MAP Charting Student Math Misunderstandings - 6th Place Solution Details

Manan Jhaveri | mananjhaveri.jam@gmail.com
Data Scientist @ Optum, Mumbai, India

Background

- Competition details
    - Competition Name: MAP - Charting Student Math Misunderstandings
    - Team Name: Manan Jhaveri
        - Solo participation
    - Private Leaderboard Score: 0.94828
    - Private Leaderboard Place: 6th
- I am a Data Scientist at Optum (United Health Group) from Mumbai, India.
- I have done my undergrad (B.Tech) in Data Science from NMIMS Mukesh Patel School of Technology Management and Engineering in Mumbai.
- My work is primarily around NLP, fine-tuning language models and working with a ton of text data on a massive scale. I also teach a subject on NLP at my alma mater and have followed NLP competitions for a while now on Kaggle. All of these experiences and my interest in the field of NLP and LLMs helped me perform well in this competition.
- I entered this competition as the problem statement looked familiar to the work I do, the dataset wasn't too massive to require incessant amounts of compute resources and I had been out of touch with Kaggle competitions for a while and was yearning to return to them.
- I spent about 15-20 hours per week on average in this competition. Sometimes it was more when I had to debug my code or generate synthetic data. Sometimes it was lesser if I was travelling or swamped with work.

Summary

- I used a mix of Qwen3-Embedding-8B, Qwen3 14B and Qwen2.5 14B.
- I finetuned these models as classification models using the transformers library.
- I used Q-LoRA for finetuning on single A5000 / A100 GPUs.
- Using Synthetic data for rare labels, Augmented text and Pseudo-labelling duplicates helped to improve the performance of the models.
- Training time:
  - For the 8B model, ~7 hours on A5000 and 3.5 hours on A100.
  - For the 14B model, ~10 hours on A5000 and 5-6 hours on A100.
  - All models were trained for 3 epochs.

Data and pre-processing

- From the start, I saw this problem as just a misconception classification problem. So I removed the True_ and False_ prefixes from the categories and used the resulting 37 labels during fine-tuning.
- Used the following input format:

```python
correctness = "Yes" if row["is_correct"] else "No"

input_text = (
    f"Question: {row['QuestionText']}\n"
    f"Answer: {row['MC_Answer']}\n"
    f"Correct: {correctness}\n"
    f"Explanation: {row['StudentExplanation']}\n"
    f"Task: Classify the misconception in the explanation."
)

"""
Example
-------
Question: What fraction of the shape is not shaded? Give your answer in its simplest form. [Image: A
triangle split into 9 equal smaller triangles. 6 of them are shaded.]
Answer: \( \frac{1}{3} \)
Correct: Yes
Explanation: One third is equal to tree nineth
Task: Classify the misconception in the explanation.
"""
```

- In the initial experiments, I noted that adding the last line of the prompt was improving CV and LB scores for the LLMs. Hence, I decided to keep it for the rest of the experiments.

- I dropped the duplicate samples altogether. Duplicates = same answer and explanation but different misconception as label.
- In the last few experiments, I pseudo-labelled these duplicates and added them back to the training data.
- In some experiments, I added ~50 augmented samples per category. Augmentation involved basic replacement of punctuations, conversion of words to number or vice versa, replacing "plus" to "+", replacing "equals to" to "equals" or "=", adding small typos, etc.
    - I had analyzed near duplicates found through fuzzy search in the training data and accordingly tried to create augmented samples.
- In the final few experiments, I added synthetic data for 10/37 categories with the lowest support. I used ChatGPT 4o to generate 100-120 samples per category.
- I used a max sequence length of 160 with dynamic padding. This made the training and inference faster for me.
- For local testing, I have used 20% test split and used full data for re-training the model for the submissions.

Training Method

Models

- I started my experiments with a variety of models like GTE, Ettin and later moved to LLMs of up to 8B parameters. I observed that the **sequence classification approach** with Qwen3-Embedding-8B performed the best for me.
- After freezing my prompt template and pre-processing, I moved to 14B param models too.
- I trained these LLMs with 4-bit Q-LoRA. Rank = 128, alpha = 32-128 (as per different experiments).
- I used A5000/A100 on Jarvis Labs for compute.
- I kept the batch size 12 for both 8B and 14B models. A learning rate of 1.5e-4 worked best for 8B models and 6e-5 worked best for 14B models. All finetuning experiments were done for 3 epochs.

Here's the summary of the models that I finally used in my ensemble:

Model	Data used	Public LB
Qwen3 Embedding 8B	Deduplicated data	0.946
Qwen3 14B	Deduplicated data	0.947
Qwen2.5 14B IT	Deduplicated data	0.947
Qwen3 Embedding 8B	Deduplicated data + Augmented data	0.945
Qwen3 Embedding 8B	Deduplicated data + Pseudo-labelled duplicates + Synthetic data	0.949 (best single model)
Qwen3 14B	Deduplicated data + Pseudo-labelled duplicates + Synthetic data	0.948
Qwen3 14B	Deduplicated data + Pseudo-labelled duplicates + Synthetic data (+ different seed than above)	0.943

Ensembling

- I used the ensembling method from this notebook by @kishanvavdara : https://www.kaggle.com/code/kishanvavdara/ensemble-gemma-qwen-deepseek
- The ensemble of the first 4 models from the above table helped me crack 0.95 on the public LB.
- I used this ensemble to pseudo-label the duplicates.
- The model with augmented data has a poor individual score compared to the rest but it contributed significantly to the ensemble. I tried removing it multiple times as I got better models but the score always dipped.
- With the ensemble of the first 6 models, I got 0.951 on the public LB (and 0.947 private LB).
- In the last 2 days, I was out of ideas and I didn't want to make small tweaks to model weightage in the ensemble method as it felt unreliable. So I decided to fine-tune Qwen3 14B with the same config as the one I used earlier in the 0.948 model, but with a different seed and random state. Its individual score was quite poor but ensembling it with the other 6 models gave me my best public as well as private LB score.
- I gave equal weightage to each model as playing with weightages wasn't improve my public LB score.

Interesting findings

1. I had my first breakthrough when I stumbled upon the Embedding models of Qwen3. Using the 8B variant beat all other models that I tried from this model size category. Other models were giving a score of 0.94-0.942 on the public LB and Qwen3-Embedding-8B gave a score of 0.944. I think it is closer to a traditional encoder model than a decoder model and that's why it performs better on discriminatory tasks like multi-class classification.
2. Dropping the duplicates and updating the prompt.
    a. This simple thing helped me jump from 0.944 to 0.946 on the public LB.
    b. This is interesting since the duplicates with conflicting labels were only ~60.
    c. Adding the line "Task: Classify the misconception in the explanation." to input template seemed a bit unintuitive and redundant as it is going to be common for all samples. But perhaps it was helping LLMs to get more context about the task.
3. Text augmentation
    a. From this public notebook - https://www.kaggle.com/code/kyoumonemui/i-love-the-answer-quick-eda - I understood that there are many samples which are small variations of each other.
    b. Building on top of this idea, I add 50 simple augmented samples per category to the training data. This didn't improve the score on the LB but helped the ensemble.
4. Synthetic samples
    a. It was obvious that the categories with less supported would performing worse than others.
    b. I used chatgpt to generate synthetic data for 10 of these rare classes.
    c. I used simple prompts on the UI itself. It is a bit messy and suboptimal but didn't have much time to invest in setting up the API and crafting the perfect prompts.
    d. This is all I did to generate these synthetic samples - https://chatgpt.com/share/68fdfc99-9a30-8004-9f83-667ca4bd102a

Simple Features and Methods

- Using the new small encoder based models like ModernBERT Base, GTE Base and Ettin-encoder-150M yield a fairly decent score, around ~0.927-0.932.
- These are smaller models but pre-trained on massive datasets and support a variety of model optimizations like flash attention, unpadding.
- On public LB, my best single model was 0.949. And in the experiments, I got a score of 0.927 using GTE Base (trained only on 80% data). That's just a loss of ~2.5% in performance but almost a 10x speedup in inference and much more in training.
    - Note - the best model uses a preprocessed dataset with synthetic data for rare labels. The GTE base model just uses the training data as is.

- So adding synthetic data and dropping duplicates should improve the GTE's performance further.

Model Execution Time

- Training time
  - Device config - 1 x 24GB A5000
  - 8B model - ~6.5 hours
  - 14B model - ~9.5-10 hours
  - Simplified model (~100M params) - ~5-7 mins
- Inference time
  - Device config - 1 x 15GB T4 on Kaggle
  - 8B model - ~1 hours
  - 14B model - ~1.5 hours
  - Simplified model (~100M params) - 1-1.5 mins

Were some labels more difficult to learn for the model than others?

I had done a high-level error analysis of my models midway through the competition. I had found 2 types of errors broadly:

- The model often confused the "Neither:NA" and "Correct:NA" categories.
  - After looking at the errored cases and some samples from the training set, I understood that it was mostly because of the inconsistency in the annotation.
  - I planned to audit such cases and update the labels but found a discussion post which said that cleaning the dataset didn't help to improve the score on the LB.
  - Later, Chris Deotte also posted that cleaning the dataset won't be a good idea since the hidden test sets will also have the same type of labelling inconsistencies (ref).
  - This is also the reason why the best LB scores stayed around ~0.95.
  - Hence - I didn't spend time on fixing this type of error.
- The other issue was that under-represented labels were not getting predicted.
  - Labels with the least support, namely - Incorrect_equivalent_fraction_addition, Wrong_Operation, Certainty, Inverse_operation, Ignores_zeroes, Base_rate, Longer_is_bigger, Interior, Definition, Shorter_is_bigger, FlipChange - were not got predicted by the model as the training data had very few samples (<100 for all) for these categories.
  - This was resolved to a decent extent by adding augmented data, synthetic data and ensembling of multiple models.

    - I feel getting a larger, cleaner dataset with more unique samples (and not just augmented ones) should help to yield great results from even a small encoder model suggested above.

Failed experiments

- Multiple pre-trained models.
- Several prompt templates.
- Adding too many augmented samples. I had shortlisted multiple patterns from my analysis and created ~10k augmented samples. But the public LB score remained ~0.945 and these models didn't add much to the ensemble.
- Different modelling approaches:
    - Using all original 65 labels.
    - I tried a 2-stage approach where the first stage model just predicts whether a student's explanation is correct, has a misconception or neither of the two. And the second stage would predict the misconception type.
    - Text Generation approach where I provided simplified descriptions of each category (that a question can possibly have) in the prompt and asked the model to predict which one best describes the reasoning behind the student's explanation.
    - Textual Entailment approach where I tried to predict whether a given sample entails any of the simplified descriptions of the misconception categories.
    - MoE-style NN for the classification head instead of LoRA adapters.
- Different loss functions like focal-loss
- Most of the above experiments failed mainly because I couldn’t commit enough time to configure them properly.