Mining Misconceptions in Mathematics

Bryan Constantine Sadihin **Hector Rodriguez Rodriguez** **Matteo Jiahao Chen**
Department of Computer Science
Tsinghua University
{wangwd24,lad24,chenjiah24}@mails.tsinghua.edu.cn

Abstract

Multiple-choice questions are widely used to evaluate student knowledge. Well-designed questions use distractors that are associated with common misconceptions. Large language models (LLMs) have performed well in math reasoning benchmarks, but they struggle with understanding misconceptions. In this work, we propose a method to determine the misconception that leads to an incorrect answer. We use an LLM of the Qwen 2.5 family to hypothesize a potential misconception, which is used to assist in the retrieval of the related misconceptions from a list of 2,587 categories. The retrieval process leverages embeddings generated by a fine-tuned Mistral-based LLM trained with a synthetic dataset. The relevant misconceptions are then analyzed by Qwen 2.5, which uses a logits processor to determine the most likely misconception. We evaluate this method using mean average precision on a Kaggle dataset of 1,868 math-related multiple-choice questions, achieving a maximum score of 0.4706. Our results demonstrate the potential of LLMs for assessing incorrect answers and identifying misconceptions in math education.

1 Introduction

From September 12 to December 13, 2024, the "Mining Misconception in Mathematics" competition [1] was hosted on the Kaggle platform. This competition challenged the participants to develop a model capable of predicting the student's misconceptions that lead them to select the incorrect answers, also known as distractors, in multiple-choice questions (MCQs).

This competition was organized by Eedi, an educational platform that helps students improve their understanding of mathematics. Eedi is known for its diagnostic questions, which are specifically designed to identify misconceptions and gaps in students knowledge. The distractors in these questions reflect common misunderstandings, allowing teachers to quickly assess where students might be struggling and to tailor their teaching accordingly. Distractors reflect common misconceptions and serve as tools to identify knowledge gaps. However, determining the relationship between distractors and misconceptions is a challenging and time-intensive task. Automating misconception mining would improve the efficiency of multiple-choice question assessments.

Reasoning about misconceptions poses greater challenges than reasoning about correct answers. Understanding the correct answer is the starting point for determining errors in the reasoning process, and high-quality questions often involve a wide variety of misconceptions, which are difficult to classify using traditional methods.

This project works toward creating models that can suggest relevant misconceptions for specific distractors, thereby streamlining what is typically a manual and time-consuming task for teachers.

Machine Learning (80245013-0) Final Report 2024.

2 Related work

Large Language Models for Math World Problems. Math Word Problems present multiple challenges for large language models (LLMs). These problems, presented in the form of written descriptions, are solved by formulating equations and require advanced reasoning [2]. They can be categorized into three types:

- Question-Answer: A basic setup where each math question is only paired with an answer. For example, SAT-Math [3] provides a high-school SAT Math dataset with multiple-choice questions.
- Question-Equation-Answer: In this setup, each question is paired with both the answer and the solving equation. Datasets like SVAMP [4] and ParaMAWPS [5] target elementary school-level problems in English, with equivalents in Chinese like Math23K [6] and CM17K [7].
- Question-Rationale-Answer: This includes not only the answer but also the reasoning behind it, akin to the Chain-of-Thought methodology. Models like SAT-Math-COT [8] utilize reasoning steps for problem-solving guidance, often generated using program induction or LLMs like AQUA [9].

Fine-tuning LLMs with these type of math-specific datasets can enhance their reasoning abilities significantly [10]. Advanced prompting techniques like Self-Consistency [11] improve the results by selecting the most consistent answer from multiple reasoning paths.

Qwen 2.5 Qwen 2.5 is an open source family of models that excels at mathematical reasoning, making it particularly effective for applications in math-related tasks such as misconception mining [12]. Qwen2.5 is available with 0.5B, 1.5B, 3B, 7B, 14B, 32B, and 72B parameters. It has demonstrated top-tier performance on a wide range of benchmarks when compared with other open and proprietary models.

SFR-Embedding-2 This open source model achieves state-of-the-art performance on the Massive Text Embedding Benchmark (MTEB) leaderboard [13]. Similarly to other top models, it is based on Mistral-7B [14]. SFR-Embedding-2 excels in retrieval augmented generation tasks.

Logits Processor. The logits processor by NVIDIA [15] operates by adjusting the model's output logits. In a multiple-choice scenario it can be used to effectively "force" the model to follow the instructions and select the most appropriate option. The logits processor can also be used to directly obtain a ranked ordering of candidates [16].

3 Preliminaries

Dataset. The dataset is composed of MCQs with four options and a single correct answer. Each incorrect answer is assigned a misconception tag. Additionally, as listed on Table 1, each question is supplemented with the question's subject and construct. The misconception tags are mapped to detailed descriptions on a separate file.

There are 163 subjects and 757 constructs. Figure 1a shows that while some subjects are overrepresented, appearing more than 40 times, others appear less than 10 times. Additionally, Fig. 1b shows that the constructs are rarely reused in different questions.

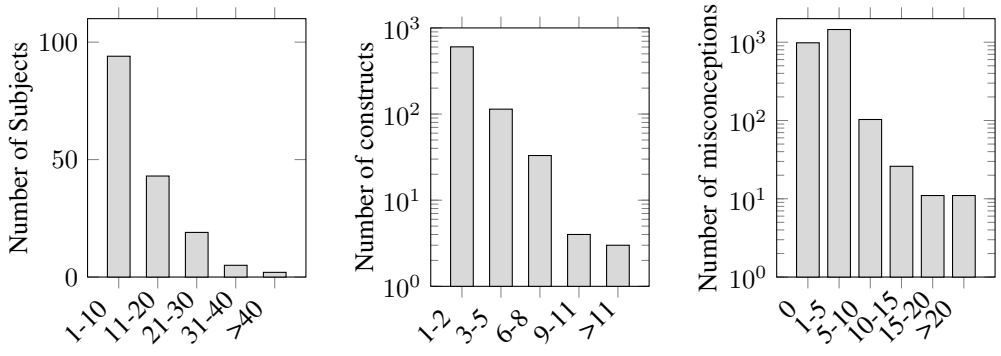
There are 2,587 possible misconceptions. Figure 1c shows that the misconception distribution is imbalanced, with 983 misconceptions not appearing in any of the questions. This indicates that the dataset should be supplemented with synthetic data, and that the final model should be able to understand each misconception from a limited number of examples.

Evaluation Metrics We evaluate the model using the Mean Average Precision (MAP), the Mean Reciprocal Rank (MRR), the Normalized Discounted Cumulative Gain (NDCG), and the Recall with the first 25 elements. They are calculated as follows:

[1]The contents were obtained by Eedi using human-in-the-loop Optical Character Recognition.

Field	Type	Description
`SubjectId`	int	Unique subject identifier
`SubjectName`	str	General context than the construct
`ConstructId`	int	Unique construct identifier
`ConstructName`	str	Granular level of knowledge related to the question
`QuestionId`	int	Unique question identifier
`QuestionText`	str	Question text extracted from the question image [1]
`Answer[A/B/C/D]Text`	str	Answer text extracted from the question image
`CorrectAnswer`	str	A, B, C, or D.
`Misconception[A/B/C/D]Id`	int	Unique misconception identifier

Table 1: Dataset structure



(a) Subject frequency (b) Construct frequency (c) Misconception frequency

Figure 1: Dataset distribution.

$$\text{MAP@25} = \frac{1}{U}\sum_{u=1}^{U}\sum_{k=1}^{\min(n,25)} P(k) \times \text{rel}(k) \tag{1}$$

$$\text{MRR@25} = \frac{1}{U}\sum_{u=1}^{U}\sum_{k=1}^{\min(n,25)} \frac{\text{rel}(k)}{k} \tag{2}$$

$$\text{NDCG@25} = \frac{1}{U}\sum_{u=1}^{U}\sum_{k=1}^{\min(n,25)} \frac{\text{rel}(k)}{\log_2(k+1)} \tag{3}$$

$$\text{Recall@25} = \frac{1}{U}\sum_{u=1}^{U}\frac{\sum_{k=1}^{\min(n,25)} \text{rel}(k)}{R_u} \tag{4}$$

Where U is the total number of observations, $P(k)$ is the precision at cutoff k, n is the number of predictions per observation, and $rel(k)$ is an indicator function. The indicator function is 1 if the item at rank k is the correct label, 0 otherwise. R_u is the total number of relevant items for observation u.

The competition evaluates submissions using the MAP@25 score, which considers the position and order of the predicted misconceptions. The MRR only considers the rank of the first relevant item, but since the dataset has a single misconception per incorrect answer, the MRR and MAP scores are equivalent. The NDCG applies a logarithmic discount to lower-ranked items. It provides a more nuanced evaluation because it ensures that items ranked lower receive diminishing credit. Finally, the Recall counts the number of relevant items that were retrieved, regardless of rank.

4 Methodology

We addressed the imbalances in the dataset that were analyzed in Section 3 by generating a supplementary synthetic dataset. This process is described in Subsection 4.1.

The misconception inference starts by preprocessing the dataset questions using a LaTeXcompiler. In Subsection 4.2 we create an inference framework with Qwen 2.5 by first analyzing the incorrect answers and then by using the reasoning behind the misconception as Hypothetical Document Embedding (HyDE) [17]. Next, in Subsection 4.3 we query the misconception space with a fine-tuned SFR Embedding 2 model to retrieve the 25 most relevant misconceptions. Finally, in Subsection 4.4 Qwen 2.5 is used again to perform a logit-based reranking of the retrieval results. The complete inference process is described in Figure 2.

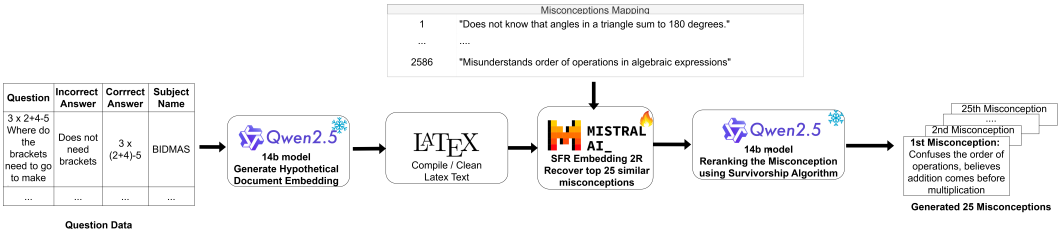


Figure 2: Inference pipeline

4.1 Synthetic data generation

To address the imbalanced distribution illustrated in Section 3, we designed a pipeline to generate a synthetic dataset. The dataset includes the question, subject, and correct and incorrect answers with their corresponding explanations.

Initially, we attempted to create the dataset from scratch using Qwen 2.5 32B. We employed zero-shot and few-shot prompting, providing an example of the desired output. However, this approach yielded inconsistent results, with variations in both the quality and relevance of the questions, particularly in the applicability of the misconceptions. Additionally, the model struggled to generate meaningful incorrect answers.

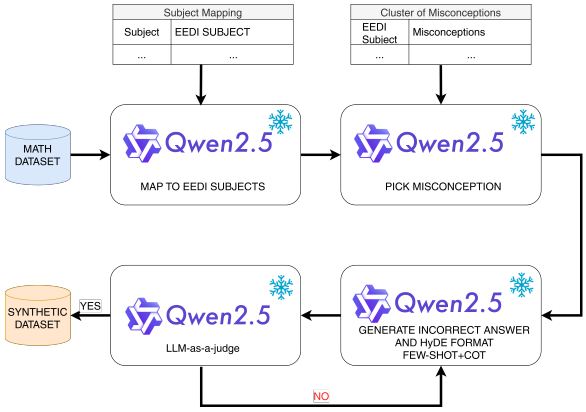


Figure 3: Synthetic data generation pipeline

Given the limitations of the initial approach, we resorted to modifying an existing math dataset. This decision was driven by the need for a reliable foundation that included well-formulated questions and correct answers, allowing us to focus on the integration of misconceptions and detailed explanations. We opted for the MATH dataset [18], as it aligns well with our requirements. The problems are categorized by difficulty level and subject. Additionally, it includes the reasoning that supports the correct answer.

Our final synthetic data generation pipeline is illustrated in Figure 3. Initially, Eedi subjects are mapped to the corresponding MATH subjects to ensure proper alignment. Following this, misconceptions are clustered according to their relevant Eedi subjects. For each MATH question, the most appropriate subject is identified, and then the most relevant misconceptions are selected. Employing a combination of few-shot and chain-of-thought prompting techniques, we then generate the incorrect answer and its corresponding HyDE based on the correct answer and the identified misconception. Once the incorrect answer and its HyDE are produced, they undergo evaluation by an LLM as a judge [19]. The LLM-as-a-judge is given the task introduction, the evaluation steps, and the evaluation criteria. If the evaluation score is below he threshold, the question is deemed unsatisfactory and consequently excluded. This process ensures that only high-quality questions are included in the final dataset. Figure 3 shows the generation process. Additional details of the dataset structure, the question generation prompts, and the LLM-as-a-judge evaluation can be found in the Appendix.

4.2 Reasoning as Hypothetical Document Embedding

Instead of directly retrieving documents from the corpus, HyDe-based retrieval systems first generate a hypothetical document based on the query [17]. This method substantially improves retrieval by accompanying queries with their associated reasoning before the embedding-based search. In this work, relying solely on question-level information—like the question string, incorrect answers, correct answers, and subject names—may be inadequate for accurately retrieving misconceptions.

We propose multiple approaches using question-level information and advanced reasoning to generate the potential misconceptions that can be used as HyDEs. We adopt zero-shot learning for misconception generation, enabling the LLM to produce contextually relevant and accurate misconception reasoning without pre-training on specific examples, enhancing adaptability and robustness across diverse educational contexts.

Building on self-consistency research, we extended our experiments by implementing zero-shot reasoning k times [11]. We utilize a self-consistency prompt to select the best results. The detailed prompts are available in the Appendix.

To ensure compatibility with the computational resources during inference, we utilize the Qwen 2.5 model for generating misconception reasoning. The model is recognized for its superior mathematical reasoning capabilities among open-source models [20]. Specifically, we deploy the 14B parameter variant to fit within the constraints of Kaggle GPU and running time regulations, as outlined in Figure 2.

4.3 Embedding search

The selected model to perform the misconception retrieval is SFR-Embedding-2 [21]. However, due to the specificity of the task at hand, the baseline performance was subpar. The model was fine-tuned using hard negative mining and multiple negatives ranking loss.

To fine-tune the model efficiently, the baseline model was quantized. Using `BF16` reduced the memory consumption by 50% while maintaining the same exponent size as `FP32`. Maintaining the more sensitive gradient computation in `FP32` ensured that the fine-tuning was faster without accuracy degradation. Furthermore, reducing the memory utilization not only speeds up the training due to the larger batch size, but also results in more accurate gradient estimates and smoother convergence because they average over more data points.

LoRA is a low-rank decomposition method to reduce the number of trainable parameters when fine-tuning large models. Figure 4 compares the fine-tuning process with and without LoRA. Since the A and B matrices are smaller than W, LoRA speeds up the fine-tuning and uses less memory [22]. Hugging Face provides a Paremeter Efficient Fine Tuning (PEFT) class that can be used to wrap a frozen model [23] for fine-tuning. PEFT initializes the LoRA weights with Kaiming-uniform [24] for the matrix A and zeros for B, resulting in an identity transform.

The objective function L is Multiple Negatives Ranking Loss (MNRL). Triplet Loss is a common loss function in supervised similarity training. However, it uses a single positive and negative for each query. MNRL makes use of all other elements in the batch as negatives, thus leveraging the information available in the batch more effectively. Thus, larger batch sizes mean more negatives are available for each query, improving the quality of the negative pool. This leads to a faster convergence,

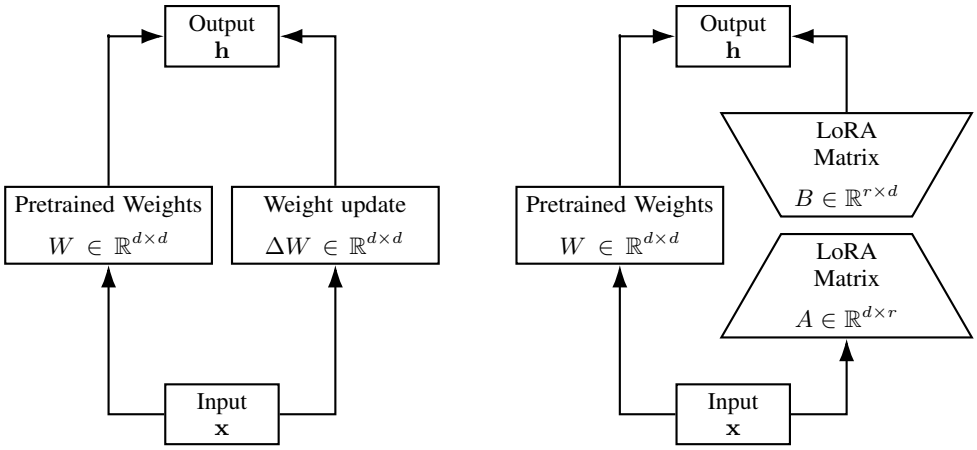


Figure 4: LoRA matrices A and B approximate the weight update matrix ΔW .

better generalization, and stronger performance. To formally define MNRL, let $\mathbf{q}_i$ be the query vector and $\{\mathbf{t_j}\}_{j=1}^{N}$ be the set of targets for all queries in the batch. The incorrect misconceptions for $\mathbf{q}_i$ are the targets $\mathbf{t}_j$ for other queries in the same batch ($j \neq i$).

MNRL ensures that the distance between the query $\mathbf{q}_i$ and its target $\mathbf{t}_i$ is smaller than the distance between $\mathbf{q}_i$ and any incorrect target $\mathbf{t}_j$ from the same batch. This is shown in Fig. 5, and it can be expressed as:

$$L_i = \sum_{j \neq i} \max(0, d(\mathbf{q}_i, \mathbf{t}_i) - d(\mathbf{q}_i, \mathbf{t}_j) + \epsilon) \tag{5}$$

Where ϵ is a margin to ensure separation, i represents the current query's index and $d(\cdot, \cdot)$ is the cosine distance, which measures the dissimilarity between two vectors using the cosine of the angle between them. Given two vectors $\mathbf{A}$ and $\mathbf{B}$, the cosine distance d is calculated as:

$$d = 1 - \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} \tag{6}$$

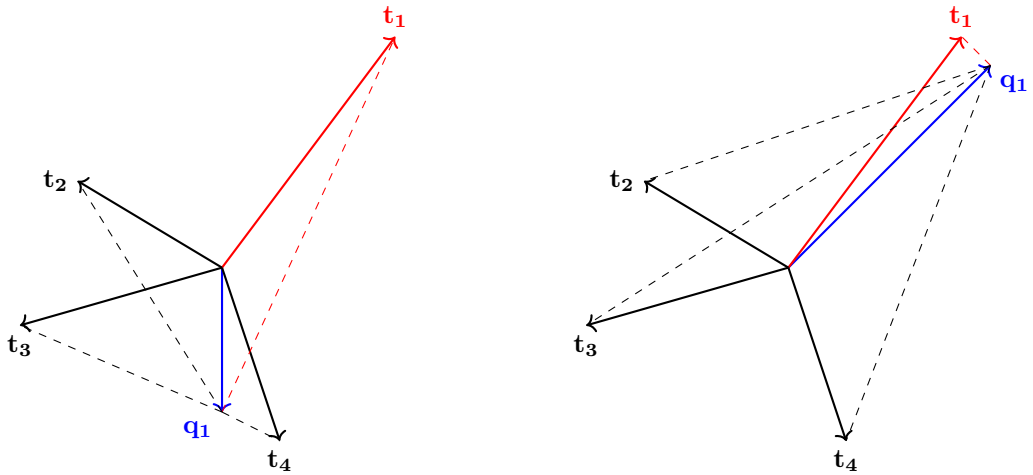


Figure 5: Multiple Negatives Ranking Loss

Hard Negative Mining is a technique used in information retrieval tasks that obtains texts that are rather similar to the target, but are not a correct match. As shown in Figure 6, they are difficult to distinguish from the correct answer, often resulting in a stronger model after training. Hugging Face's Sentence Transformer v3.1 recently released a hard negative mining utility [25].

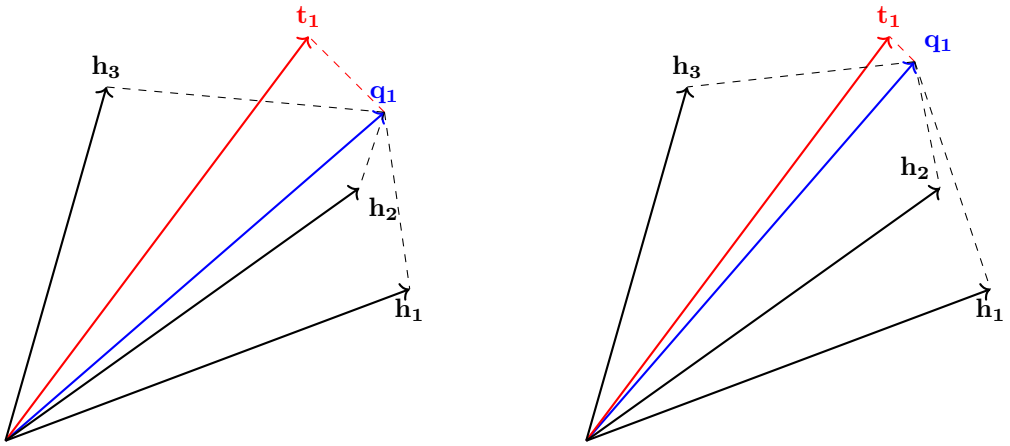


Figure 6: Multiple Negatives Ranking Loss with Hard Negative Mining

Let $\{\mathbf{h_j}\}_{j=1}^{N}$ be the set of hard negatives that was mined for each target in the batch. The incorrect misconceptions for $\mathbf{q}_i$ are the hard negatives $\mathbf{t}_j$ for the i-th query and all the other queries in the batch ($i \neq j$). The MNRL also supports hard negative mining. It ensures that the distance between the query $\mathbf{q}_i$ and its target $\mathbf{t}_i$ is smaller than the distance between $\mathbf{q}_i$ and the hard negative target $\mathbf{h}_j$ from the same batch. This is shown in Figure 6, and it can be expressed as:

$$L_i = \sum_{j=1}^{N} \max(0, d(\mathbf{q}_i, \mathbf{t}_i) - d(\mathbf{q}_i, \mathbf{h}_j) + \epsilon) \tag{7}$$

Where $d(\cdot, \cdot)$ is the cosine distance, ϵ is a margin to ensure separation, and i represents the current query's index.

4.4 Logits Reranking

We incorporate a LLM framework that uses as a reranker algorithm for the limited pool of misconceptions that are retrieved using the embedding search. It is particularly effective for reranking because it can understand and process a wider range of contextual information and complex decision-making logic such as mathematical misconceptions.

Similar to the HyDE approach, we use Qwen 2.5 to power our framework due to its mathematical reasoning ability among all other open source models [20]. Specifically, we can only use the 14b model to ensure that the whole inference pipeline mentioned in Figure 2 can met the Kaggle GPU requirements and running time regulations.

We detail our full reranking algorithm in Figure 7, introducing two key hyperparameters, *n* and *k*. These parameters define the number of misconceptions evaluated in a single pass and the number of top relevant misconceptions to retrieve [16]. Using the logistical framework provided by NVIDIA [15], our approach transforms the reranking process into a multiple-choice task where Qwen 2.5 selects the best answer from a limited pool of misconceptions.

The LLM assesses the *n* misconceptions on each round, starting from those deemed least relevant to ensure the most accurate candidates are not prematurely dismissed. The top *k* relevant misconceptions identified are then carried forward to the next evaluation phase, along with other misconceptions selected from a pool of unchecked misconceptions. This "survivorship algorithm" is an iterative process that continues until all misconceptions are evaluated, placing the top *k* misconceptions at the forefront of the final results, while the remaining 25-*k* misconceptions are ranked according to their original order from the embedding model. This system effectively refines the selection of misconceptions by integrating advanced logical reasoning capabilities of the LLM into the decision-making process, enhancing the relevance and accuracy of the final results.

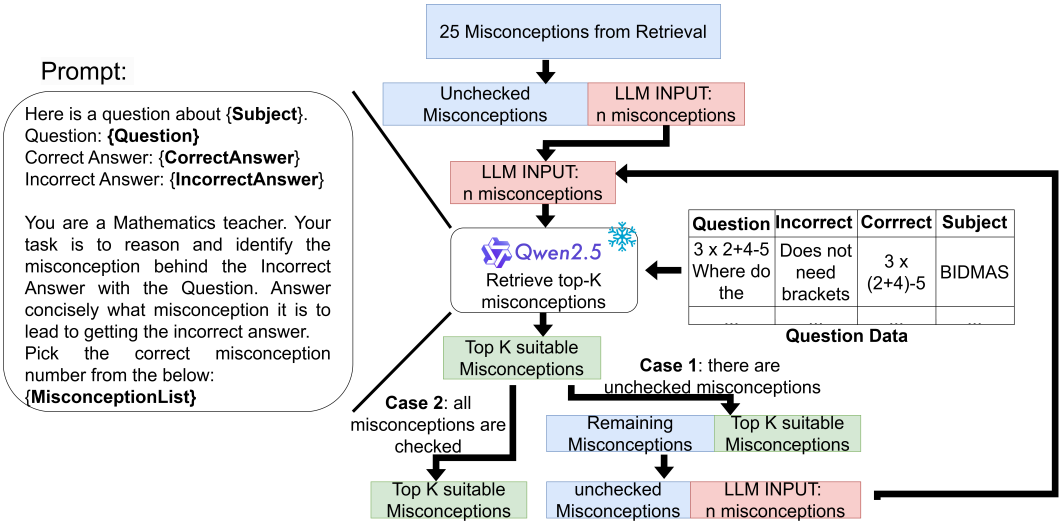


Figure 7: Reranking algorithm with LLM pipeline: Survivorship Algorithm

5 Results and Analysis

The baseline models were compared to the SFR Embedding 2 model in Table 2. The SFR Embedding 2 model performed better than the smaller BGE models out-of-the-box across all metrics. Specifically, it achieved a MAP@25 of 0.2234, which outperformed BGE-large-en and BGE-m3. This indicates that due to its larger size the SFR model has superior performance in retrieval tasks even before fine-tuning. Furthermore, SFR slightly outperformed Mistral, a model with the same number of parameters. This can be attributed to the two-time retrieval technique that improves its ranking capabilities.

Model	MAP@25	NDCG@25	Recall@25
BGE-large-en	0.1936	0.2750	0.5658
BGE-m3	0.1940	0.2707	0.5480
Mistral	0.1972	0.2760	0.5604
SFR Embedding 2	0.2234	0.3065	0.6056

Table 2: Model Selection

The fine-tuning was conducted on a single NVIDIA H100 GPU. All models were trained for 6 epochs using a third of the original training data for validation. The training process used a stratified split, ensuring that the distribution of topics was consistent across the training and validation sets. This stratification also prevented data leakage by ensuring that all incorrect answers from the same question were included in the same partition, which is critical for maintaining the integrity of the evaluation process. The Multiple Negatives Ranking Loss achieved the best results with a batch size of 12 and 8 gradient accumulation steps. This configuration proved to provide the best balance between computational efficiency and model performance.

Fine-tuning was performed on the best baseline model with various configurations, including No HyDE, Self-consistency HyDE, Zero-shot HyDE, and Zero-shot HyDE with Hard Negatives. The results in Table 3 show a clear improvement in performance as we move from No HyDE to Zero-shot HyDE with Hard Negatives, which achieved the highest MAP@25 (0.4364), NDCG@25 (0.5277), and Recall@25 (0.8381).

Interestingly, the fine-tuning with and without self-consistency did not show a significant difference in performance. This indicates that while self-consistency is a promising approach, a k factor of five it did not provide an additional boost in model performance in this particular setup. The most effective fine-tuning configuration was Zero-shot HyDE with Hard Negatives, suggesting that incorporating challenging negative examples during training can significantly improve retrieval performance.

Configuration	MAP@25	NDCG@25	Recall@25
No HyDE	0.3284	0.4351	0.8065
Self-consistency HyDE	0.3454	0.4465	0.7942
Zero-shot HyDE	0.3593	0.4589	0.8045
Zero-shot HyDE, Hard Negatives	0.4364	0.5277	0.8381

Table 3: Fine-Tuning Results

To prevent overfitting, we employed a LoRA technique with a rank of 8. This method reduced the parameter space while maintaining the model's expressive power, making it particularly useful for fine-tuning large models like SFR. Additionally, weight dropout was applied to further mitigate overfitting, which ensured that the model was more generalized and robust, especially when trained on limited data.

As seen in Table 4, the reranking approach improved the model's MAP@25 score, with the Top-3 reranking model yielding a MAP@25 score of 0.4706. Although the improvement was not drastic compared to the fine-tuning stage, this indicates that reranking offers an additional layer of refinement for improving the ranking of relevant items after the initial retrieval stage.

Configuration	MAP@25	NDCG@25	Recall@25
Best Retrieval Result	0.4364	0.5277	0.8381
Best Retrieval Result w/ Top-1 Reranking	0.4404	0.5314	0.8381
Best Retrieval Result w/ Top-3 Reranking	0.4706	0.5558	0.8381

Table 4: Reranking Results

The results highlight the effectiveness of fine-tuning, especially with hard negative examples, and show the significant impact of retrieval techniques like two-time retrieval. The SFR model, which outperformed the other baseline models out-of-the-box, demonstrated its potential for high-quality retrieval tasks even before fine-tuning. The minimal difference between training with and without self-consistency suggests that further research may be needed to fully exploit the benefits of this technique. Finally, overfitting was successfully mitigated through LoRA and weight dropout, ensuring robust model performance.

6 Conclusion

In this work, we present a framework that utilizes LLMs and embedding-based retrieval techniques to extract misconceptions related to incorrect answers in multiple-choice mathematics questions. Our approach aims to automate the process of misconception mining, which is essential for identifying common student errors and improving educational assessments.

We show that hard negative mining is crucial for enhancing the model's ability to distinguish subtle differences between misconceptions. By introducing challenging examples that are difficult for the model to differentiate, we enable better performance and more precise misconception identification. Additionally, we employ multiple negative ranking loss during training, which helps the model converge faster and generalize more effectively by exposing it to a broader set of negative examples. Furthermore, a reranking mechanism is incorporated to refine the selection of relevant misconceptions, ensuring that only the most pertinent ones are considered for further analysis.

Our approach demonstrates a robust methodology for misconception mining, leveraging fine-tuned embeddings, advanced loss functions, and reranking to enhance the accuracy of identifying misconceptions in mathematics. By automating this process, the model can provide valuable insights into student performance, allowing educators to more effectively identify areas where students struggle and tailor teaching methods accordingly. In the future, we aim to extend this framework to other educational domains and improve its generalization to new, unseen misconceptions.

References

[1] Kaggle and Eedi. Eedi - mining misconceptions in mathematics dataset. https://www.kaggle.com/competitions/eedi-mining-misconceptions-in-mathematics, 2024. Accessed: 2024-10-26.

[2] Janice Ahn, Rishu Verma, Renze Lou, Di Liu, Rui Zhang, and Wenpeng Yin. Large language models for mathematical reasoning: Progresses and challenges. *arXiv preprint arXiv:2402.00157*, 2024. EACL 2024 Student Research Workshop.

[3] Wanjun Zhong, Ruixiang Cui, Yiduo Guo, Yaobo Liang, Shuai Lu, Yanlin Wang, Amin Saied, Weizhu Chen, and Nan Duan. Agieval: A human-centric benchmark for evaluating foundation models, 2023.

[4] Arkil Patel, Satwik Bhattamishra, and Navin Goyal. Are nlp models really able to solve simple math word problems?, 2021.

[5] Syed Rifat Raiyan, Md Nafis Faiyaz, Shah Md. Jawad Kabir, Mohsinul Kabir, Hasan Mahmud, and Md Kamrul Hasan. Math word problem solving by generating linguistic variants of problem statements. In Vishakh Padmakumar, Gisela Vallejo, and Yao Fu, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 4: Student Research Workshop)*, pages 362–378, Toronto, Canada, July 2023. Association for Computational Linguistics.

[6] Zihao Zhou, Maizhen Ning, Qiufeng Wang, Jie Yao, Wei Wang, Xiaowei Huang, and Kaizhu Huang. Learning by analogy: Diverse questions generation in math word problem, 2023.

[7] Jinghui Qin, Xiaodan Liang, Yining Hong, Jianheng Tang, and Liang Lin. Neural-symbolic solver for math word problems with auxiliary tasks, 2021.

[8] Nathan Davidson. Sat math chain-of-thought dataset. https://huggingface.co/datasets/ndavidson/sat-math-chain-of-thought, 2023. Accessed: 2024-10-26.

[9] Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. Program induction by rationale generation : Learning to solve and explain algebraic word problems, 2017.

[10] Yixin Liu, Avi Singh, C. Daniel Freeman, John D. Co-Reyes, and Peter J. Liu. Improving large language model fine-tuning for solving math problems, 2023.

[11] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models, 2023.

[12] An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

[13] Niklas Muennighoff, Nouamane Tazi, Loïc Magne, and Nils Reimers. Mteb: Massive text embedding benchmark, 2022. Accessed: 2024-12-29.

[14] Mistral AI. Mistral-7b-v0.1: A 7-billion-parameter language model, 2023. Accessed: 2024-12-29.

[15] NVIDIA Corporation. Logits processor zoo. https://github.com/NVIDIA/logits-processor-zoo, 2024. Accessed: YYYY-MM-DD.

[16] Revanth Gangi Reddy, JaeHyeok Doo, Yifei Xu, Md Arafat Sultan, Deevya Swain, Avirup Sil, and Heng Ji. First: Faster improved listwise reranking with single token decoding, 2024.

[17] Luyu Gao, Xueguang Ma, Jimmy Lin, and Jamie Callan. Precise zero-shot dense retrieval without relevance labels, 2022.

[18] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *NeurIPS*, 2021.

[19] Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. G-eval: Nlg evaluation using gpt-4 with better human alignment, 2023. Microsoft Cognitive Services Research.

[20] An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, Keming Lu, Mingfeng Xue, Runji Lin, Tianyu Liu, Xingzhang Ren, and Zhenru Zhang. Qwen2.5-math technical report: Toward mathematical expert model via self-improvement, 2024.

[21] Rui Meng, Ye Liu, Shafiq Rayhan Joty, Caiming Xiong, Yingbo Zhou, and Semih Yavuz. Sfr-embedding-2: Advanced text embedding with multi-stage training, 2024. Accessed: 2024-12-29.

[22] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models, 2021. Draft V2 includes better baselines, experiments on GLUE, and more on adapter latency.

[23] Hugging Face. *LoRA: Low-Rank Adaptation in PEFT*, 2024. Accessed: 2024-12-29.

[24] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 1026–1034, 2015.

[25] Tom Aarsen. Sentence transformers v3.1 release: Hard negatives mining utility and more, 2023. Accessed: 2024-12-28.

A Synthetic dataset generation

A.1 Dataset structure

Subject	Algebra
Question	What is the positive difference 120% of 30 and 130% of 20?
Solution	120% of 30 = 36, 130% of 20 = 26, the answer is 36 - 26 = 10

Table 5: MATH dataset structure

Eedi Subject	Algebra
Question	What is the positive difference 120% of 30 and 130% of 20?
Correct Answer	120% of 30 = 36, 130% of 20 = 26, the answer is 36 - 26 = 10
Incorrect answer	1000
Misconception Id	2322
Misconception	Believes a percentage of a number will be a multiple of the percentage
HyDE	120% of 30 = 3600, 130% of 20 = 2600, the answer is 3600 - 2600 = 1000

Table 6: Synthetic dataset structure

A.2 Prompt for Eedi subject selection

Question: What is the positive difference 120% of 30 and 130% of 20?
Problem type: Algebra
Subject List: {List of Eedi subjects}
Your task: You are a genius mathematician. Identify the most relevant subject of the problem from the subject list. Before answering the question, think step by step concisely in 1-2 sentences inside `<thinking> INSERT_TEXT_HERE</thinking>` tag and respond your final subject inside `<response> INSERT_TEXT_HERE</response>` tag. If there is no relevant subject, respond `<response> No subject</response>` tag.

A.3 Prompt for misconception selection

Question: What is the positive difference 120% of 30 and 130% of 20?
Eedi Subject: Percentage
Correct answer: 26
Correct answer reasoning: 120% of 30 = 36, 130% of 20 = 26,the answer is 36 - 26 = 10
Misconception List: { List of misconceptions related to the subject}
Your task: You are a genius mathematician. Identify a possible misconception from the misconceptions list appliable to the correct answer reasoning. Before answering the question, think step by step concisely in 1-3 sentences inside `<thinking> INSERT_TEXT_HERE</thinking>` tag and respond your final Misconception inside `<response> INSERT_TEXT_HERE</response>` tag. If there is no related misconception, respond `<response> No misconception</response>` tag.

A.4 Prompt for misconception reasoning

Question: What is the positive difference 120% of 30 and 130% of 20?
Eedi Subject: Percentage
Correct answer: 26
Correct answer reasoning: 120% of 30 = 36, 130% of 20 = 26,the answer is 36 - 26 = 10
Misconception: Believes a percentage of a number will be a multiple of the percentage.

Example of your task:
Question: $3 \times 2 + 4 - 5$. Where do the brackets need to go to make the answer equal 13?
Eedi Subject:BIDMAS
Correct answer reasoning: Put brackets around 2+4, and give precedence to this operation.
Sum 2+4=6. $3 \times 6 - 5$. Multiply 3 and 6.$18 - 5$=13.
Correct answer:$3 \times (2 + 4) - 5$
Misconception: Confuses the order of operations, believes addition comes before multiplication
Wrong answer reasoning: Sum directly 2+4. $3 \times 6 - 5$. Multiply 3 and 6.$18 - 5 = 13$.
Wrong answer: Does not need brackets.

Your task: You are a student who applied the misconception. Before answering the question, think step by step concisely in 1-3 sentences how the misconception lead to the incorrect answer inside `<thinking> INSERT_TEXT_HERE</thinking>` tag and respond your final incorrect answer inside `<response> INSERT_TEXT_HERE</response>` tag.
Important guidelines:

- Do not restate the problem or misconception.
- Show exactly how the misconception led to the error.

A.5 LLM-as-a-judge

Question: What is the positive difference 120% of 30 and 130% of 20?
Eedi Subject: Percentage
Correct answer: 26
Correct answer reasoning: 120% of 30 = 36, 130% of 20 = 26,the answer is 36 - 26 = 10
Misconception: Believes a percentage of a number will be a multiple of the percentage.
Incorrect answer:1000
Incorrect answer reasoning:120% of 30 = 3600, 130% of 20 = 2600, the answer is 3600 - 2600 = 1000
Your task: You are a genius mathematician. Identify how the misconception could lead to the incorrect answer reasoning. Identify any gaps or inconsistencies in the incorrect answer reasoning Give a brief explanation, think step by step in 1-3 sentences inside `<thinking> INSERT_TEXT_HERE</thinking>`. Lastly, evaluate step by step the misconception and incorrect answer reasoning inside `<evaluating> INSERT_TEXT_HERE</evaluating>` and respond the final score based on the evaluation criteria inside `<response> INSERT_TEXT_HERE</response>`. This is the evaluation criteria for you:

- 5, Perfect alignment, the incorrect answer id directly cause by the misconception.
- 4, Logical path from misconception to incorrect answer with minor gaps.
- 3, Logical path exists but with notable gaps or misapplications.
- 2, Weak alignment with significant reasoning gaps.
- 1, Minimal alignment and reasoning is unclear or erroneous.
- 0, No connection between misconception and incorrect answer.

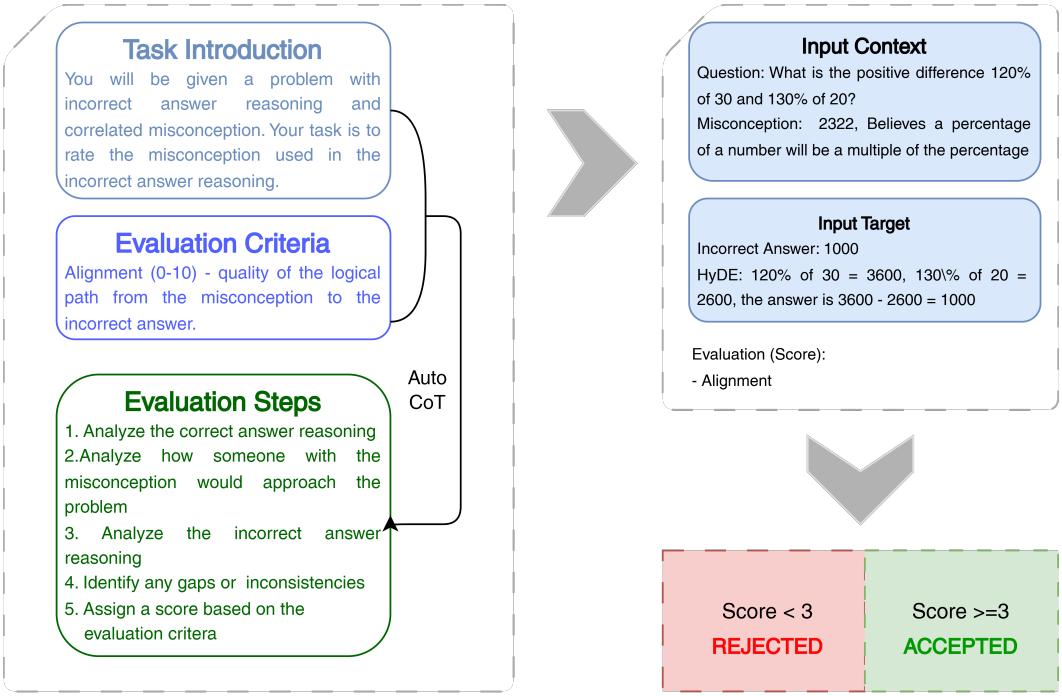


Figure 8: Structure of the LLM-as-a-judge

B Misconception identification

B.1 Prompt for zero-shot reasoning

Subject Name: BIDMAS[2]
Question: Where do the brackets need to go to make the answer equal to 13?
Correct Answer: $3 \times 2 + 4 - 5$
Incorrect Answer: $3 \times (2 + 4 - 5)$
Your task: You are a genius mathematician. Identify the misconception behind Incorrect Answer. Before answering the question, think step by step concisely in 1-2 sentences inside `<thinking> INSERT_TEXT_HERE</thinking>` tag and respond your final misconception inside `<response> INSERT_TEXT_HERE</response>` tag.

B.2 Prompt for self-consistency

Subject Name: BIDMAS
Question: Where do the brackets need to go to make the answer equal to 13?
Correct Answer: $3 \times 2 + 4 - 5$
Incorrect Answer: $3 \times (2 + 4 - 5)$
Your task: You are a genius mathematician. Identify the misconception behind Incorrect Answer from the reasoning paths. Use self-consistency to choose the accurate one. Think step by step concisely in 1-2 sentences inside `<thinking> INSERT_TEXT_HERE</thinking>` tag and respond your final misconception inside `<response> INSERT_TEXT_HERE</response>` tag. **Reasoning paths:** {ReasoningPaths}

[2]BIDMAS stands for Brackets, Indices, Division, Multiplication, Addition, Subtraction. It is a common acronym used to remember the order of operations in mathematics.

B.3 Prompt for embedding generation

Subject: BIDMAS
Construct: Simplify an algebraic fraction by factorising the numerator
Question: Where do the brackets need to go to make the answer equal to 13?
Correct Answer: $3 \times 2 + 4 - 5$
Incorrect Answer: $3 \times (2 + 4 - 5)$
Explanation: Misinterpreted operator precedence; assumes addition before multiplication

B.4 Prompt for reranking

Subject Name: BIDMAS
Question: Where do the brackets need to go to make the answer equal to 13?
Correct Answer: $3 \times 2 + 4 - 5$
Incorrect Answer: $3 \times (2 + 4 - 5)$
Your task: You are a Mathematics teacher. Identify the misconception behind Incorrect Answer with respect to the Question. Answer concisely what misconception it is that leads to getting the incorrect answer. Pick the correct misconception number from the below
Misconception List: {MisconceptionList}