained model scale. On Task 2, MathAMR+ reaches nDCG' of 0.507 and outperforms the official MathAMR run on all

metrics, a result made more notable by the fact that formula retrieval is not a training objective but an emergent capability of the formula virtual node hierarchy.

Chapter 5

Conclusion

This chapter summarizes the contributions of this thesis, discusses the limitations of the proposed approach, and identifies directions for future work.

5.1 Summary of Contributions

Existing math-aware retrieval systems either treat text and formulas as separate modalities, or adopt a unified representation but linearize it for a Transformer encoder, sacrificing the structural information that makes the representation useful. This thesis addresses both limitations with MathAMR+, a framework that jointly encodes AMR predicate-argument graphs, OPT operator trees, and SLT layout trees in a single heterogeneous graph and applies a Relational Graph Convolutional Network directly to the graph topology. The representation is further augmented with a hierarchy of virtual nodes at the formula, sentence, and document levels. After message passing, these virtual node embeddings form a multi-vector document representation that supports ColBERT-style MaxSim retrieval.

Five experiments investigated the impact of specific design decisions. Formula structure is essential, as removing formulas drops nDCG' by 0.093 on the best seed, substantially larger than any gap among formula-enabled conditions, establishing that textual AMR alone is insufficient for math-aware retrieval. Combining OPT and SLT yields the best performance on all metrics with consistent but modest gains of 0.005–0.013 across metrics over either alone, though the near-parity of OPT-only and SLT-only suggests either representation captures sufficient discriminative information at the scales tested, and the practical value of combining both depends on whether the additional computational cost is acceptable.

The virtual node structure experiment confirms that the full formula–sentence–document hierarchy achieves the highest MAP', P'@10, and Bpref. Global VN with single-vector retrieval produces the highest nDCG' overall, but at the cost of MAP' and P'@10, as collapsing all docu-

ment structure into a single embedding limits retrieval breadth. The no-VN condition performs worst on nDCG' and MAP', with the gap to the full hierarchy statistically significant. The full hierarchy provides the strongest and most consistent performance overall.

Regarding virtual node connectivity, adding auxiliary edges between sentence and formula virtual nodes consistently degrades retrieval at the scale tested. Heatmap analysis reveals that these edges cause virtual node representations to converge and lose the per-VN specificity that multi-vector matching requires, indicating that the hierarchy is most effective when virtual nodes remain structurally distinct.

Virtual node initialization has no statistically significant effect on retrieval performance, as no pairwise difference passes the Bonferroni-corrected threshold. This suggests that the RGCN layers propagate subgraph information into virtual nodes regardless of their initial state, and more complex initialization schemes are unlikely to justify their added complexity.

Multi-vector MaxSim outperforms two-stage and single-vector retrieval without auxiliary training, and HNSW approximation introduces negligible quality loss. Adding an auxiliary contrastive loss on the document virtual node largely closes the gap between efficient and accurate retrieval, with two-stage retrieval at $k = 1{,}000$ improving from nDCG' 0.352 to 0.432 and single-vector retrieval improving from 0.330 to 0.411 at $\beta = 0.3$ without degrading ColBERT MaxSim. While $\beta = 0$ further improves these efficiency-oriented modes that rely on the document virtual node, it causes a significant drop in ColBERT MaxSim nDCG' ($0.454 \rightarrow 0.365$), confirming that $\mathcal{L}_{\text{MaxSim}}$ must be retained. This makes $\beta = 0.3$ the preferred setting, and suggests that a lightweight auxiliary objective on aggregate embeddings may transfer to other multi-vector frameworks facing a similar trade-off between recall and inference cost.

Benchmarking on the full ARQMath-3 corpus confirms that MathAMR+ achieves meaningful retrieval performance at scale. On Task 1 (answer retrieval), MathAMR+ attains nDCG' of 0.334, substantially outperforming the prior AMR-based system DPRL / QQ-QA-AMR (nDCG' 0.185), which encodes linearized MathAMR graphs using Sentence-BERT with 1024-dimensional embeddings. This improvement demonstrates that graph-based relational learning with a virtual node hierarchy and ColBERT-style multi-vector MaxSim matching is a more effective strategy for exploiting MathAMR structure than linearization for a Transformer encoder. The gap to the best-performing systems reflects the hardware-constrained model scale used for this run, as those systems rely on LLMs or large transformer ensembles substantially larger than MathAMR+. On Task 2 (formula retrieval), MathAMR+ achieves nDCG' of 0.507, surpassing the official MathAMR run (nDCG' 0.316). While it falls short of the unofficial MathAMR post hoc run on nDCG' and MAP', it is notable that the model was not trained for formula retrieval at all. Formula retrieval capability emerges as a by-product of post-level MaxSim training through the formula virtual node hierarchy, and directly optimizing formula embeddings for formula search is a promising direction

for future work.

5.2 Limitations

Several limitations constrain the scope of conclusions that can be drawn from this work.

The quality of the resulting AMRs are ultimately bounded by the pretrained parser's coverage of mathematical prose. Errors or noise in predicate-argument structure propagate directly into the graph representation, degrading the quality of node embeddings and message passing without any mechanism for downstream correction.

Graph construction is computationally demanding. AMR parsing requires GPU-based neural inference, and processing the full ARQMath collection of approximately 2.47 million posts took around 22 days using 4 A100 GPUs. This limits the practicality of extending MathAMR+ to substantially larger corpora without significant compute cost.

Node features are initialized from a learned lookup table over observed node types with no transfer from pretrained language or mathematical models. The model therefore cannot leverage the semantic priors encoded in large pretrained representations, and nodes whose types were unseen at training time are mapped to a shared unknown embedding. This constraint is particularly significant for mathematical symbols and operator names, where pretrained models trained on mathematical text could provide richer initial features.

5.3 Future Work

One direct extension of this work is the integration of pretrained embeddings as node feature initializers. Replacing the learned lookup table with contextual embeddings from a model such as MathBERT [44] or a general-purpose encoder fine-tuned on mathematical text could improve the quality of initial node representations. Exploring alternative node normalization and vocabulary construction strategies could also be valuable.

Contrastive training with in-batch negatives is a simple and effective baseline, but the quality of negatives is limited by batch size and random sampling. Hard negative mining strategies, such as BM25-based retrieval of near-miss candidates or same-thread adversarial negatives drawn from highly similar questions, could provide more informative training signal and improve the model's ability to discriminate between mathematically similar but semantically distinct documents.

The virtual node hierarchy introduced in this work is a general-purpose mechanism for multi-granularity graph representation. Future work could explore attention-based aggregation within virtual nodes, replacing the uniform mean-pooling implicit in RGCN aggregation with learned weights over constituent node embeddings. Graph attention networks or graph transformer ar-

chitectures could similarly learn to focus on salient structural regions during message passing, potentially improving retrieval quality.

The current framework is designed and evaluated exclusively on the ARQMath benchmark. Extending MathAMR+ to other domains, such as chemistry, where formulas and structured notation play similar roles, would test the generality of the approach. The graph construction pipeline is domain-agnostic in principle, provided that appropriate structured representations of domain-specific notation are available.

The benchmarking runs in this thesis were conducted at a small model scale (64-dimensional hidden representations, 4 RGCN layers) due to hardware constraints. Training at larger hidden dimensions, more layers, and larger batch sizes would likely improve retrieval effectiveness. It would also be valuable to compare the scaling behavior of MathAMR+ against prior approaches to understand whether the architectural advantages observed at small scale hold or widen as model capacity increases.

For Task 2, the current model uses formula virtual node embeddings for retrieval as an emergent by-product of post-level contrastive training. A direct extension would be to include formula-level contrastive pairs during training, supervising formula embeddings with formula-to-formula relevance signal.

Finally, the MathAMR+ representation captures rich semantic and structural information about mathematical documents that may be useful beyond retrieval. Applying these graph representations to downstream tasks such as formula recommendation or retrieval-augmented generation over mathematical content represents a natural avenue for further work.

Bibliography

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

[2] Akiko Aizawa, Michael Kohlhase, and Iadh Ounis. NTCIR-10 Math Pilot Task Overview. In *Proceedings of the 10th NTCIR Workshop Meeting on Evaluation of Information Access Technologies*, Tokyo, Japan, 2013.

[3] Akiko Aizawa, Michael Kohlhase, and Iadh Ounis. NTCIR-11 Math-2 Task Overview. In *Proceedings of the 11th NTCIR Workshop Meeting on Evaluation of Information Access Technologies*, Tokyo, Japan, 2014.

[4] Bryan Amador and Richard Zanibbi. Math Formula Graph Retrieval Using Contrastive Learning Over Visual and Semantic Embeddings. In *Proceedings of the 2025 International ACM SIGIR Conference on Innovative Concepts and Theories in Information Retrieval (ICTIR)*, pages 230–237, Padua Italy, July 2025. ACM.

[5] Dan Busbridge, Dane Sherburn, Pietro Cavallo, and Nils Y. Hammerla. Relational Graph Attention Networks, April 2019. arXiv:1904.05811 [cs].

[6] Alessandro Caruso, Jacopo Venturin, Lorenzo Giambagli, Edoardo Rolando, Zakariya El-Machachi, Frank Noé, and Cecilia Clementi. Extending the range of graph neural networks with global encodings. *Nature Communications*, 17(1):1855, February 2026.

[7] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A Simple Framework for Contrastive Learning of Visual Representations. In *Proceedings of the 37th International Conference on Machine Learning (ICML 2020)*, pages 1597–1607, 2020.

[8] Kenny Davila and Richard Zanibbi. Layout and Semantics: Combining Representations for Mathematical Formula Search. In *Proceedings of the 40th International ACM SIGIR Con-*

ference on Research and Development in Information Retrieval, SIGIR '17, pages 1165–1168, New York, NY, USA, August 2017. Association for Computing Machinery.

[9] Matthias Fey and Jan Eric Lenssen. Fast Graph Representation Learning with PyTorch Geometric. In *ICLR 2019 Workshop on Representation Learning on Graphs and Manifolds*, 2019.

[10] Dallas Fraser, Andrew Kane, and Frank Wm. Tompa. Choosing Math Features for BM25 Ranking with Tangent-L. In *Proceedings of the ACM Symposium on Document Engineering 2018*, pages 1–10, Halifax NS Canada, August 2018. ACM.

[11] Qiankun Fu, Linfeng Song, Wenyu Du, and Yue Zhang. End-to-end AMR coreference resolution. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli, editors, *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4204–4214, Online, August 2021. Association for Computational Linguistics.

[12] Liangcai Gao, Zhuoren Jiang, Yue Yin, Ke Yuan, Zuoyu Yan, and Zhi Tang. Preliminary Exploration of Formula Embedding for Mathematical Information Retrieval: can mathematical formulae be embedded like a natural language?, August 2017. arXiv:1707.05154 [cs].

[13] Siqi Gao and Yiu-Kai Dennis Ng. Recommending Answers to Math Questions Based on KL-Divergence and Approximate XML Tree Matching. In *Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval in the Asia Pacific Region*, SIGIR-AP '23, pages 21–31, New York, NY, USA, November 2023. Association for Computing Machinery.

[14] Martin Geletka, Vojtěch Kalivoda, Michal Štefánik, Marek Toma, and Petr Sojka. MIRMU and MSM at ARQMath 2022. In *Working Notes of CLEF 2022 – Conference and Labs of the Evaluation Forum*, Bologna, Italy, 2022.

[15] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural Message Passing for Quantum Chemistry. In *Proceedings of the 34th International Conference on Machine Learning (ICML 2017)*, pages 1263–1272, 2017.

[16] Alex Graves and Jürgen Schmidhuber. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks: The Official Journal of the International Neural Network Society*, 18(5-6):602–610, 2005.

[17] Radu Hambasan, Michael Kohlhase, and Corneliu Prodescu. MathWebSearch at NTCIR-11. In *Proceedings of the 11th NTCIR Workshop Meeting on Evaluation of Information Access Technologies*, Tokyo, Japan, 2014.

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR 2016)*, pages 770–778. IEEE, 2016.

[19] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus), 2016.

[20] Andrew Kane. Dowsing for Answers to Math Questions: Doing Better with Less. In *Working Notes of CLEF 2022 – Conference and Labs of the Evaluation Forum*, Bologna, Italy, 2022.

[21] Tuğrul Hasan Karabulut and İnci M. Baytaş. Local Virtual Nodes for Alleviating Over-Squashing in Graph Neural Networks, August 2025.

[22] Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense Passage Retrieval for Open-Domain Question Answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781. Association for Computational Linguistics, 2020.

[23] Besat Kassaie, Andrew Kane, and Frank Wm. Tompa. Exploiting Query Reformulation and Reciprocal Rank Fusion in Math-Aware Search Engines. In *ACM Symposium on Document Engineering (DocEng '25)*, Nottingham, United Kingdom, 2025.

[24] Omar Khattab and Matei Zaharia. ColBERT: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 39–48, New York, NY, USA, July 2020. ACM.

[25] Thomas N. Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations (ICLR 2017)*, 2017.

[26] Michael Kohlhase, Bogdan A. Matican, and Corneliu-Claudiu Prodescu. MathWebSearch 0.5: Scaling an Open Formula Search Engine. In Johan Jeuring, John A. Campbell, Jacques Carette, Gabriel Dos Reis, Petr Sojka, Makarius Wenzel, and Volker Sorge, editors, *Intelligent Computer Mathematics*, pages 342–357, Berlin, Heidelberg, 2012. Springer.

[27] Giovanni Yoko Kristianto, Goran Topic, and Akiko Aizawa. MCAT Math Retrieval System for NTCIR-12 MathIR Task. In *Proceedings of the 12th NTCIR Workshop Meeting on Evaluation of Information Access Technologies*, Tokyo, Japan, 2016.

[28] Kriste Krstovski and David M. Blei. Equation Embeddings, March 2018. arXiv:1803.09123 [stat].

[29] Ray R Larson, Chloe J Reynolds, and Fredric C Gey. The Abject Failure of Keyword IR for Mathematics Search: Berkeley at NTCIR-10 Math. In *Proceedings of the 10th NTCIR Workshop Meeting on Evaluation of Information Access Technologies*, Tokyo, Japan, 2013.

[30] Zachary Levonian, Chenglu Li, Wangda Zhu, Anoushka Gade, Owen Henkel, Millie-Ellen Postle, and Wanli Xing. Retrieval-augmented generation to improve math question-answering: Trade-offs between groundedness and human preference. In *NeurIPS'23 Workshop on Generative AI for Education (GAIED)*, 2023.

[31] Ruyin Li and Xiaoyu Chen. SSEmb: A Joint Structural and Semantic Embedding Framework for Mathematical Formula Retrieval, August 2025. arXiv:2508.04162 [cs].

[32] Ilya Loshchilov and Frank Hutter. Decoupled Weight Decay Regularization. In *International Conference on Learning Representations (ICLR 2019)*, 2019.

[33] Behrooz Mansouri. Survey of Abstract Meaning Representation: Then, Now, Future, May 2025. arXiv:2505.03229 [cs].

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

[35] Behrooz Mansouri, Douglas W. Oard, and Richard Zanibbi. Contextualized Formula Search Using Math Abstract Meaning Representation. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, pages 4329–4333, Atlanta GA USA, October 2022. ACM.

[36] Behrooz Mansouri, Douglas W Oard, and Richard Zanibbi. DPRL Systems in the CLEF 2022 ARQMath Lab: Introducing MathAMR for Math-Aware Search. In *Working Notes of CLEF 2022 – Conference and Labs of the Evaluation Forum*, Bologna, Italy, 2022.

[37] Behrooz Mansouri, Shaurya Rohatgi, Douglas W. Oard, Jian Wu, C. Lee Giles, and Richard Zanibbi. 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, Santa Clara CA USA, September 2019. ACM.

[38] Behrooz Mansouri, Richard Zanibbi, Douglas W. Oard, and Anurag Agarwal. Overview of ARQMath-2 (2021): Second CLEF Lab on Answer Retrieval for Questions on Math. In

K. Selçuk Candan, Bogdan Ionescu, Lorraine Goeuriot, Birger Larsen, Henning Müller, Alexis Joly, Maria Maistro, Florina Piroi, Guglielmo Faggioli, and Nicola Ferro, editors, *Experimental IR Meets Multilinguality, Multimodality, and Interaction*, pages 215–238, Cham, 2021. Springer International Publishing.

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

[40] Bruce R. Miller and Abdou Youssef. Technical Aspects of the Digital Library of Mathematical Functions. *Annals of Mathematics and Artificial Intelligence*, 38(1):121–136, May 2003.

[41] Tahira Naseem, Austin Blodgett, Sadhana Kumaravel, Tim O'Gorman, Young-Suk Lee, Jeffrey Flanigan, Ramón Astudillo, Radu Florian, Salim Roukos, and Nathan Schneider. DocAMR: Multi-Sentence AMR Representation and Evaluation. In Marine Carpuat, Marie-Catherine de Marneffe, and Ivan Vladimir Meza Ruiz, editors, *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3496–3505, Seattle, United States, July 2022. Association for Computational Linguistics.

[42] Yin Ki Ng, Dallas J Fraser, Besat Kassaie, George Labahn, Mirette S Marzouk, Frank Wm Tompa, and Kevin Wang. Dowsing for Math Answers with Tangent-L. In *Working Notes of CLEF 2020 – Conference and Labs of the Evaluation Forum*, Thessaloniki, Greece, 2020.

[43] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation Learning with Contrastive Predictive Coding, January 2019. arXiv:1807.03748 [cs].

[44] Shuai Peng, Ke Yuan, Liangcai Gao, and Zhi Tang. MathBERT: A Pre-Trained Model for Mathematical Formula Understanding, May 2021. arXiv:2105.00377 [cs].

[45] Lukas Pfahler and Katharina Morik. Semantic Search in Millions of Equations. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '20, pages 135–143, New York, NY, USA, August 2020. Association for Computing Machinery.

[46] Trang Pham, Truyen Tran, Hoa Dam, and Svetha Venkatesh. Graph Classification via Deep Learning with Virtual Nodes, August 2017. arXiv:1708.04357 [cs].

[47] Martin F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

[48] Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Lan-*

guage Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pages 3982–3992. Association for Computational Linguistics, 2019.

[49] Anja Reusch, Julius Gonsior, Claudio Hartmann, and Wolfgang Lehner. Investigating the Usage of Formulae in Mathematical Answer Retrieval. In *Advances in Information Retrieval: 46th European Conference on Information Retrieval, ECIR 2024, Glasgow, UK, March 24–28, 2024, Proceedings, Part I*, pages 247–261, Berlin, Heidelberg, March 2024. Springer-Verlag.

[50] Anja Reusch, Maik Thiele, and Wolfgang Lehner. Transformer-Encoder-Based Mathematical Information Retrieval. In Alberto Barrón-Cedeño, Giovanni Da San Martino, Mirko Degli Esposti, Fabrizio Sebastiani, Craig Macdonald, Gabriella Pasi, Allan Hanbury, Martin Potthast, Guglielmo Faggioli, and Nicola Ferro, editors, *Experimental IR Meets Multilinguality, Multimodality, and Interaction*, pages 175–189, Cham, 2022. Springer International Publishing.

[51] Rochester Institute of Technology. Research computing services, 2025.

[52] Ankit Satpute, Noah Giessing, Andre Greiner-Petter, Moritz Schubotz, Olaf Teschke, Akiko Aizawa, and Bela Gipp. Can LLMs Master Math? Investigating Large Language Models on Math Stack Exchange. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, Washington D.C., USA, July 2024. ACM.

[53] Michael Schlichtkrull, Thomas N. Kipf, Peter Bloem, Rianne van den Berg, Ivan Titov, and Max Welling. Modeling Relational Data with Graph Convolutional Networks. In *The Semantic Web: 15th International Conference, ESWC 2018*, pages 593–607, Cham, 2018. Springer.

[54] Petr Sojka and Martin Líška. Indexing and Searching Mathematics in Digital Libraries. In James H. Davenport, William M. Farmer, Josef Urban, and Florian Rabe, editors, *Intelligent Computer Mathematics*, pages 228–243, Berlin, Heidelberg, 2011. Springer.

[55] Petr Sojka, Michal Růžička, and Vít Novotný. MIaS: Math-Aware Retrieval in Digital Mathematical Libraries. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 1923–1926, October 2018. arXiv:1808.09224 [cs].

[56] Afonso Sousa and Henrique Cardoso. SAPG: Semantically-Aware Paraphrase Generation with AMR Graphs. In *Proceedings of the 17th International Conference on Agents and Artificial Intelligence (ICAART 2025)*, pages 861–871, Porto, Portugal, February 2025. SCITEPRESS.

[57] Joshua Southern, Francesco Di Giovanni, Michael Bronstein, and Johannes F. Lutzeyer. Understanding Virtual Nodes: Oversquashing and Node Heterogeneity, May 2024.

[58] Abhinav Thanda, Ankit Agarwal, Kushal Singla, Aditya Prakash, and Abhishek Gupta. A Document Retrieval System for Math Queries. In *Proceedings of the 12th NTCIR Workshop Meeting on Evaluation of Information Access Technologies*, Tokyo, Japan, 2016.

[59] Goran Topic, Giovanni Yoko Kristianto, and Minh-Quoc Nghiem. The MCAT Math Retrieval System for NTCIR-10 Math Track. In *Proceedings of the 10th NTCIR Workshop Meeting on Evaluation of Information Access Technologies*, Tokyo, Japan, 2013.

[60] Sumedh Vemuganti, Ayu Seiya, and Nickvash Kani. Advancing Math Formula Search Using Diverse Structural and Symbolic Representations. In Claudia Hauff, Craig Macdonald, Dietmar Jannach, Gabriella Kazai, Franco Maria Nardini, Fabio Pinelli, Fabrizio Silvestri, and Nicola Tonellotto, editors, *Advances in Information Retrieval*, pages 116–131, Cham, 2025. Springer Nature Switzerland.

[61] Eric Wallace, Yizhong Wang, Sujian Li, Sameer Singh, and Matt Gardner. Do NLP Models Know Numbers? Probing Numeracy in Embeddings. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*, pages 5307–5315. Association for Computational Linguistics, 2019.

[62] Abdou Youssef. Roles of Math Search in Mathematics. In Jonathan M. Borwein and William M. Farmer, editors, *Mathematical Knowledge Management*, pages 2–16, Berlin, Heidelberg, 2006. Springer.

[63] Richard Zanibbi, Akiko Aizawa, and Michael Kohlhase. NTCIR-12 MathIR Task Overview. In *Proceedings of the 12th NTCIR Workshop Meeting on Evaluation of Information Access Technologies*, Tokyo, Japan, 2016.

[64] Richard Zanibbi, Kenny Davila, Andrew Kane, and Frank Wm. Tompa. 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, Pisa Italy, July 2016. ACM.

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

[66] Richard Zanibbi, Douglas W. Oard, Anurag Agarwal, and Behrooz Mansouri. Overview of ARQMath 2020: CLEF Lab on Answer Retrieval for Questions on Math. In Avi Arampatzis, Evangelos Kanoulas, Theodora Tsikrika, Stefanos Vrochidis, Hideo Joho, Christina Lioma, Carsten Eickhoff, Aurélie Névéol, Linda Cappellato, and Nicola Ferro, editors, *Experimental*

IR Meets Multilinguality, Multimodality, and Interaction, pages 169–193, Cham, 2020. Springer International Publishing.

[67] Wei Zhong, Sheng-Chieh Lin, Jheng-Hong Yang, and Jimmy Lin. One Blade for One Purpose: Advancing Math Information Retrieval using Hybrid Search. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 141–151, Taipei Taiwan, July 2023. ACM.

[68] Wei Zhong, Yuqing Xie, and Jimmy Lin. Answer Retrieval for Math Questions Using Structural and Dense Retrieval. In Avi Arampatzis, Evangelos Kanoulas, Theodora Tsikrika, Stefanos Vrochidis, Anastasia Giachanou, Dan Li, Mohammad Aliannejadi, Michalis Vlachos, Guglielmo Faggioli, and Nicola Ferro, editors, *Experimental IR Meets Multilinguality, Multimodality, and Interaction*, pages 209–223, Cham, 2023. Springer Nature Switzerland.

[69] Wei Zhong and Richard Zanibbi. Structural Similarity Search for Formulas Using Leaf-Root Paths in Operator Subtrees. In Leif Azzopardi, Benno Stein, Norbert Fuhr, Philipp Mayr, Claudia Hauff, and Djoerd Hiemstra, editors, *Advances in Information Retrieval*, pages 116–129, Cham, 2019. Springer International Publishing.

Appendices

Appendix A

ARQMath-3 Query Examples

This appendix presents real examples from the ARQMath-3 test collection to illustrate the nature of each task and the difficulty of the relevance judgments. For each task, one topic is shown with one judged result at each of the four relevance levels (3, 2, 1, 0). The examples are drawn from the official ARQMath-3 qrels.

A.1 Task 1: Answer Retrieval (Topic A.302)

Task 1 requires retrieving relevant answer posts from Math Stack Exchange in response to a full question post. The topic below (A.302) asks about the n-th root of a complex number and illustrates how the same underlying mathematical concept can yield answers that range from a complete rigorous derivation down to discussion of entirely unrelated topics.

Topic A.302 — n-th root of a complex number

Title: n-th root of a complex number

Question: I am confused about the following problem. With $w = se^{i\phi}$, where $s \geq 0$ and $\phi \in \mathbb{R}$, solve the equation $z^n = w$ in $\mathbb{C}$ where n is a natural number. How many solutions are there?
My approach is simply taking the n-th root, which gives $z = \sqrt[n]{s}\, e^{i\varphi/n}$. However, it seems that this problem is asking me to show the *existence* of the n-th root. Can I assume that the n-th root of a complex number already exists? Moreover, would I be correct to say that there is only one solution?

Tags: real-analysis, complex-analysis, analysis

Figure A.1: ARQMath-3 Task 1 topic A.302.

The four example answers below are all drawn from the judged pool for this topic.

Relevance 3 (High — fully answers the question) Post 3346

More generally, if you want to compute all the n-th roots of a complex number z_0, write both sides in exponential form: $z = re^{i\theta}$, $z_0 = r_0 e^{i\theta_0}$. Then $z^n = z_0$ becomes

$$r^n e^{in\theta} = r_0 e^{i\theta_0}.$$

Matching moduli gives $r = \sqrt[n]{r_0}$. Matching arguments (up to multiples of 2π) gives

$$\theta = \frac{\theta_0 + 2k\pi}{n}, \quad k = 0, 1, \ldots, n-1.$$

The values $k \geq n$ repeat, so there are exactly n distinct roots:

$$\sqrt[n]{r_0\, e^{i\theta_0}} = \sqrt[n]{r_0}\; e^{i\frac{\theta_0+2k\pi}{n}}, \quad k = 0, 1, \ldots, n-1.$$

Figure A.2: Relevance 3 answer for Task 1 topic A.302. The post gives a complete derivation establishing both the number of roots and their explicit form.

Relevance 2 (Medium — provides a useful step or formula) Post 322487

If $z = |z|\, e^{i\theta} \in \mathbb{C}^*$, then there are exactly n numbers w_k, $k \in \{0, \ldots, n-1\}$, such that $w_k^n = z$:

$$w_k = \sqrt[n]{|z|}\; e^{i(2k\pi+\theta)/n}, \quad k = 0, \ldots, n-1.$$

The principal root is the case $k = 0$.

Figure A.3: Relevance 2 answer for Task 1 topic A.302. The post states the correct formula and identifies the principal root, but does not derive the result or explain why $k \geq n$ values repeat.

Relevance 1 (Low — useful context, does not answer directly) Post 41881

To make complete sense of this, we need to look at things in the complex plane. When we choose to take the positive square root, we are just choosing a particular branch cut of the complex logarithm. In general, given a number $x \neq 0$, x will have n distinct n-th roots.
You noticed that -2 and 2 are both square roots of 4. What about the cube roots of $2^{3/2}$? We have $\sqrt{2}$, but also $-1 + i$ and $-1 - i$. Similarly, the fourth root of 16 includes 2 and -2, but also $2i$ and $-2i$.
In general, taking n-th roots is a multivalued function, so we have to make a choice, and this corresponds to choosing a branch cut for the logarithm function.

Figure A.4: Relevance 1 answer for Task 1 topic A.302. The post gives helpful intuition about multivaluedness and branch cuts, but does not provide the formula or derivation the question asks for.

Relevance 0 (Not relevant) Post 13550

It is rather standard for ζ_n to denote a primitive n-th root of unity in the algebraic closure of the field k (possible if and only if the characteristic of k does not divide n). When the characteristic does not divide n, there are precisely $\varphi(n)$ primitive n-th roots of unity, where φ is Euler's phi function.
In a context where k is a subfield of $\mathbb{C}$, it is also standard for ζ_n to denote the specific primitive n-th root of unity $e^{2\pi i/n}$, i.e., the one of minimal argument in the complex plane.

Figure A.5: Relevance 0 answer for Task 1 topic A.302. The post discusses notation for primitive roots of unity in abstract algebra, which is unrelated to the question of how to compute or derive the n-th roots of a given complex number.

The contrast across relevance levels illustrates the core difficulty of Task 1. Posts at relevance 3 and 2 address the same mathematical object (the n-th root formula in polar form) but differ in whether they include the derivation and explanation of why exactly n roots exist. The relevance 1 post mentions multivaluedness without providing the formula. The relevance 0 post uses the variable n in a related algebraic setting (roots of unity) but does not address the question at all. A retrieval system must distinguish between these cases, often in the presence of overlapping notation and surface-level textual similarity.

A.2 Task 2: Formula Retrieval (Topic B.308)

Task 2 requires retrieving relevant formulas from the collection in response to a specific query formula embedded in a question post. The query formula for topic B.308 appears in a question about Riemann's integral representation of the zeta function (topic A.308). Assessors judged each candidate formula based on how useful it is in the context of the surrounding post.

Topic B.308 — Post A.308: Riemann's definition of the zeta function

Tags: contour-integration, riemann-zeta

I am having trouble understanding Riemann's definition of the zeta function, and I will need to give a brief summary here before I can get to my question.
In his 1859 paper, Riemann derived the integral representation

Query formula (id: q_62)

$$\zeta(s) = \sum_{n=1}^{\infty} \frac{1}{n^s} = \frac{1}{\Gamma(s)} \int_0^{\infty} \frac{x^{s-1}}{e^x - 1}\, dx$$

that is valid for $\mathrm{Re}(s) > 1$, and then modified the integral in order to define a function that is defined for all complex values of s, except $s = 1$, where it has a simple pole. The extension is given by

$$\zeta(s) = \frac{\Gamma(1-s)}{2\pi i} \int_C \frac{(-z)^s}{e^z - 1} \frac{dz}{z}$$

where C is a "Hankel contour," that is, a path that travels from $+\infty$ at a small distance ϵ above the positive x-axis, circles around the origin once counterclockwise with radius δ, and returns to $+\infty$ at distance ϵ below the positive real axis. Taking the limit as $\epsilon \to 0$ and $\delta \to 0$, the integral $\int_C \frac{(-z)^s}{e^z-1} \frac{dz}{z}$ becomes $(e^{i\pi s} - e^{-i\pi s}) \int_0^{\infty} \frac{x^{s-1}}{e^x-1} dx$, and the rest follows from known identities for the Gamma function. While the original real integral over $[0, \infty)$ is divergent if $\mathrm{Re}(s) \leq 1$, the contour integral over C is defined for all complex s, because the path stays away from the singularity at $s = 0$ and from the branch cut along the positive x-axis. My problem is understanding why the integral over C does not depend on ϵ and δ, so that we can keep them at a safe positive distance from the singularities for the definition, but take the limit for the purpose of evaluating the integral. I know that by Cauchy's theorem we can modify a path of integration (without changing the value) as long as we do not cross any singularity, but this path starts and ends at infinity, so I am not sure how to rigorously proceed. Even if I start the path at $R + i\epsilon$ and end it at $R - i\epsilon$ for some large R, the starting and ending points change as ϵ changes.

Figure A.6: ARQMath-3 Task 2 topic B.308. The full question post is shown. The highlighted box marks the query formula (id q_62) that defines this Task 2 topic within the post.

Relevance 3 (Equivalent or direct match)

$$\zeta(s) = \frac{1}{\Gamma(s)} \int_0^{\infty} \frac{x^{s-1}}{e^x - 1}\, dx$$

Figure A.7: Relevance 3 formula for Task 2 topic B.308. This is the integral representation component of the query formula, differing only in that the Dirichlet series definition is omitted.

Relevance 2 (Useful but not a direct match)

$$\zeta(s) = 2^s\,\pi^{s-1}\sin\Big(\frac{\pi s}{2}\Big)\,\Gamma(1-s)\,\zeta(1-s)$$

Figure A.8: Relevance 2 formula for Task 2 topic B.308. This is the functional equation relating $\zeta(s)$ to $\zeta(1-s)$, which is directly connected to the analytic continuation discussed in the surrounding post, but is structurally distinct from the integral representation in the query.

Relevance 1 (Some chance of usefulness)

$$\zeta(s) = \sum_{n=1}^{\infty} \int_n^{n+1} \left(\frac{1}{n^s} - \frac{1}{x^s}\right) dx + \frac{1}{s-1}$$

Figure A.9: Relevance 1 formula for Task 2 topic B.308. Another representation of $\zeta(s)$ using sums and integrals, but one that is structurally distinct from the query and does not directly address the question being asked in the post.

Relevance 0 (Not relevant)

$$f'(h) = \frac{d}{dh}\big\langle \gamma(s+h) - \gamma(s),\, \gamma(s+h) - \gamma(s) \big\rangle$$

Figure A.10: Relevance 0 formula for Task 2 topic B.308. This formula appears in a post about arc-length parameterization of curves in differential geometry. The variable s is used as a path parameter, not as the complex argument of the zeta function, and the formula has no relation to the query.

Task 2 illustrates a different retrieval challenge from Task 1. The relevance 3 formula is a direct structural subexpression of the query. The relevance 2 formula is the functional equation for $\zeta(s)$, which is topically connected to the post but structurally quite different from the query. The relevance 1 formula is another representation of $\zeta(s)$ that does not address the specific question. The relevance 0 formula shares variable names with the query post but belongs to an entirely different mathematical context.

Appendix B

Example Retrieval Results

This appendix presents example retrieval results from the formula representations experiment (discussed in Section 4.3.1). For each experimental condition (Both OPT+SLT, OPT-only, SLT-only, and None), the top-3 retrieval results for the ARQMath-3 topic A.346 is shown (Figure B.1), ranked by MaxSim score. All conditions were trained on a subset of 50,000 positive pairs (45,000 train / 5,000 validation) with a scaled-down model (hyperparamers listed in Table 4.5). Relevance ratings follow the ARQMath four-level scheme: 3 (high, fully answers the question), 2 (medium, provides a useful step or clarification), 1 (low, context that might help interpretation), 0 (not relevant).

Topic A.346

Title: Greatest lower bound in Q

Question: I have a set $\{r \in \mathbb{Q} \mid r^2 > 2, r > 0\}$. I was wondering why it does not have the greatest lower bound. Isn't $0 \in \mathbb{Q}$ a greatest lower bound for this set in rational numbers?

Figure B.1: Topic A.346 from the ARQMath-3 test collection.

This question asks whether the set $\{r \in \mathbb{Q} \mid r^2 > 2,\, r > 0\}$ has a greatest lower bound in $\mathbb{Q}$. The key point is that the natural lower bound is $\sqrt{2}$, but this number is not rational. This shows that the set does not have a greatest lower bound in $\mathbb{Q}$.

B.1 Both (OPT + SLT)

Rank 1 Relevance: 2

Let us consider the set of rational numbers

$$S = \{r \in \mathbb{Q} \mid r \geq 1 \text{ and } r^2 \leq 29\}.$$

The supremum of the set equals $\sqrt{29}$. It is more interesting to show that there does not exist a supremum of this set in $\mathbb{Q}$. We show the following:

1. S does not have a largest element.
2. While S has majorants in $\mathbb{Q}$ (for instance 6), it does not have a smallest majorant in $\mathbb{Q}$.
3. Let $r = \frac{m}{n} \in S$. Then
$$\left(\frac{m}{n}\right)^2 \leq 29 \quad \Rightarrow \quad 29n^2 - m^2 \geq 0.$$
By a number-theoretic argument, equality cannot hold, so
$$29n^2 - m^2 \geq 1.$$
Rewrite this as
$$(\sqrt{29}n - m)(\sqrt{29}n + m) \geq 1.$$
Hence
$$\sqrt{29}n - m \geq \frac{1}{\sqrt{29}n + m},$$
and dividing by n gives
$$\sqrt{29} - \frac{m}{n} \geq \frac{1}{n^2\left(\sqrt{29} + \frac{m}{n}\right)} \geq \frac{1}{2\sqrt{29}\,n^2} > \frac{1}{12n^2}.$$

We can rephrase this as follows:
If $\frac{m}{n} \in S$, then
$$\frac{m}{n} + \frac{1}{12n^2} \in S.$$
Thus, S does not have a largest element.
From the above, the set of rationals that are upper bounds for S is

$$T = \{t \in \mathbb{Q} \mid t \geq 0 \text{ and } t^2 > 29\}.$$

Let $\frac{m}{n} \in T$. In a similar way as above, one can show that

$$\frac{m}{n} - \frac{1}{2mn} \in T.$$

Hence, the set T does not have a smallest element.
We conclude that S does not have a supremum in $\mathbb{Q}$.

Rank 2 Relevance: 0

If we consider S as a subset of $\mathbb{R}$, then

$$S = (-\sqrt{2}, \sqrt{2}) \cap \mathbb{Q},$$

and thus

$$\inf S = -\sqrt{2}, \quad \sup S = \sqrt{2}.$$

If we instead consider S as a subset of $\mathbb{Q}$, then neither the infimum nor the supremum exists in $\mathbb{Q}$. Informally, this is because for any $q \in \mathbb{Q}$, there is always another rational number strictly between q and $\sqrt{2}$. If a rigorous proof that $\sup S$ does not exist in $\mathbb{Q}$ is desired, one can construct such an argument explicitly.

Rank 3 Relevance: 0

If you substitute 4 for 2 in your proof, then the conclusion becomes false:

$$\{r \in \mathbb{Q} \mid r > 0 \text{ and } r^2 < 4\}$$

does have a least upper bound in $\mathbb{Q}$ (because 4 is a perfect square). Therefore, at some point in the argument, you must use the fact that if $r \in \mathbb{Q}$, then $r^2 \neq 2$.
Thus, your proof should have two parts:

1. Show that if $r \in \mathbb{Q}$, then $r^2 \neq 2$.
2. Show that if $r \in \mathbb{Q}$ and $r^2 > 2$, then there exists $t \in \mathbb{Q}$ such that

$$2 < t^2 < r^2.$$

If you can prove these, then the result follows: any upper bound r must satisfy $r^2 > 2$ (by part 1), so it cannot be a *least* upper bound (by part 2).
Edited to reply to comment: If $r^2 > 2$, then we can write

$$r^2 = 2 + \epsilon$$

for some rational $\epsilon > 0$. Now define

$$t = r - \frac{\epsilon}{2r}.$$

Figure B.2: Retrieval results for the *Both (OPT+SLT)* model on Topic A.346.

Observation. The top result is closely related to the query because it also studies bounds of a set defined by a quadratic inequality and uses an irrational number as a limit. The other results are less directly related, but still discuss upper and lower bounds in $\mathbb{Q}$.

B.2 OPT-Only

Rank 1 Relevance: 2

Let us consider the set of rational numbers

$$S = \{r \in \mathbb{Q} \mid r \geq 1 \text{ and } r^2 \leq 29\}.$$

The supremum of this set is $\sqrt{29}$. It is more interesting, however, to show that S does not have a supremum in $\mathbb{Q}$. We establish the following:

1. S does not have a largest element.
2. While S has upper bounds in $\mathbb{Q}$ (for instance, 6), it does not have a smallest upper bound in $\mathbb{Q}$.
3. Let $r = \frac{m}{n} \in S$. Then
$$\left(\frac{m}{n}\right)^2 \leq 29 \quad \Rightarrow \quad 29n^2 - m^2 \geq 0.$$
A number-theoretic argument shows that equality cannot hold, so
$$29n^2 - m^2 \geq 1.$$
Rewrite this as
$$(\sqrt{29}\,n - m)(\sqrt{29}\,n + m) \geq 1,$$
which implies
$$\sqrt{29}\,n - m \geq \frac{1}{\sqrt{29}\,n + m}.$$
Dividing by n gives
$$\sqrt{29} - \frac{m}{n} \geq \frac{1}{n^2\left(\sqrt{29} + \frac{m}{n}\right)} \geq \frac{1}{2\sqrt{29}\,n^2} > \frac{1}{12n^2}.$$

We can rephrase this as follows: if $\frac{m}{n} \in S$, then

$$\frac{m}{n} + \frac{1}{12n^2} \in S.$$

Thus, S does not have a largest element.
From the above, the set of rational numbers t such that $t \geq s$ for all $s \in S$ is

$$T = \{t \in \mathbb{Q} \mid t \geq 0 \text{ and } t^2 > 29\}.$$

Let $\frac{m}{n} \in T$. By a similar argument, one can show that

$$\frac{m}{n} - \frac{1}{2mn} \in T.$$

Hence, T does not have a smallest element.
We conclude that S does not have a supremum in $\mathbb{Q}$.

Rank 2 Relevance: 1

The set $\mathbb{Q}$ of all rational numbers, equipped with the usual linear order, is an infinite distributive lattice that is not complete. For example, $\mathbb{Q}$ itself has neither a least upper bound nor a greatest lower bound in $\mathbb{Q}$. Likewise, the bounded nonempty set

$$\{x \in \mathbb{Q} \mid x^2 < 2\}$$

has neither a least upper bound nor a greatest lower bound in $\mathbb{Q}$.

Rank 3 Relevance: 0

1. Any finite set is countable (check the definition of countable).
2. Define $f : \mathbb{Z} \to \mathbb{N}$ as follows:
$$f(n) = \begin{cases} 2n, & \text{if } n \in \mathbb{N}, \\ 2|n| - 1, & \text{if } n < 0. \end{cases}$$
3. Define $f : \mathbb{S} \to \mathbb{N}$ as follows:
$$f(p, q) = \frac{(p + q - 1)(p + q)}{2} + p + 1, \quad \forall (p, q) \in \mathbb{S}.$$

This can be seen as "counting elements in a table by diagonal", a standard argument used to show that $\mathbb{Q}$ is countable.
See: `https://www.google.fr/?gws_rd=ssl#q=count+rational+numbers`

Figure B.3: Retrieval results for the *OPT-only* model on Topic A.346.

Observation. The top result matches the main idea of the query by studying bounds of a set defined by a quadratic inequality. The lower-ranked results include other topics, such as a proof about counting rational numbers, which is not directly related to the query.

B.3 SLT-Only

Rank 1 Relevance: 2

Did you intend to write $r^2 < 5$? In that case, no supremum exists within $\mathbb{Q}$; since $\sqrt{5}$ is irrational, we have
$$\sup\{r \in \mathbb{Q} : r^2 < 5\} = \sqrt{5}$$
in $\mathbb{R}$. Indeed, the set has no maximum.
If you really meant $r < 5$, then the set has a supremum in both $\mathbb{Q}$ and $\mathbb{R}$, namely 5. By definition, for
$$\{r \in \mathbb{Q} : r < 5\},$$
we see that 5 is an upper bound. On the other hand, any upper bound must be ≥ 5, so 5 is the least upper bound. Like the previous set, it has no maximum. For any $a < 5$, we have
$$a < \frac{a + 5}{2} < 5.$$

Rank 2 Relevance: 1

The significance of the set

$$S := \{x \in \mathbb{Q}_{>0} \mid x^2 < 2\}$$

is the following: it is a nonempty subset of $\mathbb{Q}$ which is clearly bounded above, but which has no least upper bound (supremum) in $\mathbb{Q}$. As a consequence, $\mathbb{Q}$ is not order-complete and is therefore naturally extended to the larger set of real numbers $\mathbb{R}$.
In order to show that S has no supremum in $\mathbb{Q}$, it suffices to show that no number $c \in \mathbb{Q}_{>0}$ can serve as a supremum of S.
Given any candidate $c \in \mathbb{Q}_{>0}$, we know that $c^2 \neq 2$, hence either $c^2 < 2$ or $c^2 > 2$. Define the auxiliary number

$$\xi := \frac{2c+2}{c+2} \in \mathbb{Q}_{>0}.$$

Then we compute

$$\xi - c = \frac{2-c^2}{c+2}, \qquad \xi^2 - 2 = \frac{2(c^2-2)}{(c+2)^2}.$$

Now consider the two cases:
(a) If $c^2 < 2$, then $\xi > c$ and $\xi^2 < 2$, hence $\xi \in S$. Therefore, c is not an upper bound for S.
(b) If $c^2 > 2$, then $\xi < c$ and $\xi^2 > 2$. Since $t \mapsto t^2$ is strictly increasing for $t > 0$, it follows that $\xi > x$ for all $x \in S$, so ξ is an upper bound for S that is strictly smaller than c.
In both cases, c fails to be a supremum of S.
There is no systematic procedure for "inventing" such a ξ; one typically experiments with inequalities until an expression with the desired properties is obtained.

Rank 3 Relevance: 0

If A is any nonempty set of real numbers that has an upper bound, then $\sup A$ exists by the completeness property of the real numbers. So it suffices to show that A is nonempty and has an upper bound.
It is straightforward to verify that A is nonempty (for example, it suffices to exhibit an element of A). To say that A has an upper bound means that there exists an element $r \in \mathbb{R}$ such that $x \leq r$ for all $x \in A$. That is, we want a number r such that

$$x \leq r \quad \text{for all } x \in \mathbb{Q} \text{ with } x^3 < 2.$$

Can you think of such a number?
(It turns out that $\sup A = 2^{1/3}$, but you do not need to prove this here; the task is simply to identify a plausible candidate for the supremum.)

Figure B.4: Retrieval results for the *SLT-only* model on Topic A.346.

Observation. Most results match the shape of the set notation and inequalities. However, some results change the condition (for example, using $r^2 < 5$ instead of $r^2 > 2$), which changes the meaning of the set.

B.4 No Formulas

Rank 1 Relevance: 0

The number a that Jesse wants to exist is a *lower bound* of the set Z.
If a set of real numbers has only finitely many elements, then it must have both a greatest element and a least element. Therefore, it has both an upper bound and a lower bound.
However, an infinite set of real numbers may fail to have an upper bound, and it may also fail to have a lower bound.
Recall that every set is a subset of itself. What real number is less than or equal to every element of $\mathbb{R}$?

Rank 2 Relevance: 0

Hint: Consider $\mathbb{Q}$ intersected with any bounded set, whether finite or infinite. Since $\mathbb{Q}$ is countable, the resulting set is at most countable, and it can clearly be infinite; for example, $[0, 1] \cap \mathbb{Q}$ is bounded and countably infinite.

Rank 3 Relevance: 1

A lower bound of a set of numbers is a number that is less than or equal to every element of the set.
The greatest lower bound of a set of numbers is a number that is a lower bound of the set and is greater than or equal to every other lower bound of the set. The notation $\inf A$ denotes the greatest lower bound (infimum) of the set A. For example, $\inf A = 5$ means that the greatest lower bound of the set A is 5. The greatest lower bound is also called the infimum.

Figure B.5: Retrieval results for the *None* model on Topic A.346.

Observation. The results are mostly definitions or general explanations of bounds and infimum. They do not focus on the specific set condition in the query.

Appendix C

Benchmark Retrieval Comparison: Topics A.301 and B.301

This appendix presents side-by-side retrieval results from the ARQMath-3 benchmark evaluation, comparing MathAMR+ against the original MathAMR approach (DPRL submission) on topics A.301 (Task 1) and B.301 (Task 2). The top-3 retrieved results from each system are shown for each topic. Relevance ratings follow the ARQMath four-level scheme: 3 (high, fully answers the question), 2 (medium, provides a useful step or clarification), 1 (low, context that might help interpretation), 0 (not relevant).

C.1 Task 1: Topic A.301

Topic A.301 asks to prove a matrix norm inequality. The query post is shown in Figure C.1.

Topic A.301 — Inequality between norm 1, norm 2 and norm ∞ of Matrices

Title: Inequality between norm 1, norm 2 and norm ∞ of Matrices

Question: Suppose A is a $m \times n$ matrix. Then prove that

$$\|A\|_2 \leq \sqrt{\|A\|_1 \|A\|_\infty}$$

I have proved the following relations:

$$\frac{1}{\sqrt{n}}\|A\|_\infty \leq \|A\|_2 \leq \sqrt{m}\|A\|_\infty$$
$$\frac{1}{\sqrt{m}}\|A\|_1 \leq \|A\|_2 \leq \sqrt{n}\|A\|_1$$

I feel that somehow Hölder's inequality for the special case when $p = 1$ and $q = \infty$ might be useful, but I could not prove that. I would like to have a proof that does not use $\|A\|_2 = \sqrt{\rho(A^T A)}$. Usage of inequalities like Cauchy–Schwarz or Hölder is fine.

Figure C.1: ARQMath-3 Task 1 topic A.301.

C.1.1 MathAMR+ Results

MathAMR+ achieves NDCG′=0.5510, MAP′=0.3066, P′@10=0.4000, BPref=0.6339 on this topic.

Rank 1 Post 1142010 Relevance: 2 Score: 0.928

The matrix p-norm is the operator norm:

$$\|A\|_p = \max_{x \neq 0} \frac{\|Ax\|_p}{\|x\|_p}. \tag{1}$$

The fact that

$$\|A\|_2 = \rho(A^* A) \quad \text{and} \quad \|A\|_\infty = \max_i \sum_j |a_{ij}| \tag{2}$$

is just the consequence of (1). The 2-norm and ∞-norm satisfy

$$\|x\|_\infty \leq \|x\|_2 \leq \sqrt{n}\|x\|_\infty \tag{3}$$

for any n-vector x, and $\|A\|_2 \leq \sqrt{n}\|A\|_\infty$ follows by plugging (3) into (1). From (4) one can show $\|A\|_2^2 = \rho(A^T A) \leq \|A^T A\|_\infty$. Since there is an eigenvector x and eigenvalue $\lambda = \rho(A^T A)$ of $A^T A$, the estimate

$$|\lambda| \, \|x\|_\infty = \|A^T A x\|_\infty \leq \|A^T A\|_\infty \, \|x\|_\infty$$

gives $\|A\|_2^2 \leq \|A^T A\|_\infty \leq \|A^T\|_\infty \, \|A\|_\infty = \|A\|_1 \, \|A\|_\infty$.

Rank 2 Post 901526 Relevance: 3 Score: 0.923

All mentioned matrix norms are special cases of the matrix p-norm:

$$\|A\|_p := \max_{x \neq 0} \frac{\|Ax\|_p}{\|x\|_p}. \tag{a}$$

If, for two vector p-norms $\|\cdot\|_p$ and $\|\cdot\|_q$ ($p, q \in [1, \infty]$), there are positive constants α and β such that $\alpha\|x\|_q \leq \|x\|_p \leq \beta\|x\|_q$ holds for all x, then

$$\frac{\alpha}{\beta}\|A\|_q \leq \|A\|_p \leq \frac{\beta}{\alpha}\|A\|_q. \tag{c}$$

To get the norm equivalences, use the well-known relations

$$\|x\|_\infty \leq \|x\|_2 \leq \sqrt{n}\|x\|_\infty, \quad \frac{1}{\sqrt{n}}\|x\|_1 \leq \|x\|_2 \leq \|x\|_1. \tag{d}$$

For the key inequality, the 2-norm of A equals the square root of the largest eigenvalue of A^*A: there exist $\lambda = \|A\|_2^2$ and nonzero y with $\|y\| = 1$ such that $A^*Ay = \lambda y$. For any operator norm $\|\cdot\|$,

$$\|A\|_2^2 = \lambda = \|A^*Ay\| \leq \|A^*\|\,\|A\|\,\|y\| = \|A\|\,\|A^*\|.$$

In particular,

$$\|A\|_2 \leq \sqrt{\|A\|_1\,\|A^*\|_1} = \sqrt{\|A\|_1\,\|A\|_\infty}. \tag{e}$$

Rank 3 Post 1394515 Relevance: 2 Score: 0.920

There are three kinds of matrix norms, each useful in different circumstances. The *entrywise* p-norm is defined by

$$\|A\| = \left(\sum_{i,j} |a_{ij}|^p\right)^{1/p}, \quad 1 \leq p \leq \infty.$$

The *induced* (operator) norm for a matrix viewed as a linear map $T : \mathbb{R}^m \to \mathbb{R}^n$ is

$$\|T\| = \sup_{v \neq 0} \frac{\|Tv\|}{\|v\|},$$

which gives the largest factor by which T stretches a vector. The induced 2-norm equals the largest singular value of A; the induced ∞-norm equals $\max_i \sum_j |a_{ij}|$; and the induced 1-norm equals $\max_j \sum_i |a_{ij}|$.

Figure C.2: Top-3 retrieval results from MathAMR+ for Task 1 topic A.301.

Observation. MathAMR+ retrieves results that directly address the relationship between matrix norms. The rank-2 result (post 901526, relevance 3) contains a complete proof of the target inequality via the operator norm framework; the rank-1 and rank-3 results are closely related (relevance 2) and provide additional norm equivalences and context.

C.1.2 MathAMR (Baseline) Results

The original MathAMR approach achieves NDCG′=0.3980, MAP′=0.1606, P′@10=0.3000, BPref=0.5589 on this topic.

Rank 1 Post 198740 Relevance: 1 Score: 0.413

Note that

$$\sum_{i \neq j} x_i x_j = (x_1 + \cdots + x_n)^2 - \|x\|^2 = |\langle e, x\rangle|^2 - \|x\|^2,$$

with $e = (1, \ldots, 1)^T$. Hence for given $c > 0$, you are looking for x satisfying

$$|\langle e, x\rangle| \leq \sqrt{1 + \frac{1}{c}}\, \|x\|.$$

If $K = \frac{1}{\sqrt{n}}\sqrt{1 + 1/c} \geq 1$, all x satisfy this by Cauchy–Schwarz; otherwise it is the set of x whose angle θ with e satisfies $|\cos\theta| \leq K$.

Rank 2 Post 2475352 Relevance: 0 Score: 0.399

By Young's inequality, for every $a > 0$:

$$2|xy| \leq a^2 x^2 + \frac{y^2}{a^2}.$$

Hence for $a, b > 0$:

$$(1 - a^2)x^2 + \left(3 - \frac{1}{a^2}\right) y^2 \leq x^2 + 2xy + 3y^2 \leq (1 + b^2)x^2 + \left(3 + \frac{1}{b^2}\right) y^2.$$

Choosing $a^2 = \sqrt{2} - 1$ and $b^2 = \sqrt{2} + 1$ gives

$$(2 - \sqrt{2})(x^2 + y^2) \leq x^2 + 2xy + 3y^2 \leq (2 + \sqrt{2})(x^2 + y^2).$$

Rank 3 Post 1049716 Relevance: 0 Score: 0.375

$$\begin{aligned}\langle 3u + 2v,\ -u + 4v\rangle &= -3\langle u, u\rangle + 12\langle u, v\rangle - 2\langle v, u\rangle + 8\langle v, v\rangle \\ &= -3 \cdot 9 + 12 \cdot (-4) - 2 \cdot (-4) + 8 \cdot 25\end{aligned}$$

since $\|u\| = 3$, $\|v\| = 5$, and $\langle u, v\rangle = -4$.

Figure C.3: Top-3 retrieval results from MathAMR (baseline) for Task 1 topic A.301.

Observation. The baseline retrieves only a tangentially related result at rank 1 (an inner-product inequality, relevance 1) and two completely unrelated results at ranks 2 and 3 (both relevance 0). Surface-level similarity of inequality notation appears to trigger matches without capturing the matrix norm context.

C.2 Task 2: Topic B.301

Topic B.301 is a formula retrieval query derived from post A.301. The query formula is $\|A\|_2 = \sqrt{\rho(A^T A)}$, which appears in the post as the formula the questioner explicitly wants to avoid using in their proof.

Topic B.301 — Query formula from post A.301

Post title: Inequality between norm 1, norm 2 and norm ∞ of Matrices

Query formula (id: q_6):

$$\|A\|_2 = \sqrt{\rho(A^T A)}$$

Context: The formula appears in the post as the definition the questioner explicitly excludes from the proof, making it the central mathematical object of the discussion.

Figure C.4: ARQMath-3 Task 2 topic B.301.

C.2.1 MathAMR+ Results

MathAMR+ achieves NDCG′=0.7104, MAP′=0.4256, P′@10=0.5000, BPref=1.0000 on this topic.

Rank 1 Relevance: 3 Score: 1.000

$$\|A\|_2 = \sqrt{\rho(A^T A)}$$

Rank 2 Relevance: 3 Score: 0.993

$$\|A\|_2 = \sqrt{\rho(A^T A)} \leq \sqrt{\|A^T A\|_\infty}$$

Rank 3 Relevance: 3 Score: 0.986

$$\|A\|_2 = \sqrt{\rho(A^T A)} = \rho\left(\begin{bmatrix} 0 & A \\ A^T & 0 \end{bmatrix}\right)$$

Figure C.5: Top-3 retrieval results from MathAMR+ for Task 2 topic B.301.

Observation. All three top results from MathAMR+ are direct structural matches of the query formula (relevance 3). The system correctly identifies formulas that are either identical to or are algebraic extensions of $\|A\|_2 = \sqrt{\rho(A^T A)}$, demonstrating precise formula-level matching.

C.2.2 MathAMR (Baseline) Results

The original MathAMR approach achieves NDCG′=0.0000, MAP′=0.0000, P′@10=0.0000, BPref=0.0000 on this topic.

Rank 1 Relevance: 0 Score: 0.784

$$\varphi_a(z) = \frac{z - a}{1 - \overline{a}\, z}, \quad a \in \mathbb{D}$$

Rank 2 Relevance: 0 Score: 0.779

$$f(z) = e^{i\theta} \prod_{j=1}^{k} \frac{z - a_j}{1 - \overline{a}_j\, z}$$

Rank 3 Relevance: 0 Score: 0.778

$$\prod_{k=1}^{n-1} \sin\left(\frac{k\pi}{n}\right) = \frac{2n}{2^n}$$

Figure C.6: Top-3 retrieval results from MathAMR (baseline) for Task 2 topic B.301.

Observation. The baseline retrieves three completely unrelated formulas (all relevance 0), drawn from complex analysis and trigonometry. None share structural or semantic similarity to the query, yielding scores of zero across all metrics. MathAMR+ achieves perfect BPref on this topic, recovering all judged relevant formulas in the top of the ranked list.

Appendix D

Generative AI Usage Disclosure

Large language models (LLMs) were used in several supporting roles throughout this work. For software development, LLMs were consulted to generate code snippets for data preprocessing, model training utilities, data analysis, and visualization prototyping. All generated code was reviewed, modified, and validated through running experiments, comparing outputs against expected values, and visualizing results. Any bugs or unexpected behaviors were resolved manually.

LLMs were also used to assist with note-taking during the review of research papers, informing the author's independent understanding. All analysis and synthesis in this thesis are the author's original work. In later stages of writing, LLMs were used to suggest alternative phrasings and identify grammatical issues. All passages were initially drafted by the author and suggestions were selectively incorporated after review. LLMs were also used to generate preliminary LaTeX figures, which were substantially refined manually prior to inclusion. No AI-generated content was included without independent review and modification.