CSE 256 Final Project - Eedi - Mining Misconceptions in Mathematics

Ming-Kai Liu
mil151@ucsd.edu

1 Introduction

Eedi - Mining Misconceptions in Mathemtaics is a Kaggle competition focusing on NLP domain. The goal of this competition is to predict the affinity between misconceptions and incorrect answers (distractors) in multiple-choice questions. For example, given a multiple-choice question with four options:

$$5 \times 4 + 6 \div 2 =$$
$$A.23$$
$$B.13$$
$$C.25$$
$$D.35$$

The correct answer is "23". If a student selects the distractor "13", they may have the misconception "Carries out operations from left to right regardless of priority order".

There are two main challenges. First, the complexity of the mathematical content in the questions hinders the pre-trained language models to identify underlying misconceptions. The models need to understand the question thoroughly to determine what misconceptions might lead to incorrect answers. Second, the scarcity of the dataset makes it hard to fine-tune models. 2500 types of misconceptions are given, and 700 of them does not appear in the training dataset. Moreover, the data distribution is extremely unbalanced - one third of the misconceptions appear only one time in the training data.

Luckily, as Large Language Models (LLMs) revolutionized the field of natural language processing, many efforts had been made to improve LLMs' reasoning capabilities, with a particular emphasis on their ability to solve arithmetic and mathematical problems[1, 2, 3]. These math-specific LLMs are familiar with the mathematical expressions written in latex format and learned complex mathematical concepts from their vast training data, thus serve as a great solution for the first challenge.

For the second challenge, building a classifier that produces probability for each class(i.e. misconceptions)would be hard due to data scarcity and large number of classes. We need a way to leverage math-specific LLMs' mathematical reasoning ability and retrieve corresponding misconception for each incorrect answer, where text embedding model comes in play. [4] used LLM's last hidden state as the text embedding and fine-tuned[3], achieving state-of-the-art result on text embedding task.

In this work, I leveraged state-of-the-art mathematical LLM's reasoning ability in a zero-shot manner and designed a pipeline that retrieve misconception for each incorrect answer. I also cooperated with other classmates Yen-Ting Lee and Eddy Chu to fine-tune mathematical LLM as a text embedding model.

- Design the pipeline to solve the task and examine its performance: Done
- Fine-tune LLMs as a text embedding model: Going on

2 Related work

Existing methods are mostly related to following topics:

Large Language Models LLMs have been proven to be capable to solve complex mathematical problems, and mathematical reasoning ability has became one of the benchmark when comparing LLMs[5, 6, 7]. Math-specific LLMs are LLMs that are specifically tailored for mathematics[1, 2, 3]. While math-specific LLMs possesses rich mathematical knowledge, their objective are generating the correct answer, instead of identifying

the misconceptions. Therefore, naively inferring math-specific LLMs leads to a unsatisfactory result.

Text Embedding Text embeddings are vector representations of natural language that encode its semantic information. They are widely used in various natural language processing tasks, such as information retrieval, question answering, semantic textual similarity, etc. Other participants either utilized text embedding model to directly retrieve misconceptions or to retrieve most likely misconceptions and then rerank these misconceptions.

Contrastive Learning Contrastive Learning aims to minimize the distance between positive pair and maximize the distance between negative pair. Several contrastive loss function have been proposed, such as [8, 9]. It is useful when trying to learn effective representation when labeled data are scarce. In this case, it serve a good method to maximize the data utility.

3 Dataset

3.1 Statistics

The training dataset consists of 1869 multiple-choice question. Each multiple-choice question comes with 1 correct option and 3 incorrect options. Each question also comes with a *ConstructName* and *SubjectName*. *ConstructName* is the most granular level of knowledge related to question. *SubjectName* is the more general context than the *ConstructName*. Three incorrect options are labeled with the id of misconceptions. There are 2587 different types of misconceptions. For example:

Simplify the following, if possible: $(\frac{m^2+2m-3}{m-3})$

$A.(m+1)$

$B.(m+2)$

$C.(m-1)$

$D.$Does not simplify

ConstructName: Simplify an algebraic fraction by factorizing the numerator

SubjectName: Simplifying Algebraic Fractions

CorrectAnswer: D

MisconceptionAId: 2142 (Does not know that to factorise a quadratic expression, to find two numbers that add to give the coefficient of the x term, and multiply to give the non variable term)

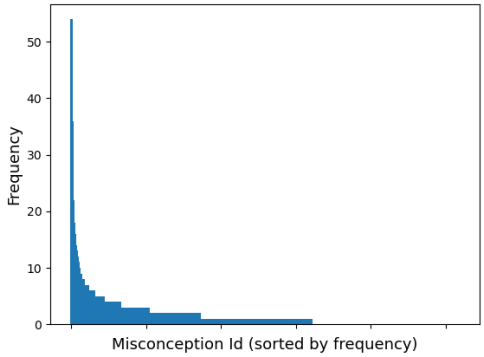
MisconceptionBId: 143 (Thinks that when you cancel identical terms from the numerator and denominator, they just disappear)

MisconceptionCId: 2142 (Does not know that to factorise a quadratic expression, to find two numbers that add to give the coefficient of the x term, and multiply to give the non variable term)

MisconceptionDId: None

During testing, the question, incorrect option, *ConstructName*, *SubjectName* and *CorrectAnswer* are given. The goal is to predict the corresponding misconception for each incorrect option. (The testing dataset is all private on Kaggle server)

Figure 1: Distribution of misconceptions



From Fig.1, one third of the misconceptions doesn't appear in training dataset. Moreover, most of the misconceptions only appeared 1-10 times. The scarcity of the training dataset makes model hard to learn its distribution.

3.2 Evaluation

During the private testing, the model will generate top 25 misconceptions for each incorrect answer. The evaluation is based on Mean Average Precision @ 25 (MAP@25):

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

where U is the number of incorrect options (i.e. testing data), $P(k)$ is the precision at cutoff k, n is the number of predictions model generate, and $rel(k)$ is an indicator function equaling 1 if the item at rank k is a correct label, zero otherwise.

4 Approach

4.1 Pipeline

The whole pipeline could be divided into three part: 1) LLM reasoning, 2) Selecting top candi-

date using embedding model, 3) LLM reranking.

LLM reasoning I believe retrieving corresponding misconception from only question, incorrect option, *ConstructName*, *SubjectName* and *CorrectAnswer* overlooks important intermediate reasoning steps which was shown vital in arithmetic task[10]. Therefore, the first step is to prompt Qwen2.5-32B-Instruct to reason the question and provide possible misconception (not necessarily related to the misconceptions in the training dataset). Qwen2.5-32B-Instruct was instruction-tuned to solve mathematical problems in a chain-of-thought manner. It should produce step-by-step analysis on misconception that lead student to choose the wrong answer. Prompt is formatted as follow:

"Here is a question about {*ConstructName*} ({*SubjectName*}):
- Question: {*question*}
- Correct Answer: {*CorrectOption*}
- Wrong Answer: {*IncorrectOption*}

Please provide a detailed analysis on what misconception or reasoning error that cause the student to derive the wrong answer. Focus only on explaining the misconception."

Selecting Top Candidates The text embedding of Qwen2.5-32B-Instruct's response is extracted using Salesforce Embedding Model[11] and compared with the text embedding of all 2587 misconceptions. Then, top 100 similar misconceptions are selected as candidates.

I compared the performance of Lora fine-tuned Salesforce Embedding Model (trained by other participant) with the original one. The Lora fine-tuned one was trained to maximize the similarity between (question, incorrect option, *ConstructName*, *SubjectName* and *CorrectAnswer*) and (correct misconception). Interesting, using Lora fine-tuned one drastically improve the overall performance, though the way it is applied is different from the original training objective.

LLM Reranking After selecting top 100 candidates, they are combined with (question, incorrect option, *ConstructName*, *SubjectName* and *CorrectAnswer*) and fed into Qwen2.5-32B-Instruct again. This time, the LLM is asked to choose the top 1 misconception. Finally, the top 1 misconception is combined with other top 24 misconceptions as my final answer.

4.2 Computing Resource and Runtime

This competition requires notebooks to be run on Kaggle server which has max 32GB vram (two T4 GPU). The total inference time takes about 4 hours (1000 private questions).

4.3 Other

I also cooperated with other two classmates Yen-Ting Lee and Eddy Chu to fine-tune Qwen2.5-Math-7B following [4]. We use the last hidden state of the LLM as text embedding. The training objective is to minimize the distance between positive pair and maximize the distance between hard negative pair.
Positive pair: {(question, incorrect option, *ConstructName*, *SubjectName* and *CorrectAnswer*), (correct misconceptions)}.
Negative pair: {(question, incorrect option, *ConstructName*, *SubjectName* and *CorrectAnswer*), (incorrect misconceptions)}.
The fine-tuned Qwen2.5-Math-7B is then used in the last stage in my pipeline to rerank the top 100 candidates.

I was mainly responsible for **designing the pipeline** due to my limited computing resource.

5 Baselines

The fine-tuned model here are all other participants' work and their score are test on private dataset.

Fine-tuned bge (bge*) Fine-tune bge[12] using Multiple Negative Ranking Loss. When testing, directly retrieve misconceptions that has top 25 simialr embedding as the (question, incorrect option, *ConstructName*, *SubjectName* and *CorrectAnswer*)

Fine-Tuned Salesforce Embedding Model (sfr*) Fine-tune Salesforce Embedding Model using Multiple Negative Ranking Loss. Same usage as the fine-tuned bge.

Fine-Tuned Qwen2.5-14B (Q-14B*) Use last hidden state as text embedding and fine-tune Qwen2.5 using contrastive loss. Same usage as the embedding models above.

Qwen2.5-32B-Instruct + Fine-Tuned bge (Q-32B+bge*) Prompt Qwen2.5-32B-Instruct to summarize core mathematical concept in the question and use the text embedding of the LLM's output to retrieve top 25 misconceptions.

Fine-Tuned Salesforce Embedding Model + Qwen2.5-32B-Instruct (sfr*+Q-32B) Retrieve top 25 misconceptions using the text embedding of (question, incorrect option, *ConstructName*, *SubjectName* and *CorrectAnswer*) and then use Qwen2.5-32B-Instruct to rerank the misconceptions.

Fine-Tuned Qwen2.5-14B + Qwen2.5-32B-Instruct (Q-14B*+Q-32B) Use Qwen2.5 as embedding model and retrieve top 25 misconceptions using the text embedding of (question, incorrect option, *ConstructName*, *SubjectName* and *CorrectAnswer*) and then use Qwen2.5-32B-Instruct to rerank the misconceptions.

Method	MAP@25
bge*	0.297
sfr*	0.353
Q-14B*	**0.482**
Q-32B+bge*	0.373
sfr*+Q-32B	0.451
Q-14B*+Q-32B	0.468
Q-32B+sfr*+Q-32B (Mine)	**0.476**

Table 1: Comparison of Map@25 Score

My method sucessfully increase the overall performance by 0.025 compared to sfr*+Q-32B. This score haven't incorporated our fine-tuned Qwen2.5-Math-7B model. We expect the score will be higher after replacing Qwen-32B-Instruct with our fine-tuned Qwen2.5-Math-7B.

6 Error analysis

Baselines mostly fail on misconceptions that didn't occur in the training dataset or those that only appear for few time in the training dataset (Fig. 1). My approach is slighly more robust on unseen or less seen misconceptions since in the first stage the Qwen2.5-32B-Instruct first provide possible misconception.

However, my approach is prone to choose more general misconceptions over specific misconceptions. For example, for the following example:

Where do the brackets need to go to make the answer equal 13 ?

$3 \times 2 + 4 - 5$

CorrectOption: $3 \times (2 + 4) - 5$

IncorrectOption: Does not need brackets

ConstructName: Use the order of operations to carry out calculations involving powers

SubjectName: BIDMAS

my method would choose *"Misunderstands order of operations in algebraic expressions"*, whereas *"Confuses the order of operations, believes addition comes before multiplication when the latter one is the correct misconception"* is the correct misconception.

I believe this is because the though Qwen2.5-32B-Instruct is excelled at generating right answers, it was not trained to identify error in the solving process. Therefore, the likely misconceptions it choose are often more general.

7 Conclusion

LLMs possess powerful zero-shot performance, especially when combing different LLMs to tackle task. Fine-tuning them will definitely increase performance by a large margin. However, fine-tuning LLMs requires tremendous computing resource... My teammate and I struggled to fit even the smallest 7B model in the Qwen2.5 series into our 24GB GPU. What's worse is that students normally do not have the access to the large VRAM gpu such as A100. I am more certain about the importance of compressing models in the future and would like to start to research on it!

8 Acknowledgements

I didn't use any generative tools to modify my report.

References

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

[2] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024.

[3] Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b, 2023.

[4] Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. Improving text embeddings with large language models, 2024.

[5] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[6] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[7] Gemini Team, Petko Georgiev, Ving Ian Lei, Ryan Burnell, Libin Bai, Anmol Gulati, Garrett Tanzer, Damien Vincent, Zhufeng Pan, Shibo Wang, et al. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*, 2024.

[8] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, page 815–823. IEEE, June 2015.

[9] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding, 2019.

[10] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2023.

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

[12] Shitao Xiao, Zheng Liu, Peitian Zhang, and Niklas Muennighoff. C-pack: Packaged resources to advance general chinese embedding, 2023.