kaggle.com

2 Meta Thoughts

At the start of the competition, I had the following hypotheses on what could be important for a good solution:

- Synthetic data should help with out-of-distribution generalization
- High quality synthetic data generation should be feasible since Math domain can be verified and curated objectively
- Distillation from a good teacher(s) should be very powerful, as Large LLMs are significantly better at reasoning
- Distilling Chain of Thoughts (CoT) from top LLMs (e.g. Claude 3.5 Sonnet) should help to tackle difficult examples
- Quantization method and the calibration datasets could be important. We may need to add recovery adapters to the quantized models for maintaining top 1 accuracy.
- Full training might be better than LoRA

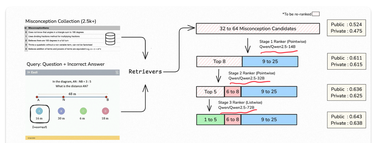
These hypotheses guided me to design my experiments throughout the competition and helped me to stick to an idea even though the initial attempts were not successful.

3 Pipeline

I implemented a retrieve-and-rerank system that involved a sequence of four steps:

- **Retrieval**: Identify top misconception candidates for each of the MCQ-Incorrect answer combinations. Specifically, I retained the top 32 retrieved candidates and included up to 32 additional candidates based on a dynamic threshold. The dynamic threshold is computed as the `top_candidate_similarity_score - constant`. Candidates with similarity scores higher than the dynamic threshold are kept.
- **14B Ranker**: Use a fine-tuned `Qwen/Qwen2.5-14B` ranker to process all retrieved candidates and shortlist top 8.
- **32B Ranker**: Use a fine-tuned `Qwen/Qwen2.5-32B` ranker to process top 8 candidates and narrow them down to top 5.
- **72B Ranker**: Use the fine-tuned `Qwen/Qwen2.5-72B` ranker to finalize the ranking of top 5 candidates.

A schematic diagram of the pipeline and the leaderboard scores from each stages are shown below:



The final 25 misconception predictions comprised of:

- 1st to 5th candidate from the 72B ranker
- 6th to 8th candidate from the 32B ranker
- 9th to 25th candidate from the 14B ranker.

4 Data

The training data mix for both the retriever and ranker models included competition data (1.8k examples) and synthetic data (10k examples). The synthetic data played a crucial role in improving both raw performance and generalization capability with respect to unseen misconceptions.

4.1 Synthetic Data

In an ideal scenario, synthetic MCQs should have the following properties:

- The designated correct answers are indeed the correct answers (high accuracy).
- The incorrect answers directly stem from the target misconception (high diagnostic power).
- Among all misconceptions in the misconception pool (2.5k+), the indicated misconception should provide the most precise explanation for a given incorrect answer (high resolution).
- The generated MCQs should cover a broad range of subjects and constructs (high diversity).

I started generating diagnostic MCQs with a simple prompt that specified the expected format and a misconception of interest. However, it resulted in a dataset that violated almost all the properties to varying degrees. It was expected since

- The simple prompt only included information about a single misconception. Hence, the data-generating LLM was completely unaware of other related misconceptions within the misconception pool.
- Lack of few shot examples meant lack of reference on the expected quality, difficulty level, tone, and language.
- Even the advanced LLMs struggle with handling misconceptions and counterfactual reasoning effectively.
- No curation was performed.

I experimented with several strategies to address the issues and managed to create a high quality synthetic dataset with a grounded synthetic data generation approach and incorporating LLM based filtering. Specifically, I first created clusters of closely related misconceptions leveraging the co-occurrence statistics of misconceptions in retriever/ranker predictions on validation data. This notebook demonstrates the clustering process. Here is an example of a misconception cluster:

```
- Thinks a*2 - a*2x is a valid form of difference of two squares
- When factorising a quadratic without a non variable term, forgets to double bracket factorise
- Incorrectly factorises a quadratic
- Believes the constant in an expanded quadratic comes from adding the two numbers in the brackets
- Believes the coefficient of x in an expanded quadratic comes from multiplying the two numbers in the brackets
- Does not recognise difference of two squares
- Does not realise a quadratic must be in the form ax^2+bx+c=0 to be factorised
- Does not know how to identify common factors from algebraic terms
- Believes the difference of 2 squares method of factorising also works for the addition of 2 squares
- When factorising, believes they can choose different factors for each term
- Believes they can factorise a difference of two squares by placing the constant in both brackets without square rooting
```

I next generated 5 new MCQs for each cluster using Claude 3.5 Sonnet with a refined prompt that included 4-5 reference examples on misconceptions from the cluster. I used the metaprompt kit from Anthropic cookbook for prompt engineering. Here is the final prompt used for data generation:

```
You will be generating Multiple Choice Questions (MCQs) that diagnose specific mathematical misconceptions. Here are the misconceptions you should focus on:

<misconceptions>
{cluster_misconceptions}
</misconceptions>

Here are reference MCQs that demonstrate how to effectively diagnose these misconceptions:

<reference_mcqs>
{reference_mcqs}
</reference_mcqs>

Your task is to generate {num_mcqs} new MCQs that diagnose misconceptions not already covered by the reference MCQs.

First, analyze the reference MCQs carefully:
1. For each reference MCQ, identify in your <analysis> tags:
   - Which misconception it targets
   - How the incorrect answers map to specific misconceptions
   - What makes the question effective at diagnosing the misconception
2. Note the style, difficulty level, and precision of language used

Then, in your <planning> tags:
- List which misconceptions still need coverage
- For each needed misconception, brainstorm mathematical contexts where it commonly appears
- Design questions where the misconception leads naturally to specific wrong answers
- Take notes on how you can craft new MCQs that adheres to the reference MCQs' style, difficulty level, and precision of language

Finally, generate new MCQs following these important guidelines:
- Make sure each incorrect answer maps clearly to exactly one misconception
- Use precise mathematical language matching the style of reference MCQs
- Make questions challenging enough that students must demonstrate real understanding
- Ensure wrong answers are plausible and stem from genuine misconceptions, not careless errors
- Use the exact wording of misconceptions as given in the misconceptions list
- Pay careful attention to subtle differences between the misconceptions and observe which one is the most appropriate for a given incorrect answer
- Keep the construct name and subject name as short as possible hiding the details of the misconception
- Questions should be of higher difficulty level than reference MCQs
```

4.1.1 Curation

I used LLM-as-a-judge to filter out low quality synthetic MCQs. I prompted `GPT-4o` to rate the quality of the synthetic MCQs as below:

```
You will analyze how well an incorrect answer reflects a suspected misconception in a mathematics problem. Your goal is to determine whether there is a clear, logical connection between the misconception and the wrong answer.

Here is the problem with both correct and incorrect answers. The suspected misconception is also provided:
<problem>
{PROBLEM_DATA}
</problem>

First, analyze the problem in your scratchpad:
<scratchpad>
1. Solve the problem independently to verify the correct answer
2. Examine how someone holding the suspected misconception would approach the problem
3. Trace the logical path from misconception to incorrect answer
4. Identify any gaps or inconsistencies in this connection
</scratchpad>

Then provide your evaluation using this format:
<evaluation>
1. Brief explanation of how the misconception could lead to the wrong answer
2. Score from 0-10 based on these criteria:
   - 10: Perfect alignment - wrong answer is direct result of misconception
   - 8-9: Strong alignment - clear logical path from misconception to answer
   - 5-7: Moderate alignment - connection exists but has some gaps
   - 1-4: Weak alignment - connection is unclear or requires assumptions
   - 0: No alignment - misconception does not explain wrong answer
</evaluation>

Important guidelines:
- Focus solely on the logical connection between misconception and wrong answer
- Do not speculate about other possible misconceptions
- Be specific about how the misconception leads to the error
- Flag and deduct scores if any assumptions are required to connect misconception to answer
- Consider whether a student with this misconception would consistently arrive at this wrong answer
```

I selected `GPT-4o` as judge based on vibe test as it felt more sensitive to the scoring scheme and as consistent as Claude 3.5 Sonnet.

4.1.2 Incorporating Additional Misconceptions

While creating synthetic data, LLM produced many misconceptions (`4k+`) that were not part of the host provided misconception pool. I decided to keep the external misconceptions to improve generalization capability of the retriever and ranker models.

Naively merging the external misconceptions with existing ones would have introduced conflicts and noise, since many of them were just rephrasing of the existing misconceptions. I adopted the following steps to handle this carefully:

- Attempt to match with existing misconceptions using strong normalization (lowercasing, removing trailing punctuation, etc.)
- Attempt to match with existing misconceptions using similarity scores from an embedding model (`thenlper/gte-base`) with very high threshold (0.995)
- Remove new misconceptions that have fairly high similarity (between 0.95 and 0.995) with any of the existing misconceptions
- Keep rest of the remaining external misconceptions

4.1.3 Dataset

The final dataset is uploaded here: Eedi - Mining Misconceptions in Mathematics. It contains

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

Reference: I'd highly recommend reading the synthetic data generation post from answer.ai: How To Synthesize Your Data.

4.2 Chain of Thought (CoT) Generation

To distill the reasoning capability of an advanced LLM in my solution pipeline, I first created a dataset containing thought processes generated by `Claude 3.5 Sonnet`. I prompted Claude with: (i) the problem statement, (ii) the correct answer, (iii) an incorrect answer, (iv) the ground truth misconception, and (v) several closely related misconceptions. Using this information, I prompted Claude to generate the likely thought process that led a student to select the incorrect answer. The prompt is shown below:

```
You will analyze a student's incorrect answer to identify the specific reasoning flaw that led to their error.
Your goal is to explain precisely how their misconception caused them to arrive at the wrong answer.

Here is the problem information:
<problem_data>
# Question: Simplify the expression: \[x \cdot y \cdot x\]
# Correct Answer: \(x^2y\)
# Incorrect Answer: \(x^3\)
# Correct Misconception: Ignores variables without explicit coefficients when multiplying
</problem_data>

Here are related misconceptions that are similar but do not explain this specific error as precisely:
<related_misconceptions>
- Thinks only like terms can be multiplied
- Fails to combine all instances of the same variable
- Incorrectly identifies an incomplete variable factor
- Does not understand how to multiply algebraic terms
</related_misconceptions>

First, examine all components of the problem carefully:
1. The problem statement and question asked
2. The correct answer and solution method
3. The student's incorrect answer
4. The primary misconception given
5. The related misconceptions that should be distinguished from the primary one

Then, reconstruct the student's likely thought process:
- Identify the exact point where their reasoning diverged from the correct solution path
- Note which specific mathematical operations or concepts they misapplied
- Connect their error directly to the stated primary misconception
- Verify that this explanation better fits the error than the related misconceptions

Write your analysis in <evaluation> tags, following this structure:
- Show the correct calculation first
- Show the incorrect calculations that demonstrate the error
- Explain the specific flaw in the student's reasoning
- Demonstrate how the misconception led to this particular error
- Distinguish from the related misconceptions
- Keep your explanation to 5-6 clear, non-repetitive sentences
- Focus solely on the reasoning that produced this specific error

Guidelines for writing your explanation:
- Do not restate the problem or name the misconception
- Be precise about the mathematical concepts involved
- Show exactly how the misconception led to the error
- Distinguish from related misconceptions
- Avoid repetition
- Stay focused on this specific error
```

Using this dataset, I fine-tuned three Qwen 2.5 series models (`Qwen/Qwen2.5-Math-7B`, `Qwen/Qwen2.5-14B`, and `Qwen/Qwen2.5-32B`) to act as reasoners. Notably, I omitted the true and related misconceptions from the input for fine-tuning. During inference, these models would see: (i) the problem statement, (ii) the correct answer, and (iii) an incorrect answer. They would then generate the likely thought process of a student behind selecting the incorrect answer. I used `vllm` for CoT generation during inference with `temperature=0.7, top_p=0.8, repetition_penalty=1.0, max_tokens=256`.

The CoT dataset is uploaded here: Eedi - CoT Dataset.

5 Train-Validation Split

I split the competition dataset into 5 folds, initially based on the `QuestionId` field. However, this produced overly optimistic estimates with a large gap between validation and leaderboard scores. Next, I tried `GroupKFold` on `SubjectId`, but this resulted in overly pessimistic estimates. I finally settled on `GroupKFold` with `ConstructId`, which yielded a narrow Validation/LB gap and good correlation up to ~0.62 public LB score.

During development, I always trained my models on folds 1-4 and validated on fold 0. The synthetic data was marked as fold 99 and was included in the training set. The final models were trained on the entire dataset (full fit) using the same hyper parameters from the development stage.

6 Retrievers

I used the standard `MultipleNegativesRankingLoss` to fine-tune the retrievers, while referencing the FlagEmbedding codebase for developing training scripts. At this stage, I tracked both `mapk25` and `recall@32` metrics and prioritized optimizing for `recall@32`, since recall was more important for overall pipeline performance.

I fine-tuned several retrievers with the following scores:

		Validation		Leaderboard	
Model	Backbone	MAP@25	Recall@32	LB (Public)	LB (Private)
1	intfloat/e5-mistral-7b-instruct	0.476	0.921	0.466	0.417
2	BAAI/bge-en-icl	0.493	0.909	0.463	0.419
3	Qwen/Qwen2.5-14B	0.486	**0.937**	0.449	0.451
4	Qwen/Qwen2.5-14B (re-ranker distillation)	**0.528**	0.926	**0.493**	**0.456**

Incorporating hard negatives into training batches and distilling re-ranker scores consistently improved `mapk25` performance, but did not positively impact `recall@32`. For my final submission selections, I chose the high-recall `Qwen/Qwen2.5-14B` encoder (Model 3) instead of the best `mapk25` encoder (Model 4). Following the same logic, I also excluded the BGE encoder (Model 2). Notably, my best encoder-only submission (Model 1 + Model 2 + Model 4) scored 0.524 on public LB and 0.475 on private LB.

Retrievers were fine-tuned using LoRA with `r=64, alpha=128, learning_rate_lora_a=1e-5, learning_rate_lora_b=5e-5, LoRA on all linear layers, batch_size=128, and epochs=12`. A few key factors that improved recall performance were:

- Setting temperature to 0.01 instead of the typically used value of 0.02 in LLM-based encoders.
- Ensuring only one demonstration per misconception appeared in each training batch, as multiple demonstrations would introduce label noise through in-batch negatives
- Pretraining the `Qwen/Qwen2.5-14B` encoder with all available synthetic data (before the curation step in section 4.1.1).

What didn't work:

- Iterative hard negative mining
- Increasing batch size through cross-device negatives
- Custom batching strategies e.g. having (query, misconception) positive pairs from the same Subjectid in the same batch
- My attempts at converting LLM retrievers to bi-directional encoders similar the the strategy used in nvidia/NV-Embed-v2.

6 Re-rankers

The Qwen 14B ranker did most of the heavy lifting by processing all retrieved misconceptions (32-64 candidates) and identifying top 8 candidates. This model was trained with a pointwise approach, where it sees one misconception at a time in the context window. The model input was structured as follows:

```
<|im_start|>system
You are an expert in detecting grade-school level math misconceptions. Verify if the incorrect answer stems from the provided misconception.<|im_end|>
<|im_start|>user
Misconception: Believes they can round when converting units

Demos for the misconception:
Question: 1.2? (mathematics)<br>Square \m^3=100 \)
Answer:\( 0.27 \)
Misconception Answer: \( 0.3 \)

Subject: Length Units
Question: Convert between m and cm
Question: To convert 3.5 metres to centimetres, you should:
Correct Answer: Multiply by 100
Incorrect Answer: Divide by 100

Thought: The student incorrectly applied the conversion factor by dividing instead of multiplying when converting from a larger unit (metres) to a smaller unit (centimetres). This reversal indicates a fundamental misunderstanding of the relationship between units, where they failed to recognize that moving from a larger unit to a smaller unit requires multiplication by the conversion factor (100). Their error stems from not understanding that when converting to a smaller unit, you need more of these units to represent the same quantity, which requires multiplication rather than division.

Does the misconception (Believes they can round when converting units) lead to the incorrect answer? Yes/No<|im_end|>
<|im_start|>assistant
```

Fine-tuning

Few Shot Example

Question
Correct & Incorrect Answers

CoT

The logits were computed by taking the difference between the 'Yes' and 'No' token scores, as below:

```
outputs = self.model(input_ids=input_ids, attention_mask=attention_mask, output_hidden_states=True)
logits_yes = outputs.logits[:, -1, self.yes_loc]  # Yes token logit at the last position [bs]
logits_no = outputs.logits[:, -1, self.no_loc]  # No token logit at the last position [bs]
logits = logits_yes - logits_no  # [bs]

logits = logits.reshape(-1, self.group_size)
labels = labels.to(logits.device).reshape(-1)
ce_loss = self.loss_fn(logits, labels)
```

Each batch contained 1 positive and `batch_size-1` negative misconceptions for a given Question-Incorrect Answer combination. The label was set to the index of the positive misconception and cross entropy loss was used for training. The models were trained with LoRA with `r=64, alpha=128, LoRA on all linear layers`. The language modelling head (`lm_head`) of the LLM (`Qwen/Qwen2.5-14B`) was frozen during fine-tuning.

What didn't work:

- Iterative hard negative mining
- Increasing batch size through cross-device negatives
- Custom batching strategies e.g. having (query, misconception) positive pairs from the same Subjectid in the same batch
- My attempts at converting LLM retrievers to bi-directional encoders similar the the strategy used in nvidia/NV-Embed-v2.

6.1 Ablations

I found the following 4 strategies to be particularly significant for boosting the reranker performance and its generalization capability:

6.1.1 Few Shot Examples

To retain and leverage the LLM's in-context learning capabilities, I optionally included a few reference misconception examples (1 to 3) in the model input context (as shown in the diagram above). Not all training examples contained these few-shot demonstrations. The objective was to encourage the model to use reference examples when available, and otherwise rely on its internal reasoning. During inference, I used 1 or 2 examples from the entire training set as demonstrations. This approach was particularly effective for the 14B ranker's performance boosting private LB score from 0.495 to 0.531 (**+0.036**). At this point, I was only using the competition data for training.

6.1.2 Distillation | Pseudo Labelling

Next, I incorporated the synthetic examples (post curation) into the training pipeline through pseudo labelling. I first fine-tuned two pointwise 7B models (`Qwen/Qwen2.5-Math-72B` and `Qwen/Qwen2.5-72B`) using only the competition dataset. These models were then used to generate pseudo labels for the synthetic examples. Finally, I used competition + pseudo-labeled data to fine-tune the next iteration of the 14B rankers. This strategy boosted the 14B reranker's private LB score from 0.531 to 0.575 (**+0.044**).

6.1.3 Negative Ratio

Next, I experimented with increasing the number of negative samples per positive. Showing the rankers a large number of negatives during training helped to improve performance. For each positive, I next increased the number of negatives to 24 from earlier 16. At this point, I also added scaled number of synthetic examples 2x. These two modifications boosted the 14B reranker's private LB score from 0.575 to 0.596 (**+0.021**).

6.1.4 Chain of Thought (CoT)

I finally added the fine-tuned CoT reasoners (Section 4.2) into training and inference pipeline. During training, CoTs produced from fine-tuned `Qwen/Qwen2.5-14B` reasoner were optionally added to the model input. 50% of training data used CoTs and 50% did not. This encouraged the model to use external reasoning when available and otherwise rely on its internal reasoning. CoT boosted the 14B reranker's private LB score from 0.596 to 0.615 (**+0.019**).

The impact of these strategies is summarized in the following table:

Ablation (Qwen/Qwen2.5-14B)

Exp	Model	CV	LB (Public)	LB (Private)
1	Qwen/Qwen2.5-14B (Baseline)	0.955	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)	0.628	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**

The live eval during fine-tuning showed consistent improvement in the 14B reranker's performance.

6.2 Pointwise Re-Ranker (`Qwen/Qwen2.5-32B`)

I fine-tuned the `Qwen/Qwen2.5-32B` model in exactly the same way as the 14B reranker (with a different seed). My best 32B model had validation score of `0.663` vs that of `0.646` for the 14B reranker. During inference, the 32B reranker was used to process top 8 candidates from the 14B reranker and narrow them down to top 5 candidates. Incorporating the 32B reranker boosted private LB score from 0.615 to 0.625 (**+0.010**).

6.3 Listwise Re-Ranker (`Qwen/Qwen2.5-72B`)

The top 5 candidates were finally ranked in a listwise manner using a fine-tuned `Qwen/Qwen2.5-72B` model. Here, the model sees all of the top 5 candidates at once. An example model input is shown below:

```
<|im_start|>system
You are an expert in detecting grade-school level math misconceptions. Rank 5 misconceptions from most to least relevant.<|im_end|>
<|im_start|>user
Rank 5 misconceptions for this incorrect answer:
Question: Calculate \( \frac{1}{2} \times \frac{3}{4} \)
Correct Answer: \( \frac{3}{8} \)
Incorrect Answer: \( \frac{3}{4} \)
Subject: Multiplying Fractions
Construct: Multiply two fractions

Misconceptions:
1. Does not know that to find a fraction of another fraction, you should multiply the fractions
2. Does not know how to find a fraction of another fraction
3. Does not know how to find the lowest common multiple of two numbers
4. Does not know that you should look for common factors in the numerator and denominator when cancelling
5. Does not know that you should look for common factors when cancelling
```