**Natural Language Processing**

**Term Project**

**Eedi – Mining Misconceptions in Mathematics**

Group 4

National Cheng Kung University N96134158 蕭力文

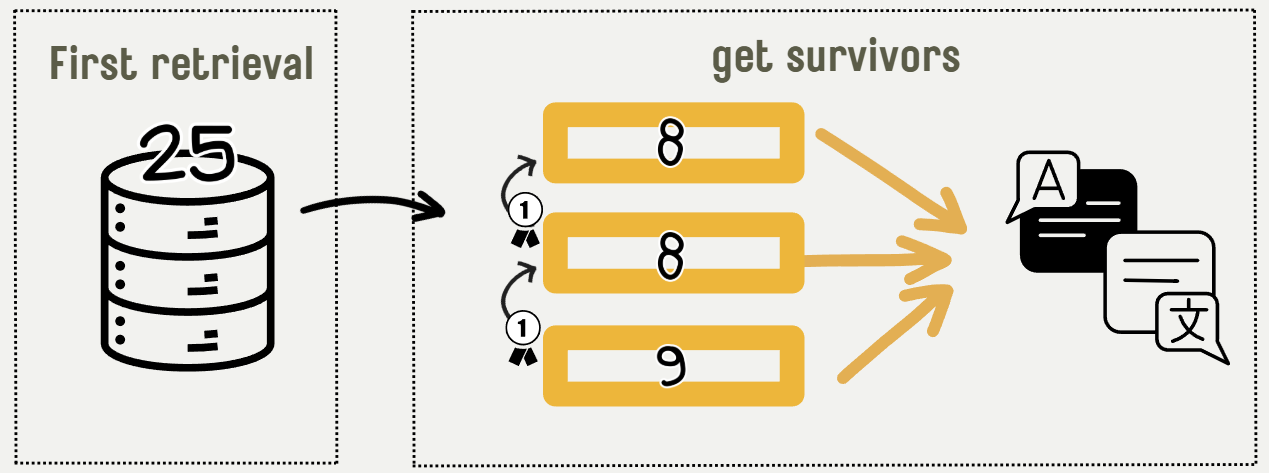
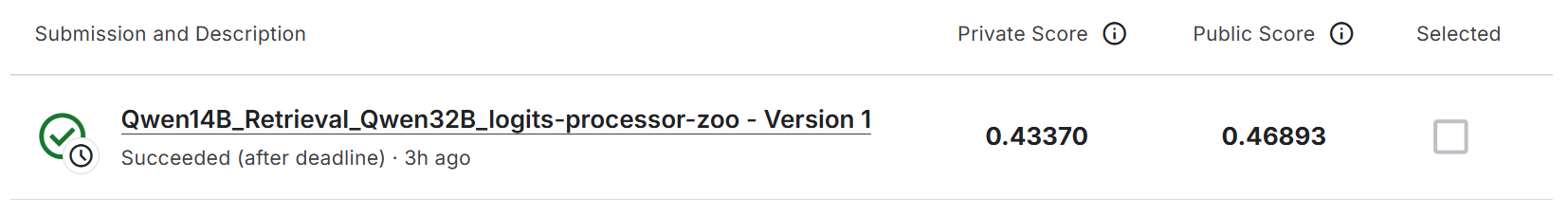
National Cheng Kung University N96134645 許宸華

**Competition Overview**

**Notebooks Review**

1. **Qwen14B\_Retrieval\_Qwen32B\_logits-processor-zoo**
2. First Retrieval:

embedding model:  
Qwen-14b( /kaggle/input/qwen2.5-14/pytorch/default/1)  
  
首先將整理好的 data 根據每一行的資料，生成一個 query\_test，該文本包含了 QuestionText、correct\_answer、incorrect\_answer 、 ConstructName 、SubjectName。這些資料會被存儲在一個新的 dataframe 中。  
  
使用對於數學問題特別擅長的模型”千問(Qwen)”來對於 query\_test 和 misconceptions 進行 embedding，再使用 n\_neighbors 的方法將 embedding 完的 misconceptions 包成 25 個一組的 indices，作為第一次的檢索。

1. LoRA finetune:  
     
   使用 LoRA 技術對模型進行微調，這樣可以減少計算所消耗的記憶體，在 kaggle 比賽上也特別適用。在 kaggle discussion 的討論區有發現大家都使用 r=64，lora\_alpha=128 的參數進行微調，我們也參照使用。  
     
   無論是 First Retrieval 使用的 Qwen-14b 或 Multiple Choice 使用的 Qwen-32b 都會經過 lora\_path 進行微調。
2. Multiple Choice:  
     
   LLM:  
   Qwen32b(/kaggle/input/qwen2.5/transformers/32b-instruct-awq/1)  
     
   First Retrieval 完的 indices 首先會將倒數最後一名作為最初的 survivors 並開始第一個輪次，每次會通過我們設計的 prompt 來進行 Multiple Choice ，詢問 Qwen32b 的語言模型來決定選項中的 misconceptions 哪一個會最適配 query\_test，然後將他存進 survivors 中，加入下一個輪次。總共會經過 3 個輪次的排序，最終會將 indices 中的misconceptions 排序完畢後存進 results，成為最終解答。  
   每輪會有 8+1 個 misconceptions，動態的更新 survivors 加入每一輪，經過3輪後會遍歷每個選項 (8\*3+1 = 25)  
   
3. 結果:  
     
   public score:0.46893; private score: 0.43370  
   ****
4. 討論:  
     
   我們認為這個程式太依賴 First Retrieval 的判斷，在 First Retrieval 會直接從 2000 多個 misconceptions 選取到剩下 25 個，在這過程中我們認為正確的答案很高機率會被丟棄。而且 First Retrieval 是透過 embedding 的語意分群來找到 25 個 misconceptions，我們也覺得光靠語意來判斷可能不夠精準。於是在 experiment 中會嘗試將 n\_neighbors 的數量增多來降低正確答案被丟棄的機率。  
     
   在 Multiple Choice 時會每輪去選擇最好的 misconceptions，希望將正確答案越往上推越好。但如果在 LLM 選錯答案時，後面的輪次可以說是無效輪次，所以增加每輪 survivors 的數量可能會有奇效；也可以去增加輪次，同時讓每輪次的 misconceptions 數變少，來增加挑選正確的概率。
5. **Eedi Qwen-2.5 32B AWQ two-time retrieval**
6. First Retrieval:

embedding model: SentenceTransformer(‘eedi-finetuned-bge-public/Eedi-finetuned-bge’) Embedding 模型採用對於此競賽finetune的SentenceTransformer。

首先對誤解對應的檔案進行資料預處理，將內容統一轉換為小寫英文、將不需要的雜訊例如網址等刪除，避免雜訊造成影響。

將'ConstructName'與'SubjectName’當作是輸入特徵，並以embedding model做embedding，接著再用semantic\_search取前100個比對分數最高的misconceptions作為第一次retrieval的結果。

1. LLM Reasoning:

LLM: Qwen2.5 32B instruct AWQ

大型語言模型選用Qwen2.5 32B instruct的量化模型，由於此競賽需要在Kaggle上的環境執行，因此採用量化模型作為模型的選擇。且為了提升效率與降低硬體部屬成本，採用vLLM框架來進行LLM的推理。

將第一次retrieval的Top-100 misconceptions與’ConstructName’, ‘SubjectName’, ‘Question’, ‘CorrectAnswer’, ‘IncorrectAnswer’以及一些提示詞作為大型語言模型的輸入prompt，相關參數設定: temperature=0; max\_token=512。

接著對於LLM的output做資料後處理，此output為LLM對於Top-100 misconceptions的結果推測。

1. Second Retrieval:

embedding model: SentenceTransformer(‘eedi-finetuned-bge-public/Eedi-finetuned-bge’) Embedding 模型採用對於此競賽finetune的SentenceTransformer。

先將第一次retrieval的資料作預處理，將內容統一轉換為小寫英文、將不需要的雜訊例如網址等刪除，避免雜訊造成影響。並將先前LLM的輸出結合第一次retrieval的內容作embedding。接著再用semantic search找到Top-25 misconceptions作為最終的答案。

1. 結果:

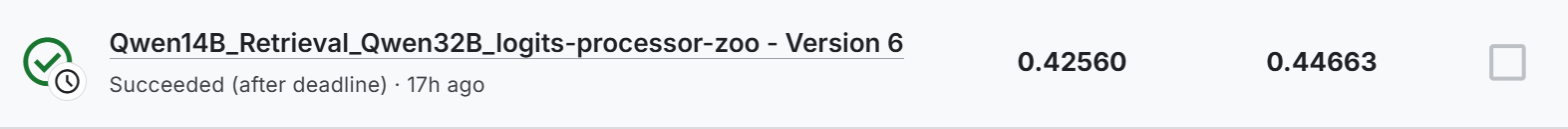
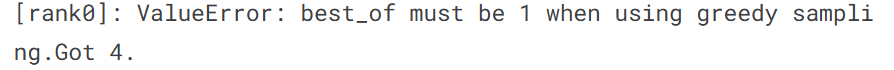
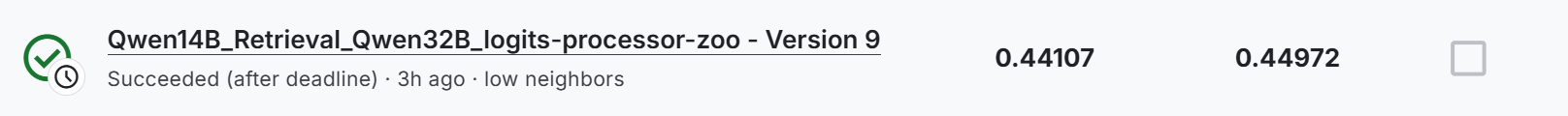
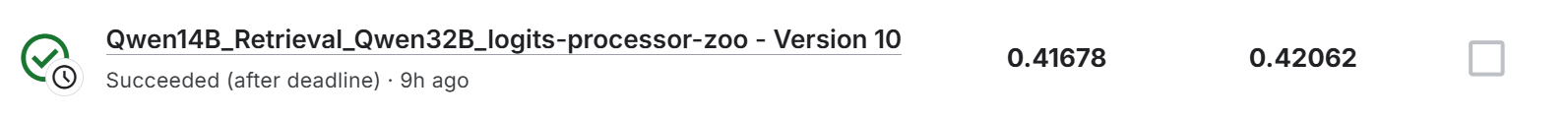
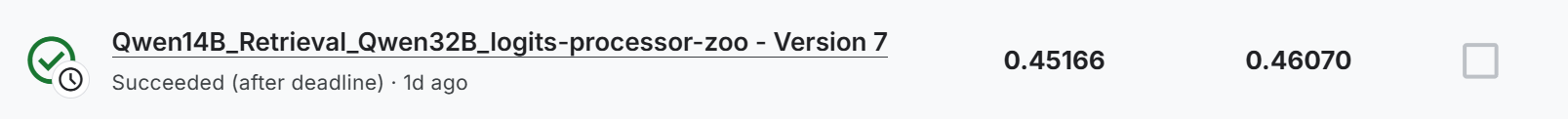
public score: 0.36097; private score: 0.30490

1. 討論:

在看了以上的Notebook後，我們覺得two-time retrieval是一個對於取top-n misconceptions是個好方法，使用兩次的retrieval且中間加入LLM的推測，使得最終在產生答案的時候可以有更多參照，使其得到不錯的效果。

問題是: 第一次retrieval時僅將'ConstructName'與'SubjectName’作為輸入特徵做embedding，我們認為這兩者對於一個問題、誤解來說並不一定會存在字面上相近的關係，非常有可能在剛開始取Top-100 misconceptions時就已經將正確答案給排除，因此我們可以從embedding的方式下手改善，此外，我們也在第六名的筆記中看到為避免大型語言模型對於取出100個誤解的排序會有依賴的情況，因此將取出之誤解打亂順序也是一個能夠提升效果的方式。

**Experiments**

1. **Enhance performance of multiple choice logits processor zoo.**
2. Increase the numbers of n\_neighbors  
     
   在本次實驗欲藉由增加 n\_neighbors 的數量來增加 First retrieval 取得正確答案的概率。  
     
   我們將 n\_neighbors 的數量增加至 40，Multiple Choice 輪次以及 survivors數量不變 (3 輪、1 個)，增加每輪的 misconceptions 至 13+1 以符合 indices 大小  
     
   然而無論是在 private score 還是 public score 都得到了比原本還要差的結果，這與我們預測的不一致。原先認為增加 n\_neighbors 的數量可以增加準確率，然而同時每輪的 misconceptions 的數量也增加是我們沒考慮到的，在面對選項變多的時候，LLM 的答對率也會相對地降低，我們推測這是為甚麼成效會變差，因此應該要在維持同樣數量的每輪的 misconceptions 情況下去增加 n\_neighbors 的數量或許才是合理的。
3. try to increase survivors in every rounds  
     
   我們試圖去增加每輪 survivors 的數量，最初設計每輪有 4 個survivors，一共 3 輪，每輪12+4 個 misconceptions，所以 n\_neighbors 的數量也一同增加至 40，來確保與 indices 數量一致。  
     
   然而經過多次調整嘗試後，程式仍然編譯不過。後來發現是在 vllm 調整時，雖然有正確改變 LLM 返回數量，但每輪返回的數量不一致導致indices 數量 error，由於對於 vllm 不夠了解，本次實驗沒有成功。  
   
4. Decrease the number of misconceptions in every rounds and increase the number of rounds  
     
   如果把每輪的 misconceptions 數量變少，是不是就不用增加 survivors 數量也可以達到類似的效果？  
   在同時我們也經由調整輪數來控制 n\_neighbors 的數量，來確定怎麼調整的效果比較好。  
     
   第一次設計每輪有 4+1 個 misconceptions，總共 12 輪，在 survivors 的數量保持為 1 的情況下 n\_neighbors 的數量為 49。  
   可以看到雖然在 public score 上有略遜於一開始的成績，但在 private score 上卻有顯著的提升，也證實了這個方法是有效的。  
     
   同時我們也設計了 n\_neighbors =13 的對照組，輪數降低至 3 輪，每輪的 misconceptions 數量不變為 4+1。  
   在 private score 和 public score 上都比 n\_neighbors 數多的還要差，也證實了增加 n\_neighbors 數對於增加準確度是有幫助的。然而同時在 private score 上還是比原先的成績要高，再次印證減少每輪的 misconceptions 數量是很有效的。
5. Use COT(Chain of Thought) to modify prompt  
     
   將原先的prompt改為COT的方式:  
   PROMPT = """Here is a question about {ConstructName}({SubjectName}).   
     
   Question: {Question}   
   Correct Answer: {CorrectAnswer}   
   Incorrect Answer: {IncorrectAnswer}   
     
   As a Mathematics teacher, let's analyze this step by step:   
   1. First, let's understand what the question is asking:   
    - What are the key mathematical concepts involved?   
    - What operation or method should be used?   
   2. Now, let's compare the correct and incorrect answers:   
    - What is different between these two answers?  
    - What steps might have led to the incorrect answer?   
   3. Based on this analysis:  
    - What fundamental misunderstanding could cause this error?  
    - How does this misunderstanding relate to {ConstructName}?  
    4. Looking at the given misconceptions:  
    {Retrival}   
   5. Final identification:  
    - Which misconception number best matches our analysis?  
    - Why is this the most appropriate choice?  
     
   Selected Misconception Number: [Your answer here]   
   Explanation: Briefly explain why this misconception led to the incorrect answer. """  
     
   除了要求 LLM 一步一步的思考外，還有比較正確答案跟錯誤答案的關係，最後也運用到 ConstructName 去比對做最後確認。  
     
   原先以為對於數學問題COT應該會有助於提升效能，沒想到結果竟然比較差。在survey其他notebook時，許多範本有明確的講到COT對於效能提升的幫助，所以這次結果比原先要差我認為是我對於COT的用法上理解不足，設計的不好，才導致結果變差。
6. 結果:  
   public score: 0.46070 ; private score: 0.45166  
   
7. 討論:  
     
   這次實驗中可以確定讓n\_neighbors增加以及讓每輪的misconceptions減少可以大大的增加表現，如果繼續嘗試說不定結果也會越來越好。而很可惜在更改prompt方面沒有達到更好的效果，這的部分我也還需要繼續專研。
8. **Combine two-time retrieval with multiple choice logits processor zoo.**

****

將two-time retrieval與multiple choice logits processor zoo結合，我們認為在multiple choice logits processor zoo中的重新排序能夠很有效地提升效果，因此想要結合兩者以達到更好的效果。

1. First Retrieval:

embedding model: SentenceTransformer(‘eedi-finetuned-bge-public/Eedi-finetuned-bge’) Embedding 模型採用對於此競賽finetune的SentenceTransformer。

首先對誤解對應的檔案進行資料預處理，將內容統一轉換為小寫英文、將不需要的雜訊例如網址等刪除，避免雜訊造成影響。

將'ConstructName'與'SubjectName’當作是輸入特徵，並以embedding model做embedding，接著再用semantic\_search取前100個比對分數最高的misconceptions作為第一次retrieval的結果。

1. LLM Reasoning:

LLM: Qwen2.5 32B instruct AWQ

大型語言模型選用Qwen2.5 32B instruct的量化模型，由於此競賽需要在Kaggle上的環境執行，因此採用量化模型作為模型的選擇。且為了提升效率與降低硬體部屬成本，採用vLLM框架來進行LLM的推理。

將第一次retrieval的Top-100 misconceptions與’ConstructName’, ‘SubjectName’, ‘Question’, ‘CorrectAnswer’, ‘IncorrectAnswer’以及一些提示詞作為大型語言模型的輸入prompt，相關參數設定: temperature=0; max\_token=512。

接著對於LLM的output做資料後處理，此output為LLM對於Top-100 misconceptions的結果推測。

1. Second Retrieval:

embedding model: SentenceTransformer(‘eedi-finetuned-bge-public/Eedi-finetuned-bge’) Embedding 模型採用對於此競賽finetune的SentenceTransformer。

先將第一次retrieval的資料作預處理，將內容統一轉換為小寫英文、將不需要的雜訊例如網址等刪除，避免雜訊造成影響。並將先前LLM的輸出結合第一次retrieval的內容作embedding。接著再用semantic search找到Top-25 misconceptions作為最終的答案。

1. Multiple Choice

LLM:  
Qwen32b(/kaggle/input/qwen2.5/transformers/32b-instruct-awq/1)  
  
First Retrieval 完的 indices 首先會將倒數最後一名作為最初的 survivors 並開始第一個輪次，每次會通過我們設計的 prompt 來進行 Multiple Choice ，詢問 Qwen32b 的語言模型來決定選項中的 misconceptions 哪一個會最適配 query\_test，然後將他存進 survivors 中，加入下一個輪次。總共會經過 3 個輪次的排序，最終會將 indices 中的misconceptions 排序完畢後存進 results，成為最終解答。

1. 結果:

public score: 0.36504 ; private score: 0.31072

1. 討論:

實驗結果表明，加入 Multiple Choice 之後，模型的 Public Score 從先前的方法提升至 0.36504，Private Score 提升至 0.31072。這證明了該方法在排序精度上的顯著提升。同時，分數的提升也間接驗證了我們的假設：在 Top-100 條目中引入語言模型的推理能力，以及在 Top-25 條目中進行多輪排序，能夠有效提升最終的預測準確率。

1. **Adjust the hyper-parameters of the LLM**
2. 我們使用的是實驗II的架構，並調整LLM的temperature至0.7，並將max\_tokens縮小至3000用以避免模型產生過長的答案。
3. 結果:

public score:0.37123 ; private score: 0.31072

1. 討論:

由於temperature=0時，模型總是會選擇機率最高的答案，生成結果偏向唯一正解，這會導致模型忽略其他潛在的可能性，而這又回到我們前面提到的，由於在embedding時的輸入特徵可能無法完全反應出本問題，所以若是每次都選擇機率最高的答案，我們就會忽略掉某些更「合適」的答案。另外，推測可能在資料集中的誤解類型間的界線，使得模型難以直接套用最高機率的選擇。因此將temperature提高至0.7有助於減少模型與資料的錯誤匹配帶來的性能下降。此外，將max\_token調製3000使得模型不會吐出超過3000個token的輸出，藉由控制輸出長度來使模型有更精確的回答。

**Other Approaches Introduction**

1. **15th Place Solution**
2. **Solution Overview and Features**

此解決方案採用兩階段的流程：

1. Retrieval Model: 用於選擇Top-25 misconceptions
2. Rerank Model: 用於根據相關性對Top-25 misconceptions進行排序

此解決方法的關鍵:

1. 重複ID處理: 對Top-25 misconceptions中重複ID的分數進行平均
2. 唯一ID: 保留其原始分數
3. 訓練數據結合: 實驗了各種訓練數據的組合以提高性能
4. **Performance Results**

Final Score:

public score: 0.617 ; private score: 0.553

Retrieval Model 1:

public score: 0.513 ; private score: 0.455 - Qwen2.5-32B

Retrieval Model 2:

public score: 0.515 ; private score: 0.450 - Qwen2.5-32B

Rerank Model 1:

public score: 0.610 ; private score: 0.538 - Qwen2.5-32B-Instruct-bnb-4bit

Rerank Model 2:

public score: 0.596 ; private score: 0.536 - Qwen2.5-32B-Instruct-bnb-4bit

1. **Synthetic Data Generation**

為了提升模型對誤解的理解，本解決方案的作者使用GPT-4o生成了合成數據，他提供10個sample questions和相關的misconceptions用於生成、問題是由misconceptions的反向工程來生成。

生成的樣本:

* 從來沒在此競賽資料中出現的misconceptions: 2種(各985個樣本)
* 僅在此競賽的資料集中出現一次的misconception: 1種(847個樣本)

1. **Training Methodology**

**Retrieval Training:**

1. 使用pre-trained SFR-Embedding-2\_R模型提取Top-200 候選項
2. 使用Qwen2.5-14B從Top-200候選項中隨機選擇30個hard negatives去做contrastive learning。重新提取Top-200 候選項。
3. 使用Qwen2.5-32B重複該過程

關鍵見解: retrieval model的目的不是識別單一正確答案，而是確保正確答案被包含在Top-25個候選項中，這種更廣泛的涵蓋對於此競賽的分數提高有非常重大的影響。

**Reranker Training:**

1. 使用retrieval model對於每個問題提取Top-25候選項
2. 從候選項中隨機選擇2~25個選項，如果之中並沒有正確答案，其中一個選項會被正確答案給替換掉
3. 使用instruction tuning從選項中去預測正確答案

作者發現改變選項數量有助於模型抓到問題的本質。

1. **Ensembling Method**

* 重複ID: 對兩個模型的分數取平均
* 唯一ID: 保留其原始分數

結果: 這樣的ensembling方法使最終分數提高了0.02。

1. **討論**
2. 數據生成依賴GPT4-o，但生成樣本的多樣性並不多(僅對於沒出現過、出現過一次的三種誤解進行生成)，且此方法極度依賴於GPT4-o的生成效果，可能會無法反映真實數據的特徵，進而導致模型的泛化能力不足。

改進方向: 使用prompt engineering來增加生成的樣本多樣性、以更多的大型語言模型來生成資料並比較其生成品質。

1. 對於重複ID的計算得分方法可以使用針對不同模型來調整權重，以達到更好的效果，並不是像現在單純取算術平均值。
2. **6th Place Solution**
3. **Synthetic Data Generation**

作者認為僅使用競賽所提供的訓練數據來訓練retrieval model的效果不好，且在使用Qwen 32B/72B instruct做LoRA pre-training與few-shot generation時發現LoRA pre-training似乎降低了模型的推理能力，推測可能是災難性遺忘，並找到了一篇論文一樣得到類似的結論，也就是few-shot generation對於此競賽更加有效。

生成細節:

* 對於最相近的訓練誤解使用3個few-shot examples。
* 每個誤解用3個數學測試並指定3個不同的random seed，對於每個誤解共計用9個數學測試。

1. **Retrievers**

使用pylatexenc套件將LaTeX的格式轉換為純文本，並刪除一些多餘的符號。

* 模型: gte-Qwen2-7B-instruct, Salesforce-SFR-Embedding-2\_R, bge-multilingual-gemma2, Qwen2.5-14B-instruct
* LoRA parameters:
  + rank = 64
  + alpha = 128
* Batch size = 32
* Negatives per sample = 5
* BNB 4-bit 量化
* 訓練策略: 2 negative mining iterations
  + Initial: 1600 steps
  + Second: 400 steps

1. **Rerankers**

訓練:

* 使用DataCollatorForCompletionOnlyLM防止提示標記影響訓練。
* 在訓練時排除所有測試的misconceptions。
* 在public score中得到0.61~0.63的分數。
* 模型: Qwen 32B-instruct

Distillation: 並沒有顯著改變。

First-Stage Reranker

* 隨機從Top-100 misconceptions中取49個negative misconceptions。
* 為了避免模型overfitting於原始的順序，將正確答案打亂在選項中。

Misconception Rephrasing

作者發現本競賽的誤解資料集敘述並沒有統一的架構，有一些敘述是在說明缺乏那些知識，有些是在說什麼是學生所相信的，因此作者在本處採用Qwen 32B-instruct模型來對此誤解進行文字重構。最終僅稍微對於public score有約0.01的進步，影響並不是很大。

Second-Stage Reranker

* 作者觀察到模型傾向於保留retrieval對於誤解的初始排序，此觀察啟發作者重新訓練一個retriever，並開發second-stage reranker來完善first-stage reranker的結果。
* Second-stage reranker選擇在first-stage reranker中排名最高的window來調整它們的排序，而作者作中試出最佳的window size為2，並使用平均的Logits作為最終的結果重新排名。

1. **討論**

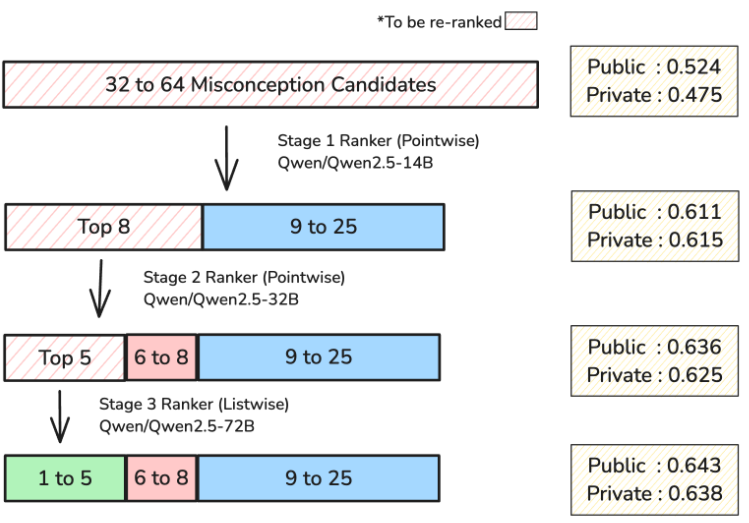
作者使用了多種embedding model來生成候選選項並進行ensemble，這點我們認為非常值得我們學習，而此處就可以嘗試如何動態分配不同embedding方法的權重，或考慮使用更多embedding方法，且在對於誤解的文句重構方面，我們也認為如果在此部分在下點工夫去分析這些誤解清單中各個誤解的屬性以及其對應的數學問題間的關係，應該可以在實際表現上有很大的突破。

同樣的在數據生成的方面有可能會因為選擇模型造成的局限以及生成出的數據品質不穩定。解決方法可以是以專家做人工校正，但這樣成本會過於龐大，或是使用自監督式的框架來過濾掉噪音，不過如何建置此框架也是另一個大工程。又或是一樣採取Prompt Engineering來對於資料的生成有更多的指引。

1. **1st Place Solution**
2. **DataMix**使用了生成數據來增加data的豐富程度，並將競賽數據與生成數據合成。  
   由GPT-4o為評審來判斷生成數據是否為可以合成的好數據，此方法大大的增加了數據量以及數據的可信度，對於表現有很好的提升。
3. **Multiple Choice**使用 3 種model 來進行Multiple Choice

* 14B Ranker: Use a fine-tuned Qwen/Qwen2.5-14B
* 32B Ranker: Use a fine-tuned Qwen/Qwen2.5-32B
* 72B Ranker: Use a fine-tuned Qwen/Qwen2.5-72B

**系統特點**

* 逐步篩選與排序的層級架構，確保每一步都能逐漸精細化候選項的質量。
* 利用多個不同大小（14B、32B、72B）的模型進行排序，一步步提升模型的推理能力和準確性。  
  

## **Train-Validation Split** 將資料依ConstructId進行切分成5份，由於ConstructName表示具體知識點或能力，比起SubjectId或QuestionId更能準確反映題目的邏輯差異。訓練時使用前4份來訓練，第5份為validation，切分後在表現上可以達到約 0.62，這是一個蠻不錯的表現。

1. **Retrievers**  
   使用了 MultipleNegativesRankingLoss，這是一種專門用於學習排名的損失函數，通過比較正負樣本間的相似度，幫助模型學會更好地區分正確答案與其他不相關選項。  
     
   專注於以下兩個數據  
   1. map@25（Mean Average Precision at 25）：用來衡量模型在前 25 名結果中的排名準確性。  
   2. recall@32（召回率@32）：評估模型是否能將相關答案包括在前 32 名中。  
   優化 recall@32 比 map@25 更重要，因為召回率會直接影響整個管道的最終表現。
2. **討論**  
   這份notebook能拿到第一名最主要的原因是在他使用了更好的資料集，也提供了如何生成的方法，他用這份資料集finetune完的模型表現非常的好，再加上他使用 MultipleNegativesRankingLoss 去讓第一次篩選選到正確答案的機率大大增加，這也讓我們看到原來還有這種方法可以解決這樣的問題。  
   在Ranking方面他甚至用了3個模型來讓每一次的Ranking都有提升，難怪他可以獲得第一名，想法太好了。

**Future Work**

我們未來會想著重於以下幾個方向以進一步提升模型的性能。首先，針對第一次檢索過程，我們可以探索使用更多元的特徵來生成嵌入，尤其是考慮將問題文本和誤解的描述一起嵌入，而不僅限於使用 ConstructName 和 SubjectName。這樣的改進有可能減少正確答案在檢索階段被錯誤排除的機會，並且能夠更精確地捕捉問題與誤解之間的語義關係。

此外，我們也可以進一步調整和優化多輪排序過程，特別是在 Multiple Choice 的每一輪中更大量的減少誤解選項。這將幫助增加模型選擇正確答案的概率，同時減少錯誤輪次的影響。增加 survivors 的數量也或許是一個可行的方向，孫然這次實驗失敗，但仍可以繼續嘗試。這樣可以有效保留更多的選項，從而提高最終結果的精確度。

另外，我們也想在實驗中更深入地探索 Chain of Thought (COT) 的使用方式，並根據具體的數學問題進行調整，讓語言模型在推理過程中能夠更好地理解誤解的來源與其背後的數學邏輯。

**Conclusion**

本次報告中，我們研究了兩種不同的的檢索與排序方法，並進行了多次實驗來優化模型在數學問題誤解識別上的表現，也嘗試將兩種方法合併，結果顯示，透過增加 n\_neighbors 的數量和調整 Multiple Choice 的排序輪次，模型的表現有了顯著提升。然而，仍有部分改進空間，特別是在第一次檢索的特徵選擇及模型推理過程中的精確度。

未來的研究將進一步探索更多樣的嵌入技術，優化每輪排序過程，並進行更多的超參數調整，期望能夠進一步提升模型的準確率，最終達到在競賽中更高的分數表現。

**References**

1. **Eedi Qwen-2.5 32B AWQ two-time retrieval** [**https://www.kaggle.com/code/takanashihumbert/eedi-qwen-2-5-32b-awq-two-time-retrieval**](https://www.kaggle.com/code/takanashihumbert/eedi-qwen-2-5-32b-awq-two-time-retrieval)
2. **Qwen14B\_Retrieval\_Qwen32B\_logits-processor-zoo** [**https://www.kaggle.com/code/jagatkiran/qwen14b-retrieval-qwen32b-logits-processor-zoo**](https://www.kaggle.com/code/jagatkiran/qwen14b-retrieval-qwen32b-logits-processor-zoo)
3. **15th Place Solution** [**https://www.kaggle.com/competitions/eedi-mining-misconceptions-in-mathematics/discussion/551501**](https://www.kaggle.com/competitions/eedi-mining-misconceptions-in-mathematics/discussion/551501)
4. **6th Place Solution** [**https://www.kaggle.com/competitions/eedi-mining-misconceptions-in-mathematics/discussion/551565**](https://www.kaggle.com/competitions/eedi-mining-misconceptions-in-mathematics/discussion/551565)
5. **1st Place Solution**[**https://www.kaggle.com/competitions/eedi-mining-misconceptions-in-mathematics/discussion/551688**](https://www.kaggle.com/competitions/eedi-mining-misconceptions-in-mathematics/discussion/551688)