Eedi - Mining Misconceptions in Mathematics

An NLP-Driven Kaggle Competition to Enhance Educational Assessments

Competition Overview

Objective:

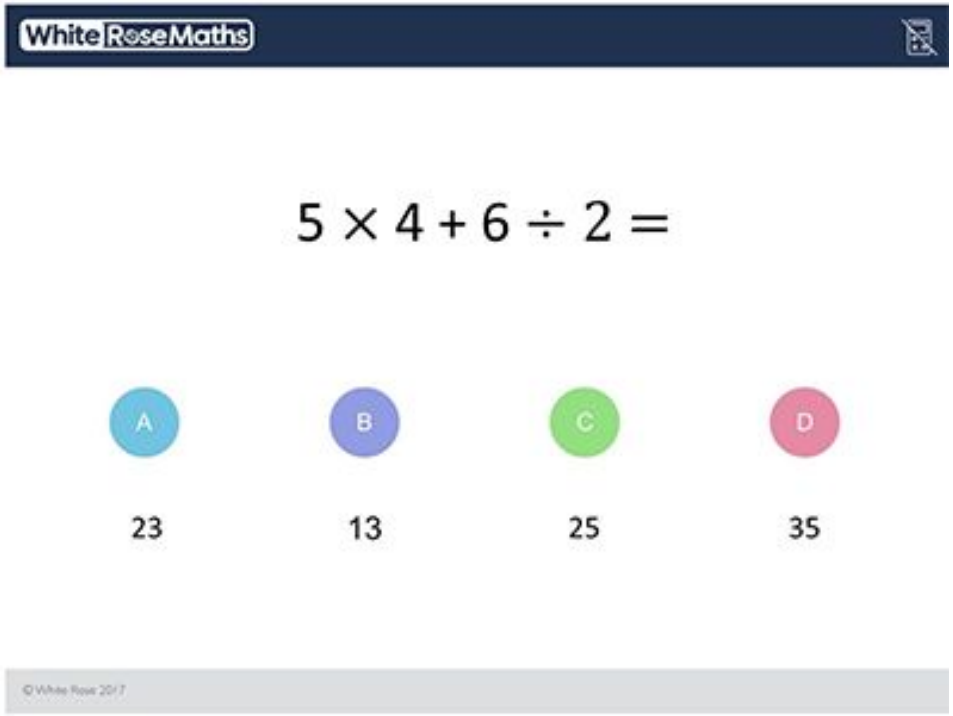
- Develop an NLP model using ML techniques.
- Predict the link between misconceptions and incorrect answers in MCQs.

Goal:

- Improve tagging of distractors to misconceptions for better educational assessment quality.

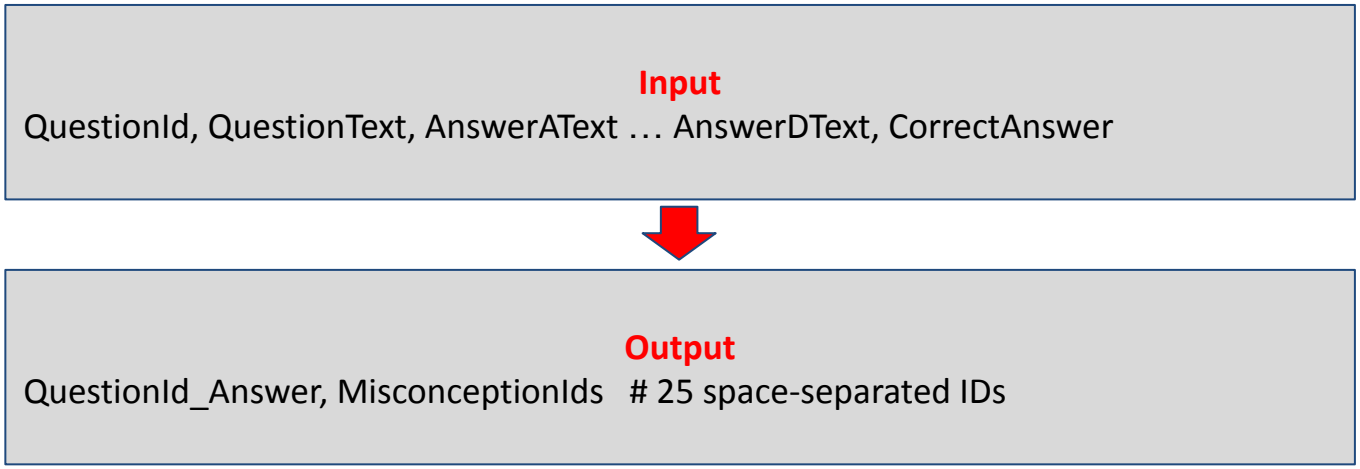
Example of a Diagnostic Question

- Diagnostic question: multiple choice question with one correct answer and three distractors (incorrect answers)
- each distractor captures a specific misconception
- distractor “13” -> "Carries out operations from left to right regardless of priority order." misconception



The competition

Task: For every (question, wrong answer) pair, suggest 25 plausible misconceptions that could have caused the mistake.

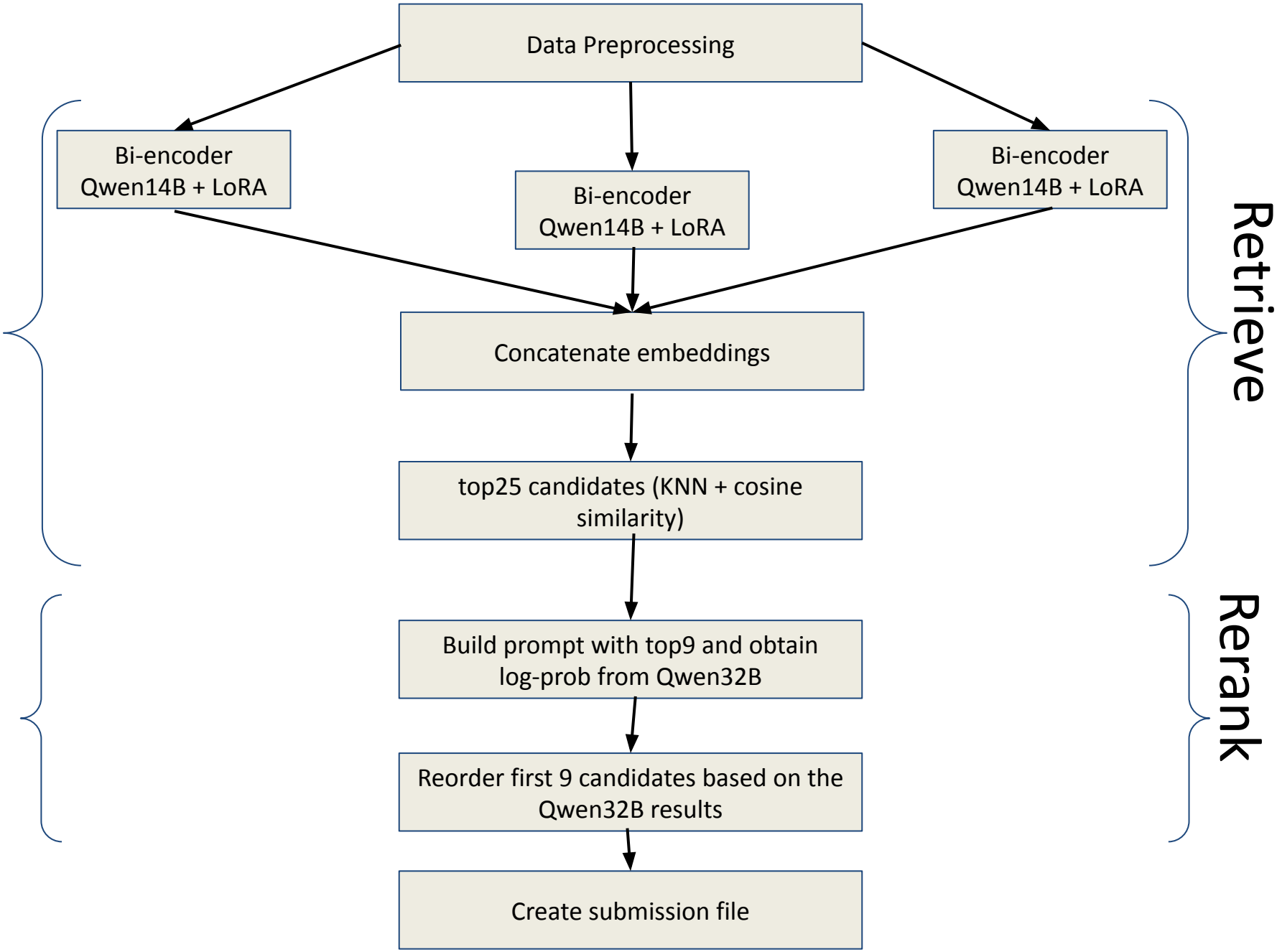


Why is it difficult?

- over 2 500 misconception labels -> extreme class imbalance
- Each question is unique, only text and domain knowledge
- Evaluation Metric - MAP@25, single correct label per distractor, the rank of the correct misconception matters a lot

Big picture

- 2 Stage solution: Retrieve & Rerank
- wide recall first -> precise rerank later


Data Preprocessing
Bi-encoder Qwen14B + LoRA
Bi-encoder Qwen14B + LoRA
Bi-encoder Qwen14B + LoRA
Concatenate embeddings
top25 candidates (KNN + cosine similarity)
Build prompt with top9 and obtain log-prob from Qwen32B
Reorder first 9 candidates based on the Qwen32B results
Create submission file
Retrieve
Rerank

Data Reshaping

- Copy the correct answer text into CorrectText to be used later in prompts.
- Melt the 4 answer columns (AnswerAText,...)
- Drop the rows with the wrong choice == correct choice
- The test rows will have question + wrong answer

1 Retrieval stage (bi-encoder)

1.1 Why use big LLMs as encoders for embeddings?

We can use a decoder-style LLM like Qwen as a sentence embedder.

- Feed the prompt as usual
- Take the hidden state of the last non-padding token
- L2-normalise the vector
  - each vector has length 1, lives on the unit hypersphere
  - gives us a better comparison between vectors

1 Retrieval stage (bi-encoder)

- Cosine similarity then becomes a plain dot product.

  With this trick we can turn a language model into a “one-vector-per-text” engine (like SBERT but with a much larger internal representation)

1.2 The three “flows”

Flow ID	Base weights	Extra weights	Prompt template	Wording type
anhvth226	Qwen-2.5-14B-Instruct (4-bit NF4)	LoRA #1 (2211-lora)	“<instruct> Given a math multiple-choice problem with a student’s wrong answer…”	explicit about the task
mschoo	Qwen-2.5-14B-Instruct (4-bit NF4)	LoRA #2 (14b-cp750)	“Instruct: Given a math question with correct answer and a misconcepted incorrect answer…”	Slightly different phrasing to encourage a distinct embedding geometry
zuoyouzuo	Qwen-2.5-14B-Instruct (4-bit NF4)	LoRA #3 (qwen14b-it-lora)	“Instruct: Given a math question with correct answer and a misconcepted incorrect answer,...”	Adds still another view of similarity.

LoRA

- EAch LoRA checkpoint was fine-tuned by other Kaggle competitors on synthetic data to make the vectors cluster by misconception
- Low-Rank Adaptation freeze the big matrix, learn two smaller matrices A (4096×64), B (64×4096) → only ~1 M parameters.
- r = 64 << 4096
- During inference: $W' = W0 + \alpha \cdot 1/r \cdot A \cdot B$
- Cheaper to train and easier to load, can be merged into 4-bit weights

1.3 The template

```
def anhvth226_template(row: pd.Series) -> str:
    template = """<instruct>Given a math multiple-choice problem with a student's wrong answer, retrieve the math misconcept
<query>Question: {question}

SubjectName: {subject}
ConstructName: {construct}
Correct answer: {correct}
Student wrong answer: {wrong}
<response>"""
    return template.format(
        question=row["QuestionText"],
        subject=row["SubjectName"],
        construct=row["ConstructName"],
        correct=row["CorrectText"],
        wrong=row["WrongText"],
    )
```

Every bi-encoder embeds the template and the misconceptions separately making a vector space where related items are close

two texts -> get the vector representation for both -> cosine similarity tells us “how related” they are

1.4 Running the models

1. Quantisation - shrinks weights 8x while keeping fp16 compute. (lets us run 14 B on 16 GB & 32 B on two 16 GB GPUs)
2. LoRA attach - overlays the small delta matrices.
3. Forward pass - done under torch.autocast(“cuda”) -> fp16 on cpu
4. Pooling + normalise + .cpu -> small cpu tensor
5. Repeat for misconceptions

1.5 Ensemble by concatenation

Concatenate along the feature axis

$[q_1 \mid q_2 \mid q_3]$ shape = 4096 + 4096 + 4096 = 12288 dims

$[m_1 \mid m_2 \mid m_3]$ shape = 12288 dims

Why not average?

- each LoRA sees “semantic distance” differently
- putting coordinates side by side preserves signals and we can still use cosine similarity
- we concatenate different “views” to boost recall without voting

1.6 K-nearest-neighbours

```
# compute KNN
nn = NearestNeighbors(n_neighbors=25, algorithm="brute", metric="cosine")
nn.fit(all_m_embeds)
dist, topk_mis = nn.kneighbors(all_q_embeds)
```

picks the 25 closest misconception vectors to each query vector.

exact search returns the 25 nearest misconception IDs

topk_mis is our current answer set - high recall but noisy order

2 Reranking stage (cross-encoder)

2.1 Model choice

- Reranking a small candidate set with a heavier model gives most of the precision for a fraction of compute.
- Qwen-2.5-32B-Instruct-AWQ,quantised with Activation-aware Weight Quantisation (2–4 bits)
- loaded with vLLM - to shard the model across 2 GPUs and streams tokens very fast
- good enough for a small candidate set (25)

2.2 Prompt construction

<instruction> You are an elite mathematics teacher
Subject, Question, Correct Answer, Wrong Answer
Possible misconceptions:
1 first candidate retrieved
2 second candidate retrieved
…
9 ninth candidate retrieved
Select one misconception, output a single number
Answer:

2.2 Prompt construction

Why only 9 candidates?

We want single token labels(“1”,...,“9”). Qwen would split “10” into two tokens “1” and “0” which breaks the next trick

We could use all 25 candidates if instead of digits we would use the A-Z letters as labels

2.3 MultipleChoiceLogitsProcessor

https://github.com/NVIDIA/logits-processor-zoo/blob/main/README.md

Smart logits masking can turn a generative model into a single-step multiple-choice classifier.

- at the first step vLLM gives us a logits vector (of dimension vecab_size)
- The processor sets logit = -inf for every token except IDs of “1”…”9”
- Softmax is applied to obtain a distribution of probabilities
- Temperature = 0 -> argmax (choose highest probability)
- We also request logprobs=9 so we get the full probability table in a single forward pass
- We obtain a score for each of the 9 candidates.

2.4 Reorder

indices = argsort(logprobs, desc=True) # 0-based 0-8
reranked = top25[indices] + top25[9:]

The cross-encoder’s preferred misconception becomes position 0, the worst of the nine becomes position 8, and positions 9-24 keep their bi-encoder order.

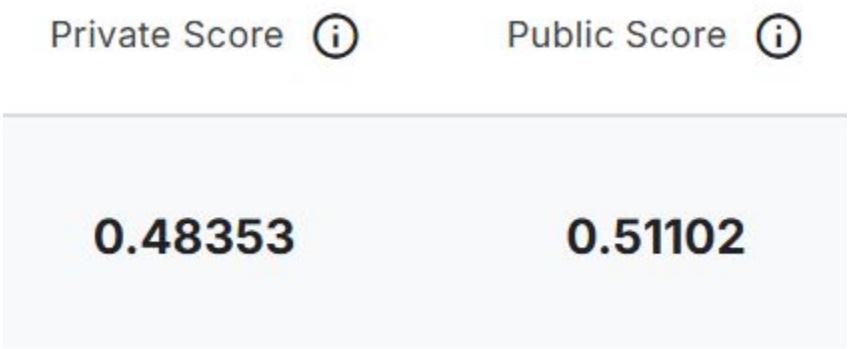
2.5 Map to misclD

Map the row indices to the MisconceptionIds and write the submission.csv file

Results

Aprox top 90 of solutions

1 place solution: 0.63881



Possible improvements

- Rerank all 25 with single-token labels (letters A-Y instead of digits)
- Fine-tune a 14 B cross-encoder
- Synthetic MCQs + pseudo-labeling
- Multi-stage rerank (14 B $\longrightarrow$ 32 B $\longrightarrow$ 72 B)
- Chain-of-Thought distillation into prompts for finetuning the models

Questions & Discussion