Natural Language Processing - Final
Group 13, NTHU, Fall 2024

Guei-Chen, Wu*
Department of Computer Science
National Tsing Hua University, Taiwan
s111062751@m111.nthu.edu.tw

Chang-Shuo, Chen
Institute of Communications Engineering
National Tsing Hua University, Taiwan
s112064531@m112.nthu.edu.tw

Shu-Rong, Yang
Department of Computer Science
National Tsing Hua University, Taiwan
daphneyang0127@gmail.com

Zi-Jun, Huang
Department of Computer Science
National Tsing Hua University, Taiwan
zijun0502@gmail.com

1 Introduction

This competition [1] aims to develop a Natural Language Processing (NLP) model leveraging Machine Learning (ML) to predict the affinity between misconceptions and distractors in multiple-choice Diagnostic Questions. Each question includes one correct answer and three distractors, carefully designed to reflect specific misconceptions. Accurately tagging these distractors is crucial yet time-consuming, often uncovering new misconceptions as human labelers work on diverse topics. Initial attempts using pre-trained language models have struggled with the complexity of mathematical content, highlighting the need for a more efficient and consistent approach. By aligning distractors with known and emerging misconceptions, the model seeks to support human labelers in improving tagging accuracy and consistency, ultimately enhancing the educational experience for students and teachers.

2 Dataset

The dataset consists of Diagnostic Questions (DQs) designed as multiple-choice questions with one correct answer and three distractors, each representing a potential misconception. Each question targets a specific construct, representing the most granular level of knowledge related to the question. The dataset includes fields such as QuestionId (unique question identifier), ConstructId, and ConstructName (related knowledge construct), CorrectAnswer (correct option: A, B, C, or D), SubjectId and SubjectName (general subject context), QuestionText (extracted text from images using human-in-the-loop OCR), and Answer[A/B/C/D]Text (answer options' text). For each distractor, Misconception[A/B/C/D]Id maps to a unique misconception identifier, with ground truth labels provided in train.csv. The misconception_mapping.csv file links MisconceptionIds to their names, and the task involves predicting MisconceptionIds for distractors in test.csv. A sample submission file (sample_submission.csv) provides the required submission format, where each question-answer pair allows for up to 25 space-delimited MisconceptionIds.

2.1 Distribution

The dataset exhibits a highly skewed distribution of Misconception ID occurrences, as shown in the Figures 1 . On the **X-axis**, the occurrence count of each Misconception ID is plotted, representing how many times each misconception appears in the dataset. On the **Y-axis**, the number of unique Misconception types with a given occurrence count is shown.

The distribution has the following characteristics:

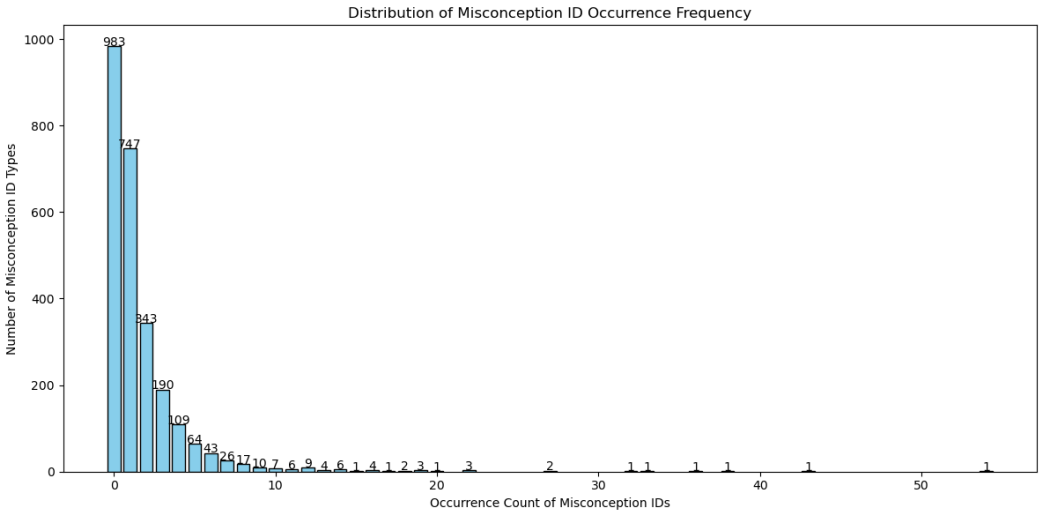

Figure 1: Distribution of Misconception ID Occurrence Frequency

- The chart represents the distribution of Misconception ID occurrence frequencies in the dataset. The X-axis denotes the occurrence count of each Misconception ID, while the Y-axis shows the number of unique Misconception types that appear with the corresponding frequency. For example:
- 983 Misconception types do not appear in the training dataset at all (occurrence count = 0).
- 747 Misconception types appear exactly once in the dataset (occurrence count = 1).
- The distribution shows a sharp decline as the occurrence count increases, indicating that only a small number of misconceptions are frequently observed, while most are rare.

This distribution highlights significant challenges in modeling:

The 983 Misconception types that do not appear in the training set suggest incomplete coverage of potential misconceptions, making it challenging for the model to learn representations for unseen types.

2.2 Missing Value

In the training dataset, which contains 1869 data points with a total of 5607 distractors, we observe that 1237 distractors lack corresponding Misconception IDs. These missing values represent a significant portion of the dataset and can arise due to various reasons, such as incomplete annotation, ambiguity in mapping certain distractors to misconceptions, or the presence of novel misconceptions that were not included in the labeling process.

The absence of Misconception IDs introduces several challenges:

- **Loss of Information:** Distractors without labeled misconceptions cannot contribute directly to the training process, reducing the effective size of the labeled data.
- **Imbalanced Dataset:** The missing values exacerbate the imbalance in the dataset, as they disproportionately affect certain constructs or subjects where misconceptions are harder to identify.

To address these issues, we considered several techniques.

2.3 Data Augmentation

Since there are imbalance distribution in the misconception classes. We aim to generate synthetic data for the rare misconceptions class to improve representation.

2.3.1 Additional Feature Generation

First, we tried to use GPT4o to generate a corresponding explanation for each wrong option. This will add four additional columns for each question in the original dataset. The four columns are as listed below:

1. Answer A Explanation
2. Answer B Explanation
3. Answer C Explanation
4. Answer D Explanation

Although the augmented data do not introduce new problems or misconception types, we assume that it will still help address the imbalance in the following ways:

1. Enhances Feature Representation for Minority Classes
   (a) The augmented data provide additional explanation texts for wrong options.
   (b) It enriches textual features that allow the model to better learn the characteristics of minority classes.
2. Improves Model Learning in Minority Classes
   (a) Models tend to focus on majority classes during training. Augmented explanation texts provide more semantic variations for minority classes, reducing bias toward majority classes.

And we planned to merged the augmented data the model structure as two listed ways:

1. Prompt engineering with additional information: For example, without augmented data, when doing prompt engineering, we can only give the model the text of the problem and each option, and ask the model to map the wrong answers to the given misconception classes. But with the augmented data, we can enclose the explanations for each wrong option in the prompt, the model can based on these information to make predictions, and we believe that this will make the reasoning process easier for the Qwen model since there are additional information provided for both majority and minority misconceptions.
2. Fine-tune LLM using Lora: we can use the augmented data to fine-tune Qwen model, as the added four features and be used to conduct semantic analysis directly, and we can use the fine-tuned model to inference the test data.

The above-mentioned techniques require actually training the model, but after doing some careful research , we observed that people use 4 A100 cards to fine-tune 7B Qwen2.5 model, and we are currently using Qwen2.5 14B and 32B model as our model structure, so it seems almost impossible to complete such a fine-tuning task as we have limited training resource.

2.3.2 Synthetic Data

In our proposal, since there are misconception classes that never appear in the training data, we have surveyed some methods to generate synthetic data to further balance the dataset. The first place on the private leader board has provided some insights of this task:
First, the challenge of synthetic data generation includes the accuracy, diagnostic power, resolution, and diversity of the data, when leveraging LLMs to generate data, it is necessary to set the goal to generate questions that specifically diagnose student misconceptions. Second, the author created clusters of closely related misconceptions leveraging co-occurrence statistics of misconceptions in retriever/ranker predictions on validation data, each cluster grouped similar but distinct misconceptions that could lead to common student errors. These cluster are includes into prompt when generating data to ensure LLMs considered related misconceptions.
After the synthetic data have been generated, the author used GPT-4o as a judge to score the generated data on a 0-10 scale based on alignment between the misconception and the incorrect answers. It evaluated whether the incorrect answer logically stemmed from the given misconception and removed low-quality questions with weak or no alignment.

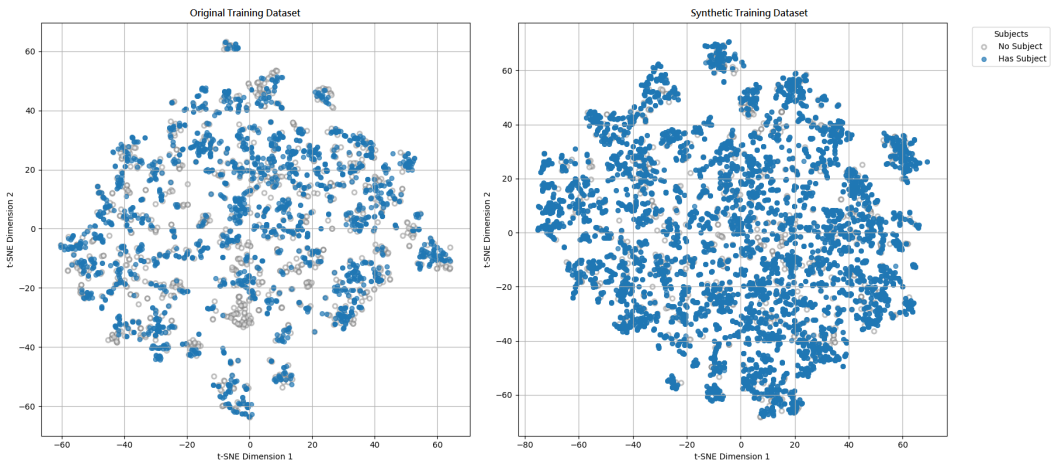

Figure 2: Misconceptions in Original Training Dataset (Left) and Synthetic Training Dataset (Right).

The last step is to manage new misconceptions generated by LLMs that weren't part of the original misconception pool, the author first performed string normalization (e.g., lowercasing, punctuation removal) for matching. And further applied similarity scores from an embedding model (threshold > 0.995 to detect duplicates). After all these steps, the augmented data became more accurate and balanced, contributed to greater model performance.
In our case, the generating part can be done using pre-trained LLMs API call and will not caused any problems. But after the generating process, the model require fine-tune as well, and the problem of limited GPU resources remain the same with 2.3.1 Additional Feature Generation.

2.3.3 Synthetic Dataset Analysis

Figure 2 compares the distribution of misconceptions in the original training dataset and the synthetic training dataset using a t-SNE visualization. The blue points represent misconceptions associated with a subject, while the gray points indicate those without an associated subject. This visualization provides insight into the differences in distribution before and after augmenting the dataset with synthetic data.

In the original training dataset (left panel), the distribution of misconceptions is notably uneven, with certain regions in the embedding space being densely populated while others remain sparse. This imbalance reflects the original dataset's inherent limitations, where certain misconceptions are underrepresented or entirely missing.

In contrast, the synthetic training dataset (right panel) demonstrates a more balanced distribution of misconceptions across the embedding space. The addition of synthetic data effectively fills the gaps in the original dataset, ensuring that previously underrepresented misconceptions are now adequately represented. This improved coverage is particularly evident in areas where gray points were sparse in the original dataset, but blue points now appear more evenly distributed.

The balanced distribution in the synthetic dataset is critical for training machine learning models, as it ensures that the model can generalize across a wider variety of misconceptions. By reducing bias toward overrepresented misconceptions, the synthetic dataset enhances the model's ability to predict less common misconceptions accurately. This highlights the effectiveness of data augmentation in improving the quality and utility of the training dataset.

2.4 Error Analysis

Figure 3 illustrates the position of the ground truth in the retrieval embedding for two different retrieval methods: K-Nearest Neighbors (KNN) and Cosine Similarity. The Position Index on the y-axis represents the rank of the ground truth Misconception ID within the retrieval results, with lower indices indicating better performance. The horizontal blue line represents the cutoff for the top

25 candidates, which is critical for ensuring that the ground truth is included in the set of candidates passed to the reranking stage.

The plot shows that for both KNN and Cosine Similarity, the majority of ground truth positions fall below the cutoff line, indicating that these retrieval methods are generally successful in capturing the correct Misconception ID within the top 25 results. However, several outliers are visible in both cases, where the ground truth appears at significantly higher indices (above 25). These outliers highlight instances where the retrieval methods fail to prioritize the correct Misconception ID, potentially impacting the overall performance of the pipeline.

These findings indicate that while the retrieval stage performs well for most examples, its limitations are primarily seen in outlier cases. Addressing these outliers may require enhancements to the retrieval model, such as incorporating additional contextual features, improving embedding quality, or using hybrid retrieval techniques that combine KNN and Cosine Similarity. Improving these aspects could ensure that the ground truth is consistently included in the top candidates, ultimately leading to better performance in the reranking and final prediction stages.

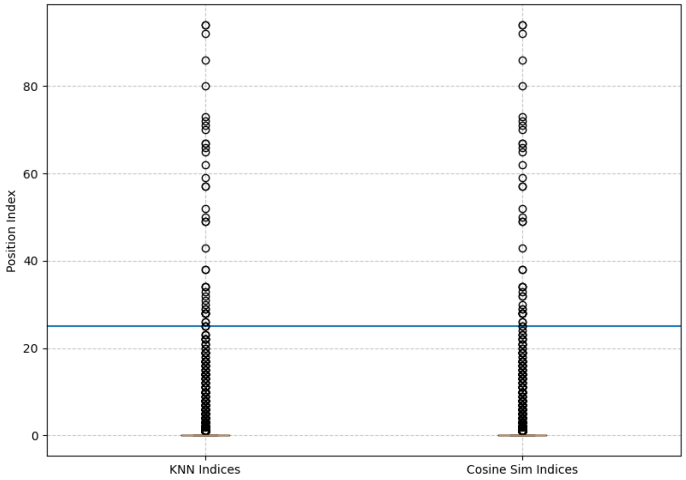

Figure 3: The Position of Ground Truth in Retrieval Embedding.

3 Methodology

3.1 Preprocessing

3.1.1 Prompt Engineering

Prompt engineering plays a crucial role in this competition. Numerous resources and experimental notebooks highlight that even small adjustments to prompts can significantly enhance model performance. For this project, our focus was on refining prompts to optimize the identification of misconceptions in mathematical problems. Specifically, the process was divided into two core tasks:

- Task 1: Retrieve a list of potential misconceptions for a given incorrect answer using a prompt.
- Task 2: Select the most accurate misconception from the retrieved list, given the question, its correct answer, and the incorrect answer.

Original Prompt for Task 1:
Given a math question with correct answer and a misconcepted incorrect answer, retrieve the most accurate misconception for the incorrect answer.

Optimized Prompt for Task 1:
Given a math question, the correct answer, and an incorrect answer caused by a misconception, select the most accurate misconception from the provided list that explains the incorrect answer.

Original Prompt for Task 2:
Here is a question about {ConstructName} ({SubjectName}). Question: {Question} Correct Answer: {CorrectAnswer} Incorrect Answer: {IncorrectAnswer}. You are a Mathematics teacher. Your task is to reason and identify the misconception behind the Incorrect Answer with the Question. Answer concisely what misconception it is to lead to getting the incorrect answer. Pick the correct misconception number from the below: {Retrival}.

Optimized Prompt for Task 2:
You are a Mathematics instructor.
Analyze the given math question about {ConstructName} ({SubjectName}),
its correct answer ({CorrectAnswer}),
and the student's incorrect answer ({IncorrectAnswer}).
From the provided list of misconceptions {Retrival}, select the most accurate misconception number that explains the student's incorrect answer.

Techniques The process of refining the prompts involved several key techniques aimed at enhancing clarity, consistency, and focus in model output. These techniques were carefully chosen and experimented.

1. **Role Specification:** One of the most notable improvements between the original and optimized prompts is the explicit definition of the LLM's role. In Task 2, for example, the original prompt referred to the model as a "Mathematics teacher" and instructed it to reason and explain. The optimized version refined this to the more specific role of "Mathematics instructor," with the task becoming clearer and more structured. This adjustment ensures that the model understands its responsibilities more clearly and thus provides more relevant and focused outputs.
2. **Task Simplification:** The original Task 1 prompt was somewhat vague, asking the model to "retrieve the most accurate misconception for the incorrect answer." The optimized version explicitly specifies that the model must "select the most accurate misconception from the provided list." This small change simplifies the instruction and focuses the task on directly identifying the misconception without any ambiguity about how the list of misconceptions should be handled.
3. **Clarity and Precision in Language:** In both tasks, we aimed to simplify and clarify the language of the prompt. For example, in Task 2, the original prompt used the phrase "reason and identify the misconception behind the Incorrect Answer." While technically clear, the phrasing was more open-ended. In contrast, the optimized version asks the model to "analyze" the question and "select the most accurate misconception," which is more straightforward and task-oriented, leading to more precise responses.
4. **Output Constraints:** The optimized prompts include a sharper focus on output expectations. By clearly stating that the model should select a misconception statement (and not provide explanations or reasoning), the prompt minimizes unnecessary verbosity and ensures that the LLM's responses are concise and aligned with the task's objectives. This aligns with prompt optimization best practices by reducing the chances of irrelevant or excessive outputs.

Performance Evaluation The performance of the original and optimized prompts was evaluated directly using the score on Kaggle.

Table 1: Performance Metrics for Prompt Applications and Incremental Improvements

Prompt Type	Private Score	Score Difference
Apply Both Original	0.433	0
Apply Optimized (1st)	0.453	+4.6%
Apply Optimized (2nd)	0.441	+1.8%
Apply Both Optimized	0.457	+5.5%

Discussion The optimized prompts demonstrated measurable improvements over the original versions, with higher score. The removal of unnecessary reasoning steps and the use of role specification

were particularly effective in achieving these results. These findings align with studies emphasizing the importance of clarity and task-focused instructions in prompt design [2, 3].

3.1.2 Hyper-parameter Optimization

For the LLM fine-tuning process, we optimized several key hyperparameters that control the model's behavior during training. These parameters were adjusted to improve the model's performance on the private data.

Fine-Tuning Parameters We focused on adjusting the following fine-tuning parameters to enhance the model's performance:

- **lora_alpha:** Increased from 128 to 256. The *lora_alpha* parameter scales the weight matrices during the fine-tuning process, effectively controlling the strength of the low-rank adaptation (LoRA) applied to the model. The value was increased to 256 to allow for larger updates to the model's weights, thereby enhancing its ability to adapt to the specific task.
- **lora_dropout:** Increased from 0.05 to 0.1. The dropout rate was raised to improve regularization, helping to reduce overfitting, particularly when a larger *lora_alpha* value is used. This adjustment aims to promote better generalization by preventing the model from memorizing the training data's patterns.
- **target_modules:** The "attention" layer was added to the fine-tuning process. Including the attention layer allows the model to update its attention parameters, enabling it to better focus on key components of the input data.

Inference Parameters The inference parameters were kept consistent throughout, with the following settings:

- **Temperature:** Remains to be 0. This setting ensures deterministic output from the model, meaning the responses are consistent and do not vary with each run, providing predictable results for selecting the most accurate misconception.
- **Top-k:** Remains to be to 1, restricting the model to select only the most probable next token. This minimizes variability in the generated responses, ensuring a focused and predictable output, which is essential when selecting the most accurate misconception.

Grid Search and Manual Tuning We manually optimized hyperparameters using a grid-search approach, as the parameter space was small. This efficient method quickly identified optimal settings without the need for complex algorithms.

3.2 Retrieval

Retrieval is the process of identifying a subset of potential Misconception IDs from the entire pool based on their relevance to a given distractor. This step is crucial for narrowing down the search space and improving the efficiency of subsequent processing stages, such as re-ranking or final prediction.

In this context, retrieval methods focus on leveraging textual and contextual similarities between the distractor and the known misconceptions.

Retrieval serves as the foundation for the entire prediction pipeline, as it directly affects the quality of the candidates provided for re-ranking. An effective retrieval system ensures that the correct Misconception ID is included in the candidate set, minimizing the risk of missing relevant matches. Additionally, retrieval methods must balance precision and recall to maintain a manageable number of candidates while maximizing the likelihood of capturing the correct answer.

3.2.1 Models

The performance of the retrieval system heavily depends on the choice of the underlying model, which is tasked with identifying relevant Misconception IDs for each distractor. Table 6 provides a comparison of several retrieval models, detailing their parameter sizes, the inclusion of re-rankers, and their respective performances on public and private test sets, as well as their average scores.

The Phi3.5 model, with 3.82 billion parameters, serves as a smaller baseline model for retrieval tasks. It achieves average scores of 0.289, which, while modest, provides a starting point for understanding the impact of retrieval on prediction accuracy. By contrast, the SFR-Embedding-2_R model, which has 7.11 billion parameters, significantly improves performance, yielding an average score of 0.331. This demonstrates the benefits of larger embeddings in capturing textual and contextual nuances for more effective retrieval.

Further improvements are observed when combining the SFR-Embedding-2_R model with a 32B re-ranker, which elevates the average score to 0.404. This highlights the effectiveness of introducing a reranking phase to refine the initial retrieval results, showcasing how additional model complexity can enhance precision.

Among the models listed, the Qwen 14B model, paired with a 32B re-ranker, achieves the best overall performance, with an average score of 0.444. This indicates that larger models with sophisticated reranking capabilities can better align distractors with their corresponding misconceptions, leveraging both the scale of parameterization and the precision of reranking.

These results underscore the trade-offs between model complexity and retrieval accuracy. While smaller models like Phi3.5 are computationally efficient, larger models, especially when combined with reranking mechanisms, consistently deliver superior performance. The findings emphasize the importance of balancing computational resources with the need for high retrieval accuracy, depending on the specific application requirements.

Table 2: Retrieval Model Performance Comparison

Retrieval Model	Parameter	Re-ranker	Public	Private	Average	Reference
Phi3.5	3.82B	-	0.309	0.281	0.289	[4]
SFR-Embedding-2_R	7.11B	-	0.353	0.322	0.331	[5]
SFR-Embedding-2_R	7.11B	32B	0.451	0.386	0.404	[6]
Qwen 14B	14B	32B	0.468	0.434	0.444	[7]

3.3 Fine tuning

We constructs a dataset by pairing each question and its correct and incorrect answers with a known misconception. Each training sample is formatted as a combined prompt and corresponding labeled output for fine-tuning. A tokenization process merges system prompts, user prompts, and correct labels into one sequence, ensuring proper attention masking and label alignment. Finally, Low-Rank Adaptation (LoRA) is applied on a pretrained Qwen model to efficiently fine-tune only selected parameters, followed by saving the specialized model and tokenizer[8].

3.4 Hard Negative Mining

Hard negative mining focuses on selecting negative examples that are most similar to the anchor, making the task of separating the anchor from these "hard" negatives more challenging. Such negatives prevent the model from becoming overconfident and push it to learn finer distinctions between truly similar examples. By training with these difficult examples, the model gains a sharper decision boundary and reduces confusion when embeddings are close to each other. Consequently, this increases the model's robustness and improves its overall performance.

3.4.1 Implementation

Inside the TripletDataset, for each sample, the code computes the anchor embedding and identifies the corresponding positive embedding (i.e., the correct misconception). It then calculates the similarity between the anchor embedding and every other misconception embedding, picking the most similar one as the hard negative. This means the chosen negative has the highest embedding similarity to the anchor yet is not the correct misconception. Finally, the triplet (anchor, positive, hard negative) is used in the triplet loss function to refine the model's embeddings.

3.5 Select Misconception Candidates

The selection of misconception candidates is a critical step in predicting the affinity between distractors and misconceptions. Various methods are employed to identify the most relevant misconception candidates based on the textual and contextual similarities between distractors and known misconceptions. Below, we detail two commonly used approaches: Cosine Similarity and K-Nearest Neighbors.

Cosine Similarity Cosine Similarity measures the cosine of the angle between two vectors in a multi-dimensional space, effectively quantifying the similarity between two textual representations. In this context, distractor and misconception texts are transformed into vector embeddings, often using techniques like TF-IDF, Word2Vec, or Sentence Transformers. High cosine similarity scores indicate that the distractor closely matches the misconception in terms of textual semantics, making it a useful metric for ranking candidate misconceptions.

K-Nearest Neighbors K-Nearest Neighbors (KNN) is a non-parametric method used to find the closest misconceptions to a given distractor based on vector distances. By embedding both distractors and misconceptions in the same feature space, KNN identifies the top-k most similar misconceptions using cosine distance. This method is particularly effective when combined with high-quality embeddings that capture semantic nuances. However, it can be computationally intensive when dealing with large datasets.

Based on our experiments, the results produced by KNN and Cosine Similarity exhibit subtle differences. These differences are mainly reflected in the order of multiple misconception pairs, where their rankings are swapped. For the top 100 candidates of each question, we observed that 1 to 15 pairs (resulting in 2 to 30 swaps) might differ between the two methods. While the overall performance of both methods remains consistent, these minor variations could impact subsequent re-ranking or model refinement steps.

Table 3: Performance Comparison of Candidate Selection Methods

Method	Public	Private	Average	Reference
Cosine Similarity	0.422	0.41856	0.4195	[9]
K-Nearest Neighbors	0.42187	0.41939	0.4201	-

3.6 Re-ranking

Re-ranking is a process used to refine the initial predictions of a model by re-ordering the candidate Misconception IDs based on additional criteria or more sophisticated methods. After generating an initial list of potential candidates, re-ranking aims to improve the alignment between the distractor and the most relevant misconception by leveraging supplementary features or advanced algorithms.

Re-ranking is particularly useful in scenarios where the initial ranking does not fully capture the nuanced relationships between distractors and misconceptions. For instance, candidates with similar textual content but different semantic meanings might be reordered based on deeper contextual understanding. Moreover, re-ranking can incorporate domain-specific knowledge, such as prioritizing misconceptions that are more prevalent in a particular subject area.

The effectiveness of re-ranking depends on the quality of the additional features and scoring mechanism. By incorporating re-ranking into the pipeline, the overall accuracy and precision of the model in identifying the correct misconceptions can be significantly enhanced.

Table 4 presents the experimental results of the first solution ranked on the leaderboard[10]. The solution employs a multi-phase re-ranking process using progressively larger rerankers. The models are described as follows:

Retrieval: Represents the baseline performance using only the initial retrieval model without re-ranking.

1st Stage Rerank (14B Reranker): Applies a 14B parameter re-ranker in the first stage, improving the initial retrieval results.

2nd Stage Rerank (32B Reranker): Further refines the re-ranked candidates using a 32B parameter re-ranker in the second stage.

3rd Stage Rerank (72B Reranker): Finalizes the re-ranking process with the largest model, a 72B parameter re-ranker, for the most precise alignment.

The retrieval-only model achieves an average score of 0.489, providing a baseline for comparison. With the 1st stage re-rank (14B re-ranker), the average score significantly improves to 0.614, demonstrating the effectiveness of the first re-ranking stage. The 2nd stage re-rank (32B re-ranker) further enhances performance, achieving an average score of 0.628, indicating that additional re-ranking stages refine the alignment between distractors and misconceptions. The 3rd stage re-rank (72B re-ranker) achieves the highest average score of 0.639, showing that the largest and most complex model delivers the most accurate results.

Table 4: Retrieval Model Performance Comparison

Model	Public	Private	Average
Retrieval	0.524	0.475	0.489
14B Ranker	0.611	0.615	0.614
32B Ranker	0.636	0.625	0.628
72B Ranker	0.643	0.638	0.639

4 Experimental Result

We calculate average score by weighted public score 28% and private score 72% [11]

4.1 Fine tuning

Table 5: Fine tuning Performance

Method	Trainable Parameters	Public	Private	Average
Without fine tuning	-	0.41678	0.42062	0.4195448
Fine tuning	0.2324%	-	-	-

4.2 Re-ranking

We first perform experiments comparing the use of a reranker versus no reranking, as shown in Table 6. The results highlight the impact of incorporating re-ranking into the pipeline. Without re-ranking, the model achieves an average score of 0.42076, serving as the baseline performance. In contrast, applying a single-phase re-ranking method with the Qwen 32B reranker significantly improves the average score to 0.45262.

On both the public and private test sets, the benefits of re-ranking are consistent.

This analysis underscores the effectiveness of the Qwen 32B reranker in refining initial retrieval results. By leveraging re-ranking, the model can better prioritize the most relevant candidates, leading to improved overall performance. These findings emphasize the importance of re-ranking as a crucial step in optimizing the pipeline for predicting the affinity between distractors and misconceptions.

Table 6: Re-ranking Method Performance Comparison

Method	Public	Private	Average
Without Re-ranking	0.41984	0.42112	0.42076
1 Phase Qwen 32B	0.47038	0.44572	0.45262

The experimental results in Table 7 to Table 10 present a detailed comparison of performance across different candidate sizes and temperatures during each re-ranking round.

For smaller candidate sizes (e.g., 3 and 4), the performance in terms of average scores is generally higher compared to larger candidate sizes (e.g., 9). This suggests that the model performs better

when the candidate pool is more focused, likely because it reduces the likelihood of introducing irrelevant options during re-ranking. For candidate sizes 3 and 4, the average performance peaks around specific configurations (e.g., Candidate = 39 for 3 choices, achieving an average of 0.46264, and Candidate = 40 for 4 choices, achieving 0.45862). However, as the candidate size increases, the average scores tend to decline.

We hypothesize that the decline in average scores with larger candidate sizes may be attributed to cases where the reranker fails to identify the ground truth. In such scenarios, the ground truth might be excluded from the top 25 re-ranked options due to the inclusion of too many candidates, leading to a drop in performance. Conversely, the observed increase in average scores with even larger candidate sizes could be explained by the inclusion of ground truth candidates that were previously excluded. This indicates that the balance between candidate size and reranker effectiveness is critical, as too many candidates can dilute performance, while too few may exclude essential options, including the ground truth.

The comparison between temperatures (0.7 and 0.0) shows minor differences in performance. While both settings yield similar trends, Temperature = 0.0 generally achieves slightly higher average scores across most configurations. This indicates that a deterministic sampling strategy (Temperature = 0.0) may provide more consistent and reliable results compared to a more stochastic approach (Temperature = 0.7).

For larger candidate sizes (e.g., 9), the average performance consistently declines. For example, with 9 candidates, the average scores drop to as low as 0.29750 (Candidate = 33, Temperature = 0.7). This is likely due to the increased difficulty in accurately ranking a larger pool of candidates, leading to a dilution of relevance among the top choices.

Some configurations, such as Candidate = 37 and Candidate = 25, show stable performance across different temperatures and candidate sizes. This suggests that certain candidate selections are inherently better aligned with the re-ranking algorithm, irrespective of hyperparameters.

Table 7: Performance Comparison for 3 Choices every Re-rank Round.

		Temperature=0.7			**Temperature=0.0**		
Candidate	**Option**	**Public**	**Private**	**Average**	**Public**	**Private**	**Average**
25	3	0.47541	0.44618	0.45436	0.47812	0.44834	0.45668
27	3	0.48371	0.45406	0.46236	0.48492	0.45460	0.46309
29	3	0.40604	0.38841	0.39335	0.40711	0.38914	0.39417
31	3	0.47720	0.44375	0.45312	0.47992	0.44667	0.45598
33	3	0.48545	0.45355	0.46248	0.48579	0.45365	0.46265
34	3	0.47793	0.44648	0.45529	0.47689	0.45437	0.46068
35	3	0.40762	0.38545	0.39166	0.40871	0.38610	0.39243
37	3	0.47881	0.44722	0.45607	0.48152	0.44970	0.45861
39	3	0.48703	0.45315	**0.46264**	0.48736	0.45365	**0.46309**

Table 8: Performance Comparison for 4 Choices every Re-rank Round.

		Temperature=0.7			**Temperature=0.0**		
Candidate	**Option**	**Public**	**Private**	**Average**	**Public**	**Private**	**Average**
25	4	0.47326	0.45008	0.45657	0.47427	0.45285	0.45885
28	4	0.47723	0.44838	0.45646	0.48019	0.44870	0.45752
31	4	0.39134	0.34806	0.36018	0.38810	0.34883	0.35983
34	4	0.41328	0.39284	0.39856	0.41045	0.39205	0.39720
37	4	0.47800	0.45272	**0.45980**	0.47674	0.45556	**0.46149**
40	4	0.48272	0.44925	0.45862	0.48438	0.44958	0.45932

Table 9: Performance Comparison for 5 Choices every Re-rank Round.

		Temperature=0.7			Temperature=0.0		
Candidate	**Option**	**Public**	**Private**	**Average**	**Public**	**Private**	**Average**
25	5	0.47158	0.44823	**0.45477**	0.47225	0.45087	**0.45686**
29	5	0.36683	0.33175	0.34157	0.36036	0.33023	0.33867
33	5	0.39202	0.34656	0.35929	0.39331	0.34590	0.35917
37	5	0.41701	0.39368	0.40021	0.41553	0.39539	0.40103
41	5	0.46888	0.44826	0.45403	0.39516	0.34766	0.36096

Table 10: Performance Comparison for 9 Choices every Re-rank Round.

		Temperature=0.7			Temperature=0.0		
Candidate	**Option**	**Public**	**Private**	**Average**	**Public**	**Private**	**Average**
25	9	0.47125	0.44323	**0.45108**	0.47038	0.44572	**0.45262**
33	9	0.31122	0.29216	0.29750	0.31428	0.29215	0.29835
41	9	0.32543	0.31168	0.31553	0.32543	0.31169	0.31554
49	9	0.34637	0.31348	0.32269	0.35071	0.31432	0.32451

Table 11: Performance Comparison for Different # Choices every Re-rank Round.

		Temperature=0.7			Temperature=0.0		
Candidate	**Option**	**Public**	**Private**	**Average**	**Public**	**Private**	**Average**
25	3	0.47541	0.44618	0.45436	0.47812	0.44834	0.45668
25	4	0.47326	0.45008	0.45657	0.47427	0.45285	0.45885
25	5	0.47158	0.44823	0.45477	0.47225	0.45087	0.45686
25	9	0.47125	0.44323	0.45108	0.47038	0.44572	0.45262

5 Conclusion

In this study, we conducted research focused on addressing the problem of aligning distractors with misconceptions in diagnostic questions. Our work emphasizes the crucial role of retrieval and reranking strategies in improving the overall system performance. Retrieval serves as a fundamental step for narrowing down the pool of potential candidates, ensuring that relevant misconceptions are included, while reranking further refines these candidates to prioritize the most accurate matches. Through this targeted approach, we aim to enhance the system's ability to identify and predict misconceptions effectively.

Our baseline model achieved a score of 0.45262. Through our experiments, we successfully improved this score to 0.46309, demonstrating the efficacy of our proposed refinements. The highest observed score in this domain is 0.67081, and while we were unable to reach this level, we believe that the insights gained from our experiments could be applied to more advanced pipelines to achieve even better performance. These findings highlight the potential of combining our discoveries with high-performing models to further push the boundaries of performance in this task.

Our experiments reveal that model complexity and the incorporation of reranking significantly impact performance. Larger models, such as Qwen 14B with a 32B reranker, consistently outperform smaller baselines, underscoring the role of scale and sophisticated ranking mechanisms in achieving state-of-the-art results. Additionally, the multi-phase reranking approach adopted by leading solutions highlights the cumulative benefits of iterative refinement in aligning distractors with misconceptions.

The study also highlights the challenges posed by imbalanced datasets and missing values, which can affect model training and evaluation. Addressing these challenges through strategies such as data augmentation, imputation, and balanced sampling is essential for building robust systems capable of generalizing across diverse question sets and misconception types.

Overall, this work demonstrates that combining retrieval and reranking with advanced models provides a powerful framework for improving the understanding and management of misconceptions in educational contexts. Future efforts could explore integrating additional contextual features,

leveraging semi-supervised learning for unlabeled data, or optimizing computational efficiency to further enhance the pipeline's scalability and applicability.

Work Load

Table 12 shows the work load of each member.

Table 12: Work Load of Each Member

Member	Work load
Guei-Chen, Wu	Organize, Sec. 3.3-4, 4.1
Chang-Shuo, Chen	Organize, Sec. 2.1-4, 3.2, 3.5-6, 4.2, 5
Shu-Rong, Yang	Data Augmentation survey and testing
Zi-Jun, Huang	

References

[1] The Learning Agency Lab. "Eedi - Mining Misconceptions in Mathematics". In: *Kaggle* (2024). Available at: https://www.kaggle.com/competitions/eedi-mining-misconceptions-in-mathematics.

[2] Tom B. Brown et al. "Language Models are Few-Shot Learners". In: *arXiv preprint arXiv:2005.14165* (2020).

[3] Long Ouyang et al. "Training language models to follow instructions with human feedback". In: *arXiv preprint arXiv:2203.02155* (2022).

[4] Pratham Batra. "Hybrid Search (Ensemble Encoder + BM25)". In: *Kaggle* (2024). Available at: https://www.kaggle.com/code/prtm1908/hybrid-search-ensemble-encoder-bm25.

[5] sayoulala. "Use LLM Embedding Recall Infer". In: *Kaggle* (2024). Available at: https://www.kaggle.com/code/sayoulala/use-llm-embedding-recall-infer.

[6] hoon0303. "Eedi Qwen32B vllm". In: *Kaggle* (2024). Available at: https://www.kaggle.com/code/hoon0303/eedi-qwen32b-vllm.

[7] Jagat Kiran. "Qwen14B_Retrieval_Qwen32B_logits-processor-zoo". In: *Kaggle* (2024). Available at: https://www.kaggle.com/code/jagatkiran/qwen14b-retrieval-qwen32b-logits-processor-zoo/notebook.

[8] Datawhale. "Open-Source Large Model Usage Guide". In: *Github* (2024). Available at: https://github.com/datawhalechina/self-llm.

[9] zuoyouzuo. "Eedi qwen2.5-14b-it a simple infer (LB: 0.422)". In: *Kaggle* (2024). Available at: https://www.kaggle.com/code/zuoyouzuo/eedi-qwen2-5-14b-it-a-simple-infer-lb-0-422.

[10] Raja Biswas. "1st Place Detailed Solution". In: *Kaggle* (2024). Available at: https://www.kaggle.com/competitions/eedi-mining-misconceptions-in-mathematics/discussion/551688.

[11] The Learning Agency Lab. "Eedi - Mining Misconceptions in Mathematics - Leader Board". In: *Kaggle* (2024). Available at: https://www.kaggle.com/competitions/eedi-mining-misconceptions-in-mathematics/leaderboard.