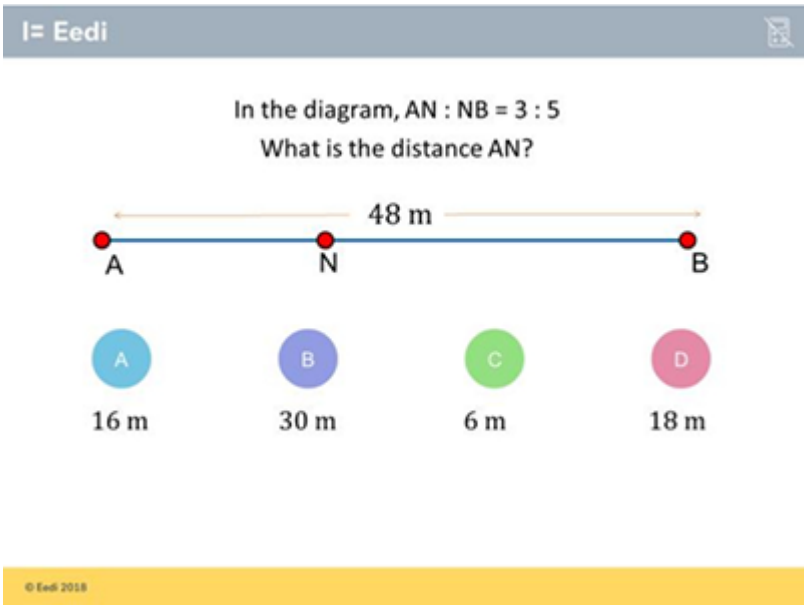
Mining Misconceptions in Mathematics

The "Eedi - Mining Misconceptions in Mathematics" competition on Kaggle challenges participants to develop a Natural Language Processing (NLP) model driven by Machine Learning (ML) techniques. The primary goal is to predict the affinity between mathematical misconceptions and incorrect answers (distractors) in multiple-choice questions (MCQs). This initiative aims to streamline the process of tagging distractors with appropriate misconceptions, thereby enhancing the quality and consistency of educational assessments.

Dataset Description

On Eedi, students answer Diagnostic Questions (DQs), which are multiple-choice questions featuring one correct answer and three incorrect answers, known as distractors. Each question targets a specific construct (also referred to as a skill), representing the most granular level of knowledge relevant to the question. Each distractor is designed to correspond with a potential misconception. Example of a Diagnostic Question:



In this example, the options for the question are labeled with misconceptions as follows:

- **A**: Divides total amount by each side of the ratio instead of dividing by the sum of the parts
- **B**: Mixes up sides of a ratio
- **C**: Finds one part of a ratio but doesn't multiply that by the number of parts needed
- **D**: Correct answer

The competition provides several datasets essential for training and evaluating the model:

- **train.csv**: Contains the training data with ground truth labels.
- **test.csv**: Contains the test data for which predictions are to be made.
- **misconception_mapping.csv**: Maps each *MisconceptionId* to its corresponding *MisconceptionName*.
- **sample_submission.csv**: Provides the correct format for the submission file.

The primary datasets (*train.csv* and *test.csv*) consist of multiple-choice questions with associated metadata. Each question is accompanied by four answer options: one correct answer and three distractors (incorrect answers). The datasets include the following fields:

- **QuestionId**: Unique identifier for each question (integer).
- **ConstructId**: Unique identifier for the construct or skill the question targets (integer).
- **ConstructName**: Description of the construct or skill (string).
- **CorrectAnswer**: The correct answer option, labeled as 'A', 'B', 'C', or 'D' (character).
- **SubjectId**: Unique identifier for the subject area (integer).
- **SubjectName**: Name of the subject area (string).
- **QuestionText**: The text of the question, extracted via human-in-the-loop Optical Character Recognition (OCR) (string).
- **AnswerAText**: Text of answer option A (string).
- **AnswerBText**: Text of answer option B (string).
- **AnswerCText**: Text of answer option C (string).
- **AnswerDText**: Text of answer option D (string).
- **Misconception[A/B/C/D]Id**: Unique identifier(s) for the misconception(s) associated with each distractor in *train.csv* (integer). These are the ground truth labels to be predicted for *test.csv*.

Participants are required to submit a CSV file (*submission.csv*) that contains predictions for each distractor in the test set. The expected structure is as follows:

- **QuestionId_Answer**: A unique identifier combining the *QuestionId* and the distractor label ('A', 'B', or 'C'). Since the correct answer ('D') does not require prediction, it is excluded.
- **MisconceptionId**: A space-delimited list of up to 25 predicted *MisconceptionId* values for each distractor.

Evaluation Method

The submissions are evaluated using the **Mean Average Precision at 25 (MAP@25)** metric. This metric assesses the quality of the predicted misconceptions by considering both the relevance and the rank of the predictions.

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

- U is the total number of observations (i.e., the total number of distractors to predict).
- n is the number of predictions submitted per observation (up to 25).
- P(k) is the precision at cutoff k, calculated as the proportion of relevant misconceptions in the top k predictions.
- rel(k) is a relevance function that equals 1 if the item at rank k is a correct label (i.e., the predicted *MisconceptionId* matches the ground truth), and 0 otherwise.

Important points:

- **Single Correct Label**: Each distractor is associated with only one correct *MisconceptionId*.
- **No Repetition of Correct Labels**: Once a correct label has been scored for an observation, additional predictions of that label are ignored in the calculation.
- **Ranking Matters**: Higher-ranked predictions contribute more to the MAP@25 score, emphasizing the importance of ordering the predictions based on confidence.

Example of Model Functionality

Input Line Example from *train.csv*:

QuestionId	ConstructId	Construct...	SubjectId	SubjectN...	CorrectAn...	QuestionT...
7	314	Divide decimals by 10	224	Multiplying and Dividing with Decimals	A	\(43.2 \div 10= \)

AnswerAT...	AnswerBT...	AnswerCT...	AnswerDT...	# Misconce...	# Misconce...	# Misconce...
\(4.32 \)	\(0.432 \)	\(33.2 \)	\(43.02 \)	2123.0	2273.0	2133.0

From the *misconception_mapping.csv*, we have:

- **MisconceptionId 2123**: "Divided by 100 rather than 10"
- **MisconceptionId 2273**: "Subtracts instead of divides"
- **MisconceptionId 2133**: "When dividing a decimal by a multiple of 10, just divides the fractional place values"

In this question, students are asked to divide 43.2 by 10. The correct answer is 4.32 (option A). The distractors (incorrect answers) are designed to reflect specific misconceptions. Our task is to predict the *MisconceptionId* for each distractor in the test

dataset. Since we are using an example from *train.csv*, we know the ground truth *MisconceptionId* values. Assuming that this question appeared in the test dataset, our model should generate a submission file (*submission.csv*) with predictions for each distractor (options B, C, and D).

Synthetic Dataset:

- 1.8k competition MCQs + 10.6k generated MCQs
- 4791 misconceptions and follows the same format as the original competition dataset.

SOTA Literature Review

Liu, N., Sonkar, S., Wang, Z., Woodhead, S., & Baraniuk, R. G. (2023). *Novice Learner and Expert Tutor: Evaluating Math Reasoning Abilities of Large Language Models with Misconceptions*. arXiv preprint arXiv:2310.02439.

The paper analyzes the ability of large language models (LLMs) to identify and handle mathematical misconceptions. Unlike traditional evaluations that measure accuracy in providing correct answers, this study introduces a testing method in which:

- LLMs are instructed to answer incorrectly according to a specific misconception (simulating a novice learner).
- LLMs must identify the misconceptions that led to an incorrect answer (simulating an expert tutor).

The paper uses an Eedi dataset containing grade-school-level math questions, each with four answer choices (one correct, three incorrect). Just like in the competition, each distractor is associated with a specific misconception.

To evaluate LLMs, two types of experiments were conducted:

1. **Simulating a novice learner** – Models had to select an incorrect answer based on a given misconception.
    - **Relevant observation:** The models (including GPT-4) performed significantly worse in this task compared to providing correct answers, indicating that recognizing systematic errors is challenging.
    -
2. **Simulating an expert tutor** – Models had to identify the misconception from a given set based on an incorrect answer.
    - **Relevant observation:** GPT-4 performed well when the number of misconceptions was small, but its accuracy significantly decreased as the

number of misconceptions increased (e.g., from **91.9% to 39.8%** when the set grew to **100 misconceptions**).

Relevant observations:

- They highlight the difficulty NLP models have in associating specific errors with misconceptions.
- They show that reducing the number of misconceptions per question can improve model performance.
- They suggest that advanced prompt engineering (e.g., "Chain-of-Thought" prompting) and few-shot learning methods can help improve accuracy.

Sadihin, B. C., Rodriguez Rodriguez, H., & Chen, M. J. (2023). *Mining Misconceptions in Mathematics*. OpenReview.net. Available at https://openreview.net/forum?id=CE85qdNSlp

This paper proposes a model to predict misconceptions associated with incorrect math answers in multiple-choice questions (MCQs). The goal is to automate **misconception mining**, which is a core challenge of the **Eedi - Mining Misconceptions in Mathematics** competition.

Key challenges:

- **High variability of misconceptions** makes classification difficult.
- **LLMs struggle to understand misconceptions** because they are optimized to provide correct answers rather than reason through errors.

To address these challenges, the paper suggests using an optimized LLM to analyze incorrect answers and match them to misconceptions through **vector embeddings.**

Proposed Approach

The study suggests the following techniques to improve misconception prediction:

1. **Vector Embedding Search for Misconception Retrieval**
    - Instead of feeding the entire misconception list (which may be too large for the LLM's context window), the model retrieves the most relevant misconceptions based on vector embeddings (e.g., cosine similarity, dense retrieval).

- This approach narrows down the list of candidate misconceptions, making LLM predictions more precise

2. **Multi-Vector Embedding Retrieval**

   - Incorrect answers often have high similarity in misconceptions (e.g., multiple mistakes in algebraic operations might relate to similar errors in order of operations).
   - The authors propose using multi-vector embedding retrieval (such as BGE-M3 Embedding) to leverage the similarity between incorrect answers and their misconceptions.
   - If multiple misconceptions are similar, retrieving multiple relevant embeddings instead of just one may improve ranking quality.

3. **Combine Fine-Tuning and In-Context Learning**

   - LLMs such as Phi-3.5 have strong zero-shot reasoning abilities but may struggle with misconception retrieval due to limited context size.
   - The authors propose fine-tuning and in-context learning to improve the model's ability to associate misconceptions with incorrect answers.
   - Chain-of-Thought prompting could further enhance reasoning.

4. **Balance the Dataset by Generating More Misconceptions**

   - For the misconceptions that are underrepresented, **synthetic misconception-aligned distractors** generated by LLMs may improve model generalization.

Williams, M., Taub-Tabib, H., Boyle, K., Reingold, M., Dhole, K., Munro, D., Chen, A. L., Piech, C., & Zhang, R. (2024). Exploring Automated Distractor Generation for Math Multiple-choice Questions via Large Language Models. arXiv preprint arXiv:2404.02124.

This paper focuses on automatically generating plausible distractors for math MCQs using Large Language Models (LLMs) which is very important for generating the synthetic data for our competition.

Core Insights:

- **LLMs Can Generate Valid Distractors – But Not Always Aligned with Real Misconceptions**

    - LLMs (e.g., GPT-4, ChatGPT) are good at producing mathematically valid distractors, but they struggle to reflect real student misconceptions.
- **Best Performing Approach: In-Context Learning with kNN Selection**
    - The kNN in-context learning method (using 3 nearest MCQs as prompts) outperformed all others.
    - These similar questions guide the LLM to generate distractors in the correct format and with similar misconception patterns.

Observations and Takeaways:

- **High similarity between in-context examples and target question** improves performance.
- Including explanations and feedback in prompts enhances distractor generation.
- Many human-labeled distractors may not reflect real student behavior, introducing label noise.
- **In-context learning with kNN** is highly effective, especially when the question stem, key, and explanation are included.
- LLM-generated distractors can be useful for enriching misconception coverage, especially for underrepresented topics.

Qin, Z., Jagerman, R., Hui, K., Zhuang, H., Wu, J., Yan, L., Shen, J., Liu, T., Liu, J., Metzler, D., Wang, X., & Bendersky, M. (2024). *Large Language Models are Effective Text Rankers with Pairwise Ranking Prompting*. arXiv preprint arXiv:2306.17563.

This paper introduces a method called **Pairwise Ranking Prompting (PRP)** that is highly relevant to the task of ranking misconceptions based on their affinity with incorrect answers in math questions.

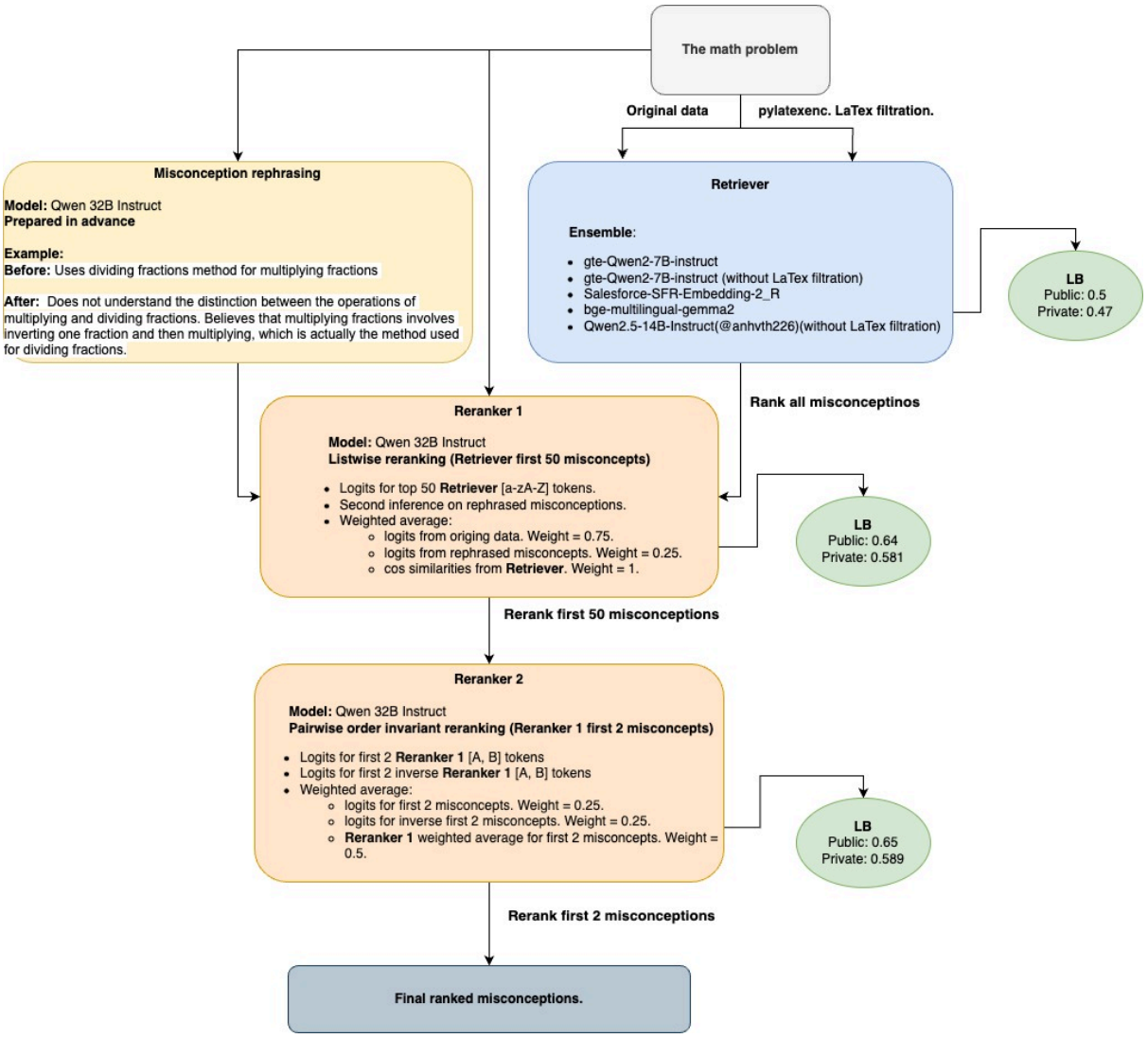
Observations:

- **Pairwise Ranking Prompting (PRP):**
    - Instead of comparing a list of items (listwise) or scoring one at a time (pointwise), PRP asks the LLM to choose between two items (e.g., “Which of these two misconceptions better fits this distractor?”).
    - This is simpler for LLMs and reduces generation failures compared to listwise methods.
- **The PRP-Sliding-K strategy:**
    - Perform K bubble-sort-like passes to optimize top-K ranking

Takeaways:

- **PRP** is an good approach because:
    - It allows ranking misconceptions relative to each other for a given distractor.
    - It's designed to be robust even without fine-tuning, perfect for settings with limited labeled data or where misconceptions may not have appeared in training.
    - PRP can integrate easily in pipelines that generate a shortlist of candidates using a retriever

Team Nikita Babych (2024). *6th Place Solution.* Available at https://www.kaggle.com/competitions/eedi-mining-misconceptions-in-mathematics/discussion/551565



This solution focused on generalization, robust validation, and a two-stage reranking system, enhanced with synthetic data and misconception rephrasing.

Validation Strategy

- Traditional *GroupKFold* by *QuestionId* led to overfitting on seen misconceptions.
- Instead, the author used a single-fold strategy with balanced seen/unseen misconceptions, mimicking the leaderboard setup.
- This yielded more reliable CV scores, preventing misleading local performance.

Synthetic Data Generation

- Used **Qwen 72B Instruct** for few-shot generation (3 examples × 3 seeds = 9 questions per misconception).
- Ensured generated math tests had only 1 distractor aligned to the misconception.
- **LoRA** pretraining degraded model reasoning, so it was avoided.

Misconception Rephrasing

- Misconceptions were reformulated using **Qwen 32B Instruct** into a unified format:
    - Combined missing knowledge and student belief.
- These rephrased versions improved generalization and acted as regularization.
- During inference, logits from both original and rephrased misconceptions were averaged.

Retriever Ensemble

- Used 5 retriever models (e.g., **gte-Qwen2-7B, Salesforce-SFR-Embedding-2_R**).
- Some versions used LaTeX filtration (via **pylatexenc**), others didn't, ensuring robustness.

Reranking Pipeline

- Reranker 1 (**Qwen 32B Instruct**)
    - Listwise reranking over top 50 misconceptions from the retriever.
    - Used token-based logits (a-z, A-Z).
    - Logits were averaged from:
        - Original data (weight 0.75)
        - Rephrased misconceptions (0.25)
        - Retriever's cosine similarity (1.0)

- Reranker 2 (**Qwen 32B Instruct**)
    - After analyzing the logits obtained from the first-stage reranker, the author noticed that Qwen tends to preserve the initial order of misconceptions as sorted by the retriever. This made him retrain Retrievers with more focus on **MRR (Mean Reciprocal Rank)** and develop the second-stage reranker to refine the results of the first stage reranker.
    - In the second stage, the author selected a small window of the top ranked misconceptions from the first stage reranker and applied all possible permutations to their order. Then he averaged the logits across these permutations and reranked misconceptions from that window again. (optimal window size was 2)
    - **Pairwise reranking** of the **top 2 misconceptions**.
    - Applied **order-invariant inference** by permuting top 2 options and averaging logits.
    - Final reranking used a weighted average of:
        - Direct logits (0.25)
        - Inverse logits (0.25)
        - Reranker 1's average logits (0.5)

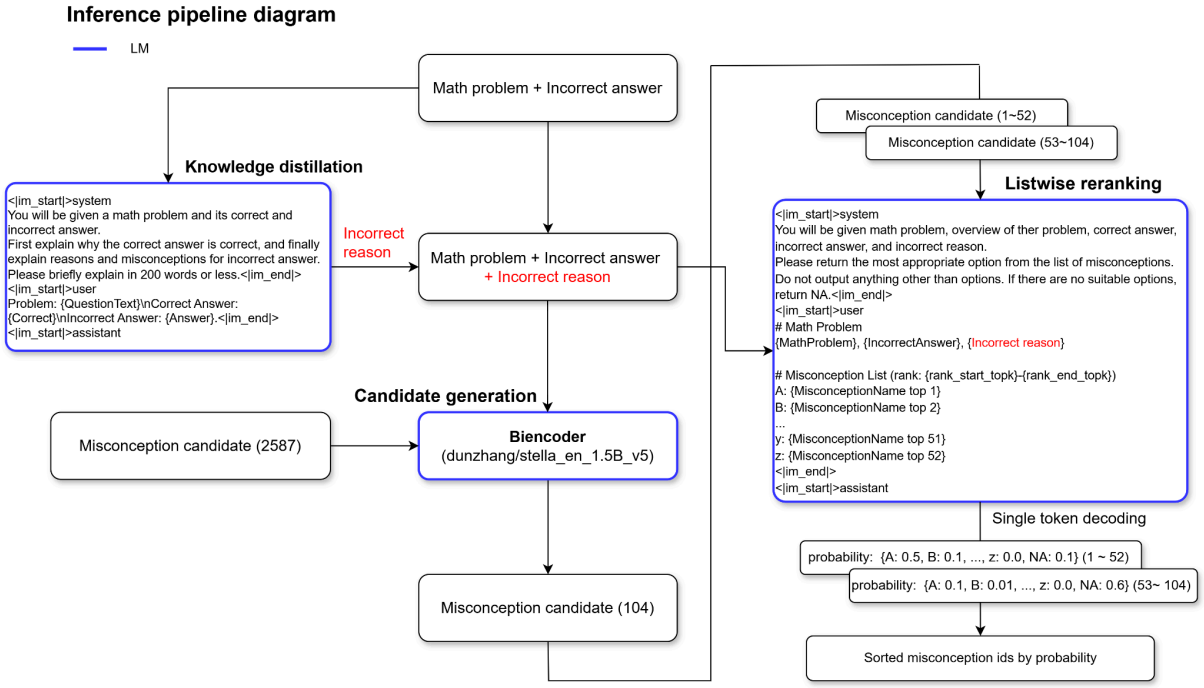
Distillation

- Retrained a model using **KL divergence** on logits generated with a different seed.
- Prevented overfitting to a single correct misconception, capturing ambiguity.
- Boosted public LB by ~0.01

What didn't work

- Using **LoRA adapters** for rerankers (fp16 + quantization failed).
- Training **Qwen 72B** as a reranker gave no benefit over 32B.
- Using synthetic data in reranker training didn't help.

Team Ebi & ktr (2024). *5th Place Solution.* Available at https://www.kaggle.com/competitions/eedi-mining-misconceptions-in-mathematics/discussion/551391



This solution ranked 5th in the competition and used a combination of synthetic data generation, knowledge distillation, biencoder models, listwise reranking, and ensembling to improve misconception prediction accuracy.

Synthetic Data Generation

- Since some *MisconceptionIds* were missing from the training dataset, the team used **LLMs (Gemini 1.5 Pro)** to generate synthetic problems.
- They applied **few-shot prompting** (4 examples) to generate questions and incorrect answers aligned with missing misconceptions.
- **Vector embeddings** were used to find similar misconceptions and include relevant problems.
- +0.01 MAP@25 improvement over random synthetic data.

Knowledge Distillation

- Used **Qwen 2.5 32B Instruct** to generate explanations for incorrect answers.
- The generated explanations were used as input for the **biencoder model** and **listwise reranker**.
- Improved biencoder's cross-validation (cv) score by +0.04 and reranker's cv by +0.01.

Candidate Generation

- A **biencoder model (dunzhang/stella_en_1.5B_v5)** was used to narrow down misconception candidates. (104 candidates)
- Negative mining was used, but hard negatives did not improve performance.

Listwise Reranking

- Using LLMs for reranking misconception candidates.
- The top 104 candidates from the biencoder were split into two groups (52 each) for inference.
- **LLM (Qwen 2.5 32B Instruct)** was fine-tuned to predict the best misconception choice
- **Single-token decoding** was used, meaning the model only outputted a **single letter (A-Z, a-z)** corresponding to a misconception.
- The approach was inspired by **token-based sorting of logits** (reference: arXiv:2406.15657).

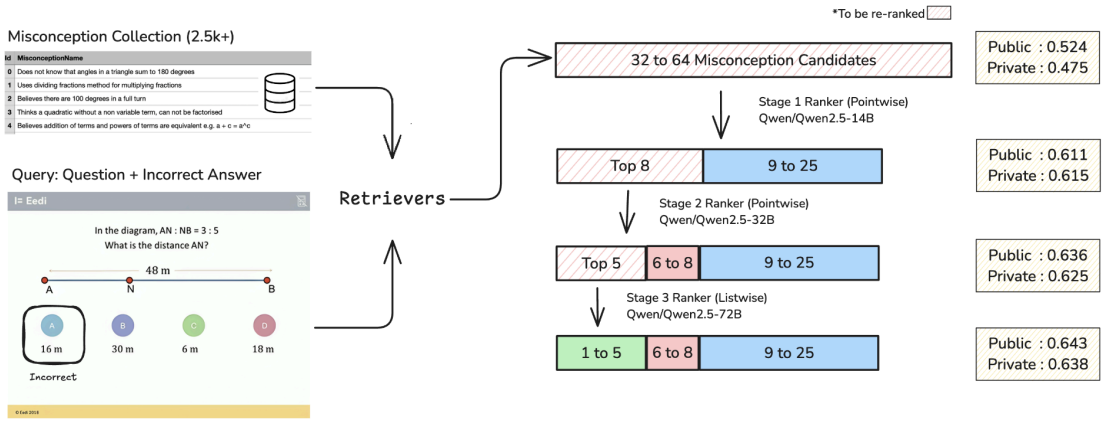
Ensembling

- Three-fold ensembling was applied to improve robustness.

Cross-Validation Strategy

- Initially, *GroupKFold(k=5)* using *QuestionId* was used, but the local CV was much higher than the leaderboard score, suspecting that there might be many *MisconceptionIds* that only appear in the test data.
- They switched to *GroupKFold(k=5)* using *SubjectId*, which resulted in a higher proportion of *MisconceptionIds* that only appear in validation data. This tended to make the CV closer to the leaderboard score.

Team Raja Biswas (2024). *1st Place Solution.* Available at https://www.kaggle.com/competitions/eedi-mining-misconceptions-in-mathematics/discussion/551688



Key Ideas

- Synthetic data is essential for improving generalization to unseen misconceptions.
- High-quality synthetic data is feasible in math because correctness can be verified objectively.
- Distilling reasoning (especially CoT) from large LLMs like **Claude 3.5** can boost performance on harder cases.
- Quantization and calibration methods can impact performance.
- Full fine-tuning may outperform LoRA in some settings.
- Smart validation split (*GroupKFold* by *ConstructId*) to avoid leaderboard overfitting
- Optimization based on both recall and ranking quality

Pipeline Overview

- Retriever: Selects 32–64 misconception candidates per question + incorrect answer.
- Stage 1 Reranker (**Qwen2.5-14B**): Pointwise ranking, selects top 8.
- Stage 2 Reranker (**Qwen2.5-32B**): Pointwise ranking on top 8 → keeps top 5.
- Stage 3 Reranker (**Qwen2.5-72B**): Listwise ranking on top 5 (includes CoT and examples).

Training Data

- 1.8k original (competition) questions
- 10.6k synthetic questions generated using LLMs (Claude 3.5 + GPT-4o filtering)
- Final pool: ~4,791 misconceptions (original + external, carefully cleaned)

Synthetic Data Generation

- Created clusters of similar misconceptions using co-occurrence stats from retriever/ranker outputs.
- Generated 5 new questions per cluster with Claude 3.5 and 4–5 reference examples per prompt.
- Used GPT-4o as a judge to rate the quality of synthetic questions and filter out bad ones.
- Kept new misconceptions only if they weren't duplicates (similarity < 0.95 via embedding scores).

Distillation & Chain of Thought

- Used Claude 3.5 to generate reasoning chains explaining why a student might choose the incorrect answer.
- These CoTs were added to the training data (50% of examples had CoTs).

Retriever Tuning

- Multiple retrievers trained with LoRA (r=64, alpha=128), optimized for Recall@32.
- *temperature = 0.01* instead of 0.02
- One example per misconception per batch
- Pretrained on all synthetic data (even before filtering)

Reranker Contributions

Ablation (Qwen/Qwen2.5-14B)

Exp	Model	CV	LB (Public)	LB (Private)
1	Qwen/Qwen2.5-14B (Baseline)	0.555	0.545	0.495
2	Qwen/Qwen2.5-14B (Baseline + Few Shot)	0.580	0.559	0.531
3	Qwen/Qwen2.5-14B (Baseline + Few Shot + Distillation)	0.626	0.601	0.575
4	Qwen/Qwen2.5-14B (Baseline + Few Shot + Distillation + Negative Ratio + Extra Data)	0.642	0.608	0.596
5	Qwen/Qwen2.5-14B (Baseline + Few Shot + distillation + Negative Ratio + Extra Data + CoT)	**0.646**	**0.611**	**0.615**

What didn't work

- Iterative hard negative mining
- Cross-device batch size increase
- Qwen/QwQ-32B-Preview performed worse despite fine-tuning
- Mergekit-style continuous fine-tuning strategies