# Q1: Retriever & Reranker Tuning

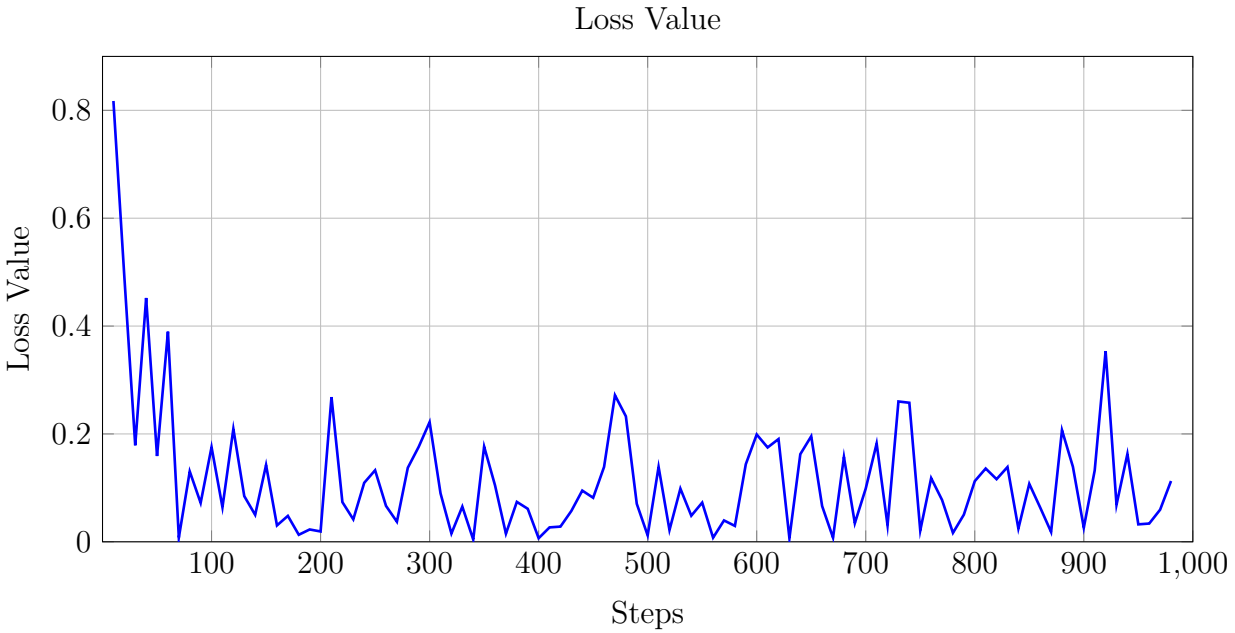
## Retriever Training Process

- Anchor: query_text (from rewrite in train.txt)
- Positive Sampling: 根據 qrels.txt 檔案，我們知道每一個 qid 對應到哪一個 positive_passage_id。從 corpus.txt 中取出這個正樣本的 positive_passage_text。InputExample 使用`texts=[f"query: {query_text}", f"passage: {positive_passage_text}"]` 建立 positive sample pairs。
- Negative Sampling: 這裡使用 In-Batch Negatives：當我們建立一個 batch_size 為 32 的 batch 時，這個 batch 裡有 32 個 (query, positive_passage) pairs。對於此 batch 中的第一個 query，P1 是 Positive，其他的 31 個 passage 都當作 Negative。
- Loss: MultipleNegativesRankingLoss (MNRL)，是一種 Contrastive Learning Loss，可以讓 query vector 和「正樣本 passage 向量」之間的相似度（通常是餘弦相似度或點積）最大化，同時讓「query 向量」和「所有 batch 內負樣本 passage 向量」之間的相似度最小化。
- batch_size: 32
- epochs: 1
- learning_rate: 1e-5

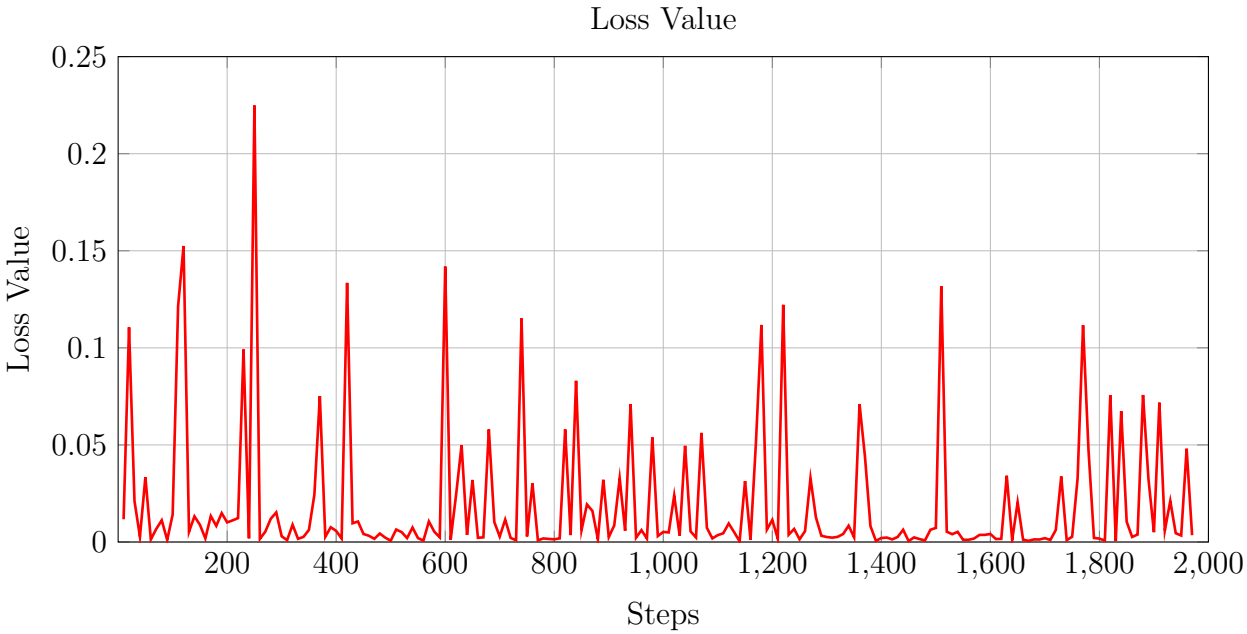
## Reranker Training Process

- Anchor: 同 Retriever
- Positive Sampling: 同 Retriever ，但是 InputExample 使用 `texts=[query_text, positive_passage_text], label=0.0`
- Negative Sampling: 對於每一個 Positive，我們從 corpus_map 中隨機挑選一個不是 Positive sample 的 negative_passage_id。
- Loss: MultipleNegativesRankingLoss (MNRL)，是一種 Contrastive Learning Loss，可以讓 query vector 和「正樣本 passage 向量」之間的相似度（通常是餘弦相似度或點積）最大化，同時讓「query 向量」和「所有 batch 內負樣本 passage 向量」之間的相似度最小化。
- batch_size: 32
- epochs: 1
- learning_rate: 1e-5

### Loss Curve of Retriever




### Loss Curve of Reranker




## Q2: Prompt Optimization

#### System Prompt

**Section A:** You are a helpful question-answering assistant. Your primary goal is to answer the user's question accurately and concisely, using *only* the information present in the context passages provided below. Do not use any external knowledge or make assumptions beyond the text. Please make a genuine effort to find the answer within the provided context. Synthesize information across passages if needed.

**Section B:** However, it is crucial that your answer is fully supported by the text. If, after careful consideration, you determine that the provided context passages definitively do not contain the information needed to answer the question, and *only* in that situation, respond with the single word : CANNOTANSWER. Otherwise, provide the best possible answer derived solely from the context.

**Section C:** Think step-by-step.

最後使用的是 Section A + Section B + Section C 的組合。

### User Prompt

```
formatted_context = ""
if not context_list:
    formatted_context = "No context provided."
else:
    for i, context in enumerate(context_list, 1):
        formatted_context += f"[Context {i}]: {context.strip()}\n\n"
prompt = f"""
Here is the user's question:
{query}

---
Here are the relevant context passages I found:
{formatted_context}
---

Based *ONLY* on the context passages provided above, please answer the user's question.
"""
```

## Different Prompt Comparison and Results

這裡都是僅有修改 System Prompt。Section B 是讓模型知道在無法回答時要回應 CANNOTANSWER，而 Section C 是讓模型進行逐步思考。以下是不同版本的結果比較：

| CosSim | With Section B | Without Section B |
|---|---|---|
| With Section C | **0.3668** | 0.3625 |
| Without Section C | 0.3626 | <u>0.3638</u> |

單純的 CoT 指令可能會產生幻覺，使得結果較差；單純的 CANNOTANSWER 指令也是不夠的，可能沒辦法進行深度推理，模型會傾向於「不作為」導致結果較差。兩者結合才能有較好的結果

# Q3: Additional Analysis

比較 train Reranker 和不 train 的結果，Prompt 不變（註：不同 Retriever DB 是獨立建置的）

| Recall@10 | Trained Retriever | Untrained Retriever |
|---|---|---|
| Trained Reranker | **0.8196** | 0.7151 |
| UnTrained Reranker | **0.8196** | 0.7151 |

| MRR@10 | Trained Retriever | Untrained Retriever |
|---|---|---|
| Trained Reranker | **0.7168** | 0.6436 |
| UnTrained Reranker | 0.7018 | 0.6335 |

| CosSim | Trained Retriever | Untrained Retriever |
|---|---|---|
| Trained Reranker | **0.3668** | 0.3615 |
| UnTrained Reranker | 0.3198 | 0.3560 |

新增了 Retriever 的訓練後 Recall@10 是有明顯提升的，但是我們可以發現 CosSim 並沒有太大的變化。另外不論是有無使用 Trained Retriever 我們發現 Reranker 去 Train 對於 MRR@10 僅有些微的提升。

最特別的結果是我們使用 Trained Retriever 但不使用 Trained Reranker 時，雖然 MRR 上升了，CosSim 反而下降了許多。這邊我的推測是即使 Retriever 可能有更好的 Recall，但可能帶來更多更難區分的 Sample，導致未經訓練的 Reranker 並無法好好區分第一名，使得 Prompt 沒有很好的效果。因此這裡需要有兩者的合作才能達到最佳效果。