Enhancing LLMs in Answering Math Questions Based on Unification and Vector Embedding

Connor Cihula
Computer Science Department
Brigham Young University
Provo, Utah, USA
ccihula@student.byu.edu

Yiu-Kai Ng
Computer Science Department
Brigham Young University
Provo, Utah, USA
ng@compsci.byu.edu

Abstract—**The diverse nature of math makes it uniquely relevant to a wide variety of professional fields and facets of daily life. Math is applied in manufacturing cars and designing phones, and it is used in cooking recipes and calculating property taxes. The immense relevance of math combined with the ever-increasing reliance on digital platforms for problem-solving has created the demand for math question-answering (QA) platforms, such as Math Stack Exchange (MSE), which are popular due to the ease at which they can be accessed from online. Unfortunately, one of the major drawbacks of these platforms is the fact that users are sometimes forced to wait for an unspecified amount of time before receiving an answer to their question. With the recent widespread adoption of Large Language Models (LLM)s, instead of requiring users to wait for a human to provide an answer, users create prompts on an LLM for answers. However, accuracy of these generated answers leaves much room for improvement. In this paper, we present a similarity search tool that matches existing math questions and answers based off of the similarity of the text and equation portions of the content. This tool is used to improve the accuracy of the answers that LLMs give when prompted with math questions found on MSE. This research presents new ideas for the Math QA community to consider, concluding that the proposed similarity search tool improves the math question-answering capability of LLMs.**

Index Terms—**Math QA, LLM, formulas, textual content**

I. Introduction

Math plays a critical role in a wide variety of disciplines, including engineering, economics, computer science, biology, etc. It is the foundation for modern advancements in medicine, transportation, entertainment, and many other fields. Outside of a professional setting, math is also an integral part of every-day life, appearing in ordinary settings such as finance management, shopping, cooking, scheduling, etc. The relevance of math extends far into daily life, yet many perceive math as an inherently difficult topic [5], leading to reluctance to engage with it [18]. This is especially problematic as the demand for technical literacy increases [14]. In fact, math affects individuals even outside of a career, as a lack of math skills increases the likelihood of making illogical and poor decisions [8].

The reluctance to engage with math does not reflect the importance of math, but highlights the need to make math learning more accessible. If learning math can be perceived as a less daunting task, then unwillingness to engage with it will diminish. Learning math can be made less intimidating by making math learning more accessible. In a world heavily reliant on the internet, one of the simplest and most accessible ways to learn is through using technology, and therefore online Math QA systems have become an answer to the problem over the past couple decades as a way to conveniently get answers to math questions [17]. Math QA systems allow users to ask questions and receives answer to the questions. Some Math QA systems have a very large user base of people that are knowledgeable about math who are willing to play the role of answer provider as a hobby, all the way to people who professionally practice math at a high level. These systems provide a convenient way of connecting those who have math questions with those who are able to answer math questions, for a wide range of math topics and levels of difficulty.

While existing Math QA systems promote learning, a major issue is that users sometimes wait indefinitely for answers, undermining the convenience of digital access. Moreover, answers to questions are time-sensitive due to work deadlines or school due dates. And quite often a user simply does not want to wait for an unknown amount of time, as they might lose interest or become frustrated with the inability to find an answer in a timely manner. Math QA systems attempt to handle this issue by recommending to the question-asker existing similar questions that have already been answered, but this process is heavily reliant upon the quality of the recommendations. Poorly matched recommendations can discourage users, leaving them dissatisfied and potentially harming their desire to learn math. Improving the accuracy of these recommendations is thus a critical step towards enhancing the user experience on Math QA platforms, and by extension a critical step towards improving math education. To address this problem and improve the accuracy of Math QA systems, we propose to develop a Math QA system using *unification*, a technique from formal proofs, to match existing answered questions using equation similarity. In addition, we use vector embeddings to measure the similarity of mathematical text, providing an additional dimension for matching math questions with answers. Our algorithm processes MathML-formatted equations by arranging their elements into arrays and computes a similarity score based on element matches and sequence alignment.

A few studies have addressed Math QA through answer-matching [18] in the past, but none have taken the approach described above; this research evaluates its novelty and viability. Our method is also notable for its speed and simplicity that matches questions with answers by unifying equations and aligning textual content. The retrieved answers can then be used to fine-tune an LLM, improving its performance on Math QA. Fine-tuning an LLM with relevant math questions and answers enhances its performance by aligning the model's parameters more closely with mathematical reasoning patterns. Unlike general pretraining, which covers broad language use, fine-tuning on domain-specific data helps the model better understand mathematical terminology, problem structures, and solution strategies. This focused exposure improves its accuracy in solving problems, interpreting symbolic expressions, and generating step-by-step explanations. Additionally, it reduces hallucinations and enhances consistency, especially in tasks requiring logical rigor and precise calculations. Moreover, the model becomes more reliable and effective in educational, research, and applied math contexts.

II. Related Work

The recognition of room for improvement in the accuracy of Math QA systems has driven researchers to devise methods for more effective consideration of formulas, equations, symbols, and other math notations that traditional IR systems struggle to handle [6]. Tangent-S [3] uses symbol layout trees to represent equations as tree structures, explicitly capturing the relationship between operands and operators, as well as avoiding conversion errors. Variable Typing [20], which links math symbols with their natural language context and creates a typed formula index, enhances the training and accuracy of Math QA models compared to those without typed formula indexing. ARQMath [22] and ARQMath-2 [16] address Math QA with both text and equations, retrieving relevant answers by evaluating the similarity between equations and the text surrounding them. Our approach, which considers math equations represented in hierarchical MathML trees, introduces the novel combination of a depth first search, unification to match the math symbols in the equation, and a scoring system that considers both the complexity difference between math equations and vector representation of math textual contents.

Recognizing the performance gap between natural language tasks and Math QA has also driven researchers into improving the technical and mathematical capabilities of LLMs. Minerva [10], which trains the PathWays Language Model (PaLM) on a large swath of scientific and mathematical content, uses few-shot prompting on LLMs to advance the task of answering math questions. Employing Sampling Fine-Tuning (RFT) [21] improves the mathematical capabilities of LLMs by training LLMs on automatically augmented math data and fine-tunes LLMs to increase the performance for mathematical tasks. Reasoning with Reinforced Fine-Tuning (ReFT) [13] first applies reinforced fine-tuning to warm up an LLM on chain of thought data and then uses proximal policy automation reinforcement learning to further train LLMs. Retrieval-augmented Generation (RAG) [9], which focuses on improving LLMs

TABLE I: List of all MathML tags

Tags	Interpretation
<math>	Top level element, indicating the start of a formula
<mfrac>	Branches into two children, numerator & denominator
<mrow>	Groups items together, such as parentheses
<msqrt>	Square root
<mroot>	Branches into two children, the root and its degree
<msub>	Branches into two children, the base and its subscript
<msup>	Same as above except second child is superscript
<msubsup>	Branches into three children, the base, its subscript, and its superscript
<munder>	Branches into two children, the base and under
<mover>	Branches into two children, the base and over
<munderover>	Branches into three children, the base, under, and over
<mn>	Wraps a single literal (decimal or int)
<mi>	Wraps a single "identifier", i.e., variable, function (cos, sin, etc.), symbol (Π, θ, etc.))
<mo>	Wraps any single operator (+, -, etc.)
<mtext>	Wraps text (if, maps to, etc.)

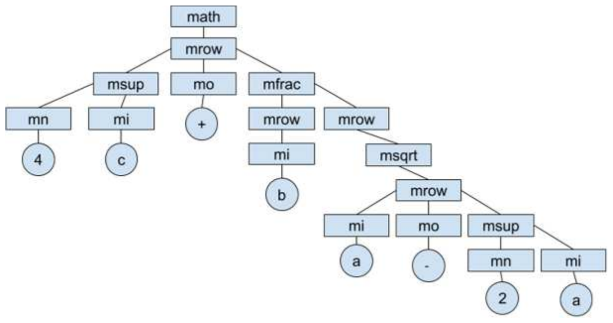

Fig. 1: Tree representation of the equation $\frac{\sqrt{a^2-a}}{b} + c^4$

math reasoning capabilities in a primary educational setting, reduces hallucinations and improve the relevancy of generated answers to math questions. Instead, we introduce a similarity search tool built with unification and sentence transformers to improve the math capabilities of lightweight LLMs.

III. Our Math QA Model

Our Math QA model determines the similarity of math equations through matching of operators and operands, which is a crucial component of unification, and math content. First, similar math operators and operands specified in a question are matched with others in answers. The matching is done by using *unification*, an algorithmic process frequently used for theorem proofs and logic programming. Unification, in this context, determines the solution for a system of two equations.

To make unification applicable to Math QA, we have designed a unification algorithm that is based on math equations specified in $MathML$. In MathML, math equations are represented in XML, using tags such as <mo> for operators, <mi> for variables, <mn> for literals (see Table I for a list of all MathML tags), where the relationships among the tags are represented in a hierarchical manner, suitable for an XML structured language (see Figure 1, where a math equation is represented in standard math notation along with its respective MathML tree representation).

Our unification approach is made up of two parts: a traversal of the MathML tree representing the math equations being compared that yields two arrays of arrays, and a scoring scheme that compares the two arrays of arrays. The two arrays of arrays represent the math equations being compared (see Section III-A for details), and they are in array format for ease of comparison, since it is simpler to compare strings in an array than it is to compare elements of a MathML file parsed into Python Elements using Python's ElementTree module. The comparison determines how similar they are to each other, and therefore how similar the formulas in questions and answers in comparing with each other. The final output is a value between 0-1, where 1 represents two equations being identical, and 0 represents two equations being completely dissimilar.

A. MathML Tree Traversal

Our Math QA algorithm first applies a *depth-first search* on the two equations being compared using the ElementTree module built into Python, generating two arrays of arrays. The arrays of arrays represent the branches of the equations in (XML) tree format. One array of arrays is derived from the equation in the user's query, referred QT_Tree, whereas another array of arrays is derived from the equation in the answer to be matched with the user's math question, referred Ans_Tree. The subarrays represent the path to a specific leaf node, starting with the 'math' tag, the root of all equations.

The algorithm injects numbers into the arrays of arrays that represent the various branches of a tag. The purpose is to ensure that when matching is performed, the differences between the denominator and numerator, base and exponent, base and subscript, base and over-symbol, and base and under-symbol are accounted for. While the difference between the branches of these tags must be accounted for when matching, the difference between what the tags <mi> (representing variables) and <mo> (representing constants) represent should *not* be considered, since their matching are irrelevant.

B. Computing the Similarity Scores among Equations

We compute the score that represents the degree to which the branches of any two equations match, and by extension, the degree to which the two equations being compared match. The score is calculated based on two sub-scores: the *depth score* and *complexity score*.

1) The Depth Score: Depth score is calculated by comparing each branch in the QT_Tree to each branch in the Ans_Tree to determine which branch in the Ans_Tree is most similar to each branch in the QT_Tree. The depth score calculation utilizes dynamic programming to find the *longest contiguous subsequence* of MathML tags found in both the Ans_Tree branch and QT_Tree branch. In other words, the depth score for each branch is measured by determining the number of tags both branches have in common, in a row. The calculation of the depth score does not consider the MathML <math> tag found at the root of every branch, which serves the purpose of allowing MathML to embed directly in XML, to ensure a more accurate score.

After the depth score is calculated, a *weight* is applied to enhance the accuracy of the depth score to better reflect how *similar* the branches being compared are. As the algorithm determines the depth score for each branch, it also keeps track of *weighted tags* found in each branch. Weighted tags are tags that need special consideration when comparing branches because they frequently represent large *structural differences* in the equations. This means that a weighted tag has a greater effect on the structure of an equation compared to an *unweighted tag*. The presence of a weighted tag in one branch but not the other means that the two branches have a large structural difference, and so the weights of the weighted tags are applied to the depth score, lowering the score based on how much the weighted tags affect the structure of that branch of the equation. With that in mind, we have assigned weight to tags based on the *geometric sequence* so that <mfrac>, <msup>, <msubsup>, and <msqrt>, which are used for annotations of fractions, superscript, superscript and subscript, and square roots, respectively and are assigned the weight of 1.0. The weights of the tags <mover>, <munder>, and <munderover>, which represent accents, limits, and combined notation, are given the weight of 0.5, respectively. The remaining tags as shown in Table I are set aside the weight of 0.25, since they do not greatly affect the structure of an equation. The final depth score of a tree, i.e., its corresponding equation, is calculated by taking the average of the depth scores of all its branches, which falls between 0 and 1.

2) The Complexity Score: The complexity score accounts for the case where one equation is significantly more *complex* than the other, which is determined by comparing the number of tags found in the Ans_Tree and QT_Tree. Since the depth score allows multiple branches in the QT_Tree to match the same branch in the Ans_Tree, relying solely on the depth score can result in an inflated total score that does not accurately reflect the (dis)similarity between the two corresponding equations, and that problem is handled with the inclusion of the complexity score. Equation 1 formally defines the complexity score of two trees, in which ATT and QTT denote the *number* of MathML tags in the Ans_Tree and QT_Tree, respectively.

$$Complexity\_Score = \frac{min(ATT, QTT)}{max(ATT, QTT)} \tag{1}$$

The *complexity score* achieves great efficiency without unnecessary complexity, reinforcing the proposed Math QA system that focuses on speed and simplicity. While the *depth score* accounts for what are in common between two equations, the *complexity score* determines what are the difference in the two equations. Considering both scores yields an accurate total score than if only one or the other is considered.

3) The Equation Similarity Score: The *equation similarity score*, denoted EQS_Score, of any two equations considers both the *depth score* and *complexity score* of the equations. As shown in Equation 2, since the *complexity score* is multiplied by the *depth score* of branches, the *higher* the two scores, the *higher* the *similarity score* is. In Equation 2, the EQS_Score computes the degree of similarity between Ans_Tree that represents the equation specified in a math answer and QT_Tree

that represents the equation in a math question. Furthermore, n denotes the number of *branches* in Ans_Tree.

$$EQS\_Score = Complexity\_Score \times \sum_{i=1}^{n} \frac{Depth\_Score_i}{n} \tag{2}$$

C. The Context Similarity Score

Besides computing the equation similarity score of equations specified in a math question A and its potential answer Q, our Math QA model also consider the context similarity between A and Q. Our Math QA model applies the all-MiniLM-L6-v2 sentences transformer with Facebook AI Similarity Search (FAISS) to quickly and efficiently compare and determine the similar content of A and Q based on the large corpus of Math Stack Exchange (MSE) corpus[1]. The complete code for the equation and context similarity search tools are available at https://github.com/ElemehnoP/CS497R.

1) The all-MiniLM-L6-v2 Sentence Transformer: The all-MiniLM-L6-v2 sentence transformer is used to generate vector representations of text. This transformer model is pre-trained on a large corpus of text so that it learns to understand context, syntax, and other semantics. It then uses this knowledge to convert new text into dense vector embedding, where each piece of text is represented as a vector in a high-dimensional space, allowing for retention of the semantic meaning of the text. This process ensures that semantically similar content lies in close proximity, which is measured by cosine similarity. The close proximity in a high-dimension space is represented by the direction of the vector. Two similar texts will have vectors that have a similar direction, so the similarity of two texts can be represented by the angle between the two vectors. (See Table II for an example of the text comparison process.)

TABLE II: Example of sentence comparison using the all-MiniLM-L6-v2 sentence transformer

Sentence #	**Sentence Content**	**Vector Embedding**	**Cosine Sim. Compared w/ Sentence #1**
1	The quick brown fox jumps over the lazy dog	[0.044 0.059 0.0482 0.078 0.0267 -0.0376 -0.003 -0.059 ...]	1.0
2	The fast dark-colored fox leaps over the sleepy canine	[0.049 0.005 0.006 0.138 0.034 -0.014 -0.002 -0.046 ...]	0.809
3	The health of the economy depends on many factors	[0.078 -0.004 -0.007 0.005 0.034 0.025 -0.005 0.001...]	0.005

The ability of the sentence transformer is also surprisingly effective for comparing equations embedded in textual contents of math questions and answers represented in MathML. Much like semantic text, MathML is structured so that items' meanings are based on the context they are used in. By treating MathML equations as plain text strings, rather than as XML, the transformer easily processes information like it would on textual text, with minimal meaning loss. Moreover, the transformer's small size and efficient inference capabilities make it well-suited for processing large volumes of MSE content without incurring excessive computational demands.

[1]MSE corpus refers to the collection of questions, answers, and discussions on the Mathematics Stack Exchange website (math.stackexchange.com). It's a vast repository of knowledge in math, curated by a community of experts and enthusiasts.

2) FAISS: After the text and embedded equations are transformed into embedding vectors using the all-MiniLM-L6-v2 tool, the results are fed into FAISS to generate an index of math questions and answers. FAISS is a library that enables efficient similarity search by looking up information stored as vectors in an index. FAISS is particularly well-suited for our Math QA model due to its ability to handle large datasets and perform nearest neighbor searches with minimal computational overhead while retaining high accuracy. This is accomplished by using techniques such as product quantization and optimized clustering, which reduce dimensionality while preserving approximate similarity relationships.

In order to most effectively take advantage of the speed offered by the FAISS index that is populated with the output of MSE information fed through the all-MiniLM-L6-v2 sentence transformer, the tool will first generate an index of MSE questions, rather than MSE answers. This is because each math question has anywhere from 5-30 corresponding answers, and despite FAISS's efficient similarity search, indexing and searching through all answers would be computationally expensive and reduce performance. So instead, the MSE question that is selected as the query is instead compared to all the MSE questions in the FAISS index, and the top 5 most similar questions are retrieved. Then, a second FAISS index is created and populated with the answers of the 5 most similar questions that were retrieved. This keeps the FAISS indexes large enough to make use of FAISS's efficiency, but small enough to remain within manageable computational bounds. The query is then compared with the compiled math answers, and the top 5 most similar math answers are returned. Similarity is determined by computing the distances between both semantic text embedding and equation embedding.

The *EQS_Score* and *Context Similarity score* of a math question A and its potential answer Q are combined using the *Borda Count strategy* [4] to yield the final score of A and Q. The Borda Count, which is a rank-based voting method used to aggregate preferences from multiple sources, i.e., the equation and context similarity scores in our case, balances out extreme opinions and reflects the overall preference distribution. The Borda Count is applied to other potential answers to A and the end result is a list of ranked answers to A.

D. Advancing Math QA Systems

The novelty of the proposed Math QA model lies in the usages of (i) MathML tree traversal to determine the similarity of two math equations and (ii) the all-MiniLM-L6-v2 sentence transformer and FAISS to compute degree of resemblance of textual contents of math questions and answers. While the sentence transformer is traditionally used for natural language processing tasks, it is also suitable for capturing important relationships between math symbols represented as text. Combining the usage of FAISS and sentence transformers allows for the comparison between equations and also natural language, which is present in nearly every math question and answer. Our math question and answer comparison approach is very accurate, and the novelty of the proposed Math QA model has been proven with extensive testing, in addition to its incorporation with LLMs (see next section for details), a technology relevant in each professional field, especially in math.

E. Enhancing LLM Precision Using Our Math QA Model

In this section, we present our strategy to enhancing the math capabilities of LLMs through n-shot and fine-tuning training LLMs with math questions and answers retrieved by our Math QA model. In this regard, we propose a novel method to enhance LLMs' ability to answer math questions by jointly considering math equations and textual similarity using our Math QA model.

1) Fine-Tuning LLMs: Fine-tuning an LLM refers to the process of taking a pre-trained language model and adapting it to perform better on a specific task or domain by further training it on a smaller, task-specific dataset. This is different from training a model from scratch, as the model already possesses a general understanding of language. Fine-tuning simply adjusts its parameters to improve performance on a specialized task. This approach not only boosts task-specific performance but also avoids the enormous computational cost of full model training, making it far more practical and accessible. In our case, the goal of fine-tuning an LLM is to enhance performance on mathematical question answering, particularly within the domain of MSE.

2) LLMs to be Fine-Tuned: To demonstrate the improved performance of LLMs fine-tuned with math questions and answers retrieved by our Math QA model, we selected three language models: LLaMA3.1-8B, Qwen2.5-7B, and Gemma3-4B. These models were chosen primarily for their reasonable size, which allows for experimentation without requiring extensive computational resources. While larger models may offer superior performance in some scenarios, the selected models provide a good balance between capability, efficiency, and hardware requirements. A brief introduction to each LLM is provided below to highlight their novelty and merits.

- **LLaMA 3.1-8B**. LLaMA is developed by Meta. The 3.1-8B version, which is an improved variant in the LLaMA 3 series with 8 billion parameters, is designed to offer strong performance while remaining computationally efficient. It is suitable for fine-tuning on domain-specific tasks or running on mid-scale infrastructure for language generation, QA, and summarization tasks.
- **Qwen 2.5-7B**. Qwen is Alibaba's open-source LLM series. The 2.5-7B version contains 7 billion parameters and represents the latest improvements in Qwen's architecture and training methodology. It is effective in both research and industry settings, particularly for multilingual applications and alignment with downstream tasks.
- **Gemma 3-4B**. Gemma is an open-weight LLM family developed by Google DeepMind. The 3-4B version refers to a 4 billion parameter model optimized for efficient inference and fine-tuning. The model is ideal for developers looking to integrate LLMs into real-world applications while maintaining efficiency and control.

3) The Fine-Tuning Goal and Anticipated Results: The novelty of using our Math QA model for fine-tuning LLMs is proven by comparing the baseline performance of the LLMs to the fine-tuned but otherwise identical LLMs.

Each LLM is first queried with a subset of MSE questions. This establishes a baseline performance on mathematical queries without any fine-tuning. Hereafter, the prompt is modified to include both questions and answers retrieved by our Math QA model from a different subset of MSE posts. Although the initial MSE question subset and the tool-based fine-tuning subset are distinct, augmenting the prompt with retrieved, contextually relevant content results in measurable improvements in answer quality and accuracy. The fine-tuning demonstrates that even when different data subsets are used, targeted and contextually relevant information enhances LLM performance. When the same model is evaluated under the same conditions, with the only difference being the fine-tuning process, the resulting performance gains confirm the effectiveness of our retrieval-based fine-tuning strategy.

IV. Experimental Results

In this section, we evaluate the performance of our model.

A. Datasets

For evaluating the performance of a proposed Math QA model that processes text and equations in math, we have created a custom dataset that is extracted as a subset of the Math Stack Exchange (MSE) dataset. This text-equation dataset contains 64,860 questions with their respective list of answers, scores, acceptance marking, and image versions of each question and answer generated from the stored text with embedded LaTeX math markup. In this dataset there are 730,941 answers in total, with 10.8 answers per question on average. The text-equation dataset has been split by 90% for training, 5% for testing, and 5% for validation. The purpose of the training portion of the text-equation dataset is to train the overarching Math QA model. This should enable inputting math questions using textual, mathematical, and visual aspects and be able to return answers which are also in these formats, enabling more comprehensive and flexible answering of math questions with greater cross modality representation and understanding for learners.

B. Existing Math QA Models Used for Comparisons

To evaluate the performance of our Math QA model, we compare the proposed model with four other well-established Math QA systems in the field in terms of their reputation in answering math questions.

- **MaRec** [6]. The Math QA system ranks answers by combining textual and formulaic relevance features. Text similarity is measured via $KL\text{-}divergence$ [15], while formula similarity is computed by matching tree-structured XML representations. Their combined score determines the most relevant answers.

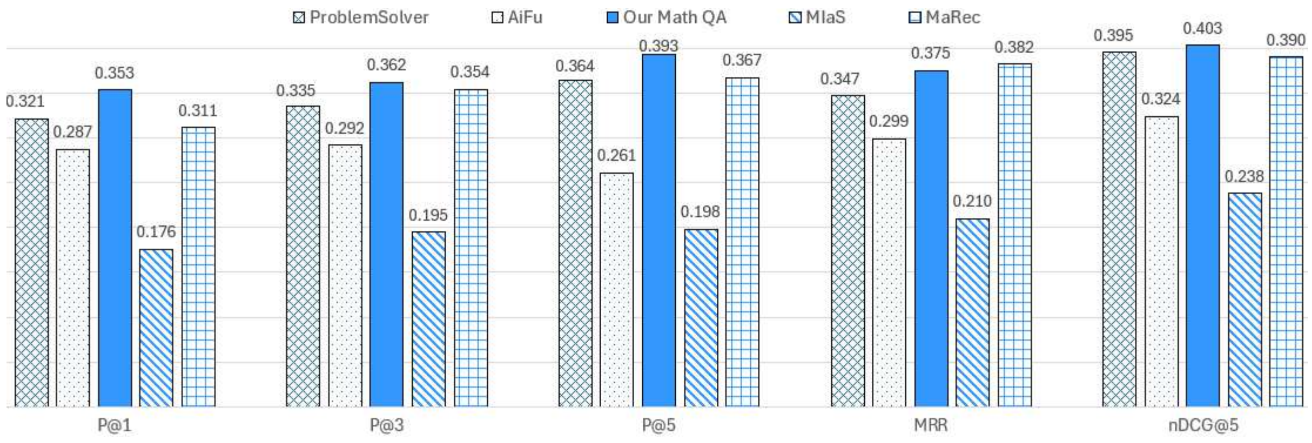

Fig. 2: The performance evaluation metrics of various Math QA systems, including our proposed Math QA model

- **ProblemSolver**. ProblemSolver [12] combines a sequence-to-sequence neural model for question-answer matching and an arithmetic tree approach for ranking math answers. Trained on over 600K questions, the system delivers high performance across diverse question types using augmented data (i.e., variations of existing datasets) and tree matching for improved accuracy.
- **The AiFu system**. Using a semantic parser, AiFu [11] transforms math questions into logical forms that is processed by a symbolic solver. It incorporates a neural network-based model trained to answer multiple-choice questions based on question word sequences. AiFu contains two subsystems, AiFu 1 and AiFu 2.
- **MIaS** [19]. Math-Aware Information Retrieval System) is a Math QA system designed to retrieve relevant documents and answers by indexing both textual content and mathematical expressions. It converts math formulas into a canonical form to support structure-aware matching and leverages specialized indexing techniques to handle math notation alongside natural language.

Users of a ranking system tend to look at only the top few ranked results to find relevant suggestions. Some search tasks have only the top-ranked suggestion, i.e., Precision at rank 1 ($P@1$), in mind, whereas others might consider the top-3 or top-5, i.e., Precision at rank 3 or rank 5 ($P@3$ or $P@5$), ranked suggestions. We computed the $P@1$, $P@5$, MRR, and $NDCG_5$ values, which are widely-used measures in information retrieval, for the five Math QA systems, i.e., our proposed Math QA, MaRec, ProblemSolver, AiFu, and MIaS, involved in our empirical study. Figure 2 shows the performance measures for the *average precision* at rank 1 (i.e., average $P@1$) and at rank 5 (i.e., average P@5), in addition to the average of the reciprocal ranks at which the *first* useful suggestion (among all the top-5 ranked suggestions) for each test question is made, i.e., MRR, and the $NDCG_5$ values. All the performance metric scores of our Math QA model are higher than the corresponding scores of MaRec, ProblemSolver, MIaS and AiFu, and the results are statistically significant based on the Wilcoxon Signed-Rank test ($p <$ 0.045).

C. Performance Evaluation Using Math QA-LLM Pipeline

Based on the performance metrics depicted in Table III, we answer the research questions, RQ_1, RQ_2, and RQ_3, below.

- RQ_1. *What improvements are seen when comparing the fine-tuned LLMs to the ones without?* The results (see Table III) indicate that the fine-tuning process generally enhances the mathematical question answering capabilities of the LLMs, though the degree of improvement varies. Qwen 2.5-7B demonstrates the most significant performance boost. While fine-tuning has slightly worse RR and RR@5 values, all other metrics show improvement, most notably P@3, which increases from 0.833 to 0.967. LLaMA 3.1-8B shows more moderate improvements after fine-tuning. Every metric improves after fine-tuning except for P@3, which remains unchanged. While the improvements vary from category to category, overall, the improvements seen in Qwen 2.5-7B and LLaMA 3.1-8B suggest that the similarity search tool is effective for fine-tuning LLMs using the Math QA model to achieve better math answer relevance.
- RQ_2. *What are the limitations of using the proposed Math QA tool to fine-tune LLMs to improve relevant answer generation for math questions?* In sharp contrast to Qwen 2.5-7B and LLaMA 3.1-8B, Gemma 3-4B's results show that it performs better without fine-tuning than with it. P@5 (0.640 $\rightarrow$ 0.500), AP (0.681 $\rightarrow$ 0.612), RR (0.817 $\rightarrow$ 0.650), and nDCG (0.777 $\rightarrow$ 0.656) all decrease after fine-tuning, with only P@3 (0.600 $\rightarrow$ 0.633) showing a modest improvement. A possible explanation for these results is perhaps the proposed Math QA model is effective at improving larger LLMs (7B+) but not small LLMs, as Gemma 3-4B (4B).
- RQ_3. *What do the observed patterns of performance gains and regression reveal about the robustness of our Math QA-LLM pipeline model and future optimization using the proposed Math QA model?* These observations suggest that, while the retrieval-based fine-tuning strategy is effective for certain models, further investigation is needed to understand its limitations and to optimize it for models like Gemma 3-4B. Factors such as model size, internal architecture, training strategies, etc. are all worth considering.

TABLE III: LLM performance metrics among various LLMs, including our pipeline model

Model	Fine-Tuned	P@1	P@3	P@5	AP	RR	RR@5	nDCG	nDCG@5
LLaMA 3.1-8B	No	0.675	0.633	0.560	0.611	0.683	0.683	0.695	0.707
	Yes	**0.687**	0.633	**0.580**	**0.689**	**0.750**	**0.750**	**0.774**	**0.785**
Qwen 2.5-7B	No	0.721	0.833	0.780	0.868	1.000	1.000	0.950	0.975
	Yes	**0.782**	**0.967**	**0.900**	**0.943**	0.980	0.980	**0.972**	**0.976**
Gemma 3-4B	No	0.632	0.600	0.640	0.681	0.817	0.800	0.777	0.757
	Yes	0.631	**0.633**	0.500	0.612	0.650	0.650	0.656	0.668

D. Performance Evaluation Using Multiple Datasets

To further verify the novelty of the proposed Math QA-LLM pipeline model, we assess the model using the following datasets for solving math tasks against LLaMA 3.1-8B, Qwen 2.5-7B, and Gemma 3-4B:

(a) MATH [7]: Math contains 12,500 challenging high school competition math problems. Each problem in MATH has a full step-by-step solution which can be used to teach models to generate answer derivations and explanations.
(b) GSM8k [2]: A dataset of 8.5K high quality linguistically diverse grade school math word problems. This dataset is designed to have high linguistic diversity while relying on relatively simple grade school math concepts.
(c) SAT [1]: SAT consists of 32 math questions without figures from the May 2023 College Board SAT examination.
(d) MMLU-STEM [7]: This dataset contains 3.15K questions ranged in difficulty from an elementary level to an advanced professional level. It tests both world knowledge and problem-solving ability. Its STEM subjects includes physics, computer science, mathematics, and others, and require knowledge of empirical methods, fluid intelligence, and procedural knowledge.

For our pipeline model and each of the three baseline models we run them on unique but similarly formatted math benchmark datasets. Each of the four datasets listed above cover different tasks, e.g., math competition, word problems, SAT exam, STEM knowledge, etc., but are formatted as a list of textual QA pairs, with user formatted questions paired with answers that have been validated by users as being valid, accurate, and ideal answers to the questions they were intended to solve. Both the questions and provided correct answers contain natural language and math content in the form of text, and are 1-to-1 pairs, meaning that there is only one answer given per math question. Answers typically start out with explanation or steps taken to work through the given problems, but always end with a clearly marked answer in math notation to the question at the end of a given answer, and exact matching is used to verify that the LLM generated answer is correct in comparison to the dataset's final answer for the question. An example of a math question and answer from the MATH dataset is shown below.

The comparative performance of the different Math LLM models on the benchmark datasets is shown in Figure 3. Overall, the comparison clearly shows that our pipeline model outperforms other baseline Math LLMs on various datasets, and the results are statistically significant ($p < 0.035$).

E. Question Processing Time

One of the design goals of our proposed Math QA-LLM pipeline model is to process user questions with processing time compatible with existing Web search engines. With that in mind, we have conducted a performance evaluation to compute the average question processing time of the proposed model and other baseline Math QA models, i.e., MaRec, PowerSolver, MIaS, and AiFu, along with ChatGPT, an LLM, for retrieving and ranking the answers to each math question in the MSE dataset. The question processing time is evaluated on the same laptop PC. The PC is a Macbook Pro with M1 chip, 8-core CPU with 4 performance cores and 4 efficiency cores, 8-core GPU, and 16-core Neural Engine. As shown in Table IV, the proposed pipeline model runs twice as far as ChatGPT, even though it is comparatively slower than other baseline Math QA models in terms of the question processing time. The results supports our claim that our pipeline model is an enhancement of state-of-the-art LLMs in handing Math QA.

Note that AiFu is relatively slower compared with other Math QA models because it employs two subsystems: AiFu 1 and AiFu 2, which are combined to generate the final ranking result for a math question. Furthermore, ChatGPT is the slowest since it takes longer time to produce a new answer to a question, while other models, with the exception of our proposed pipeline model, use existing answers to return.

V. Conclusion

Math is a very diverse and important field of study. It is relevant to daily life directly and indirectly, as it is used in designing computers and developing new medicine, as well as shopping and finance management. Better math skills lead to better decision making capabilities; however, math learning is inhibited by the fact that there is a general reluctance to engage with it. Making math learning more accessible is crucial to solving this problem, and Math QA platforms address this need. However, the inconvenience of not receiving a timely answer or receiving non-relevant answer to a question can be a roadblock to math learning. People with math questions attempt to mitigate this issue by turning to LLMs for math answers, but the effectiveness of LLMs for answering math questions is heavily dependent upon the quality of the generated answers. Inaccurate answers can discourage those with questions, and so we propose a Math QA-LLM pipeline model that uses MathML tree matching, sentence transformers, FAISS, and fine-tuning to improve the quality of generated answers. The proposed pipeline model is notable for its unique design and its effectiveness on LLMs with less than 10 billion parameters. Conducted empirical analysis has proven the novelty and the effectiveness of the proposed pipeline system.

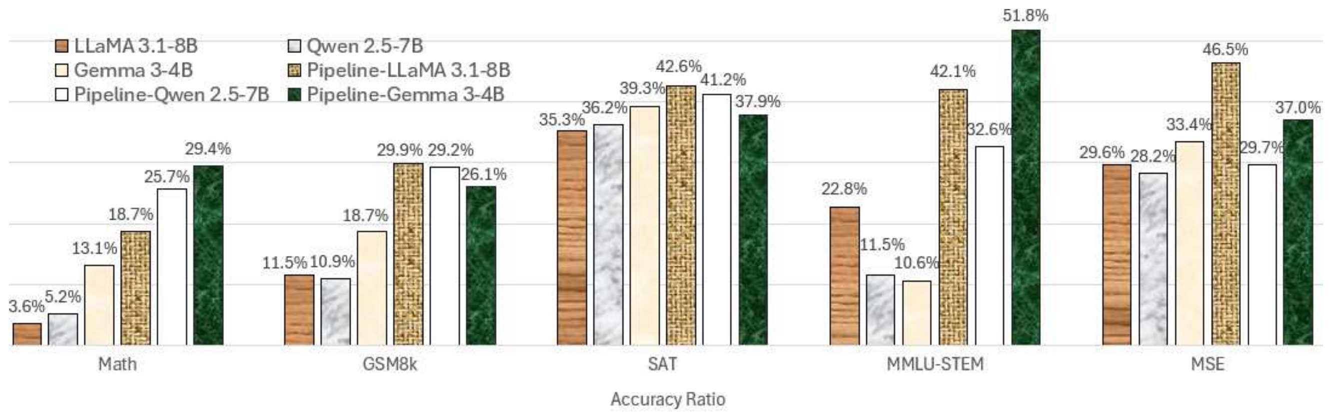

Fig. 3: The accuracy ratios of various LLMs on four baseline math task datasets measured by percentage from 0-100%

TABLE IV: Question processing time (in seconds) between $MaRec$, $ProblemSolver$, $MIaS$, $AiFu$, the proposed pipeline model, and $ChatGPT$

Models/Processing Time	MaRec	Problem Solver	MIaS	AiFu	Our Pipeline Model	ChatGPT
	0.037s	0.229s	2.05s	4.000s	7.912s	15.042s

References

[1] Z. Azerbayev et al. Llemma: An Open Language Model for Mathematics. *arXiv preprint arXiv:2310.10631*, 2023.

[2] K. Cobbe et al. Training Verifiers to Solve Math Word Problems. *arXiv preprint arXiv:2110.14168*, 2021.

[3] K. Davila and R. Zanibbi. Layout and Semantics: Combining Representations for Mathematical Formula Search. In *Proc. of ACM SIGIR*, pages 1165–1168, 2017.

[4] J. Fraenkel and B. Grofman. The Borda Count and Its Real-World Alternatives: Comparing Scoring Rules in Nauru and Slovenia. *Australian Journal of Political Science*, 49(2):186–205, 2014.

[5] A. Fritz, V. Haase, and P. Rasanen. International Handbook of Mathematical Learning Difficulties. *Cham, Switzerland: Springer*, 2019.

[6] S. Gao and Y. Ng. Recommending Answers to Math Questions Based on KL-Divergence and Approximate XML Tree Matching. In *Proc. of ACM SIGIR-AP*, pages 21–31, 2023.

[7] D. Hendrycks et al. Measuring Mathematical Problem Solving with the Math Dataset. In *Proc. of the NIPS Track on Datasets and Benchmarks*, 2021.

[8] R. Jiang et al. How Mathematics Anxiety Affects Students' Inflexible Perseverance in Mathematics Problem-Solving: Examining the Mediating Role of Cognitive Reflection. *Educational Psychology*, 91(1):237–260, 2021.

[9] Z. Levonian et al. Retrieval-Augmented Generation to Improve Math Question-Answering: Trade-Offs between Groundedness and Human Preference. *arXiv preprint arXiv:2310.03184*, 2023.

[10] A. Lewkowycz et al. Solving Quantitative Reasoning Problems with Language Models. *Advances in Neural Information Processing Systems*, 35:3843–3857, 2022.

[11] Y. Liu, K. Ding, and Y. Zhou. Aifu at semeval-2019 task 10: A symbolic and sub-symbolic integrated system for sat math question answering. In *Proc. of the 13th Workshop on Semantic Evaluation*, pages 900–906, 2019.

[12] X. Luo, A. Baranova, and J. Biegert. Problemsolver at Semeval-2019 Task 10: Sequence-to-Sequence Learning and Expression trees. In *Proc. of the 13th Workshop on SemEval*, pages 1292–1296, 2019.

[13] T. Luong et al. Reft: Reasoning with Reinforced Fine-Tuning. In *Proc. of ACL*, pages 7601–7614, 2024.

[14] D. MacIsaac.US Government Releases Charting a Course for Success:America's Strategy for STEM Education, Report Guiding Federal Agencies that Offer STEM Funding Opportunities. *Physics Teacher*, 57(2):126–126, 2019.

[15] A. Malinin and M. Gales. Reverse KL-Divergence Training of Prior Networks: Improved Uncertainty and Adversarial Robustness. *ANIPS*, 32, 2019.

[16] B. Mansouri, R. Zanibbi, D. Oard, and A. Agarwal. Overview of ARQMath-2 (2021): 2nd CLEF Lab on Answer Retrieval for Questions on Math. In *Proc. of Experimental CLEF*, pages 215–238, 2021.

[17] T. Nguyen, K. Chang, and S. Hui. A Math-Aware Search Engine for Math Question Answering System. In *Proc. of ACM CIKM*, pages 724–733, 2012.

[18] A. Rattan, C. Good, and C. Dweck. 'It's OK — Not Everyone can be Good at Math': Instructors with an Entity Theory Comfort (and Demotivate) Students. *Experimental Social Psychology*, 48(3):731–737, 2012.

[19] P. Sojka, M. Ruzicka, and V. Novotny. MIaS: Math-Aware Retrieval in Digital Mathematical Libraries. In *Proc. of ACM CIKM*, pages 1923–1926, 2018.

[20] Y. Stathopoulos, S. Baker, M. Rei, and S. Teufel. Variable Typing: Assigning Meaning to Variables in Mathematical Text. In *Proc. of the ACL*, pages 303–312, 2018.

[21] Z. Yuan et al. Scaling Relationship on Learning Mathematical Reasoning with Large Language Models. In *Proc. of ICLR*, 2024.

[22] R. Zanibbi, D. Oard, A. Agarwal, and B. Mansouri. Overview of ARQMath 2020: CLEF Lab on Answer Retrieval for Questions on Math. In *Proc. of CLEF*, pages 169–193, 2020.