Project Report: ReAct-based Medical Pre-Diagnosis Agent based on MedGemma 4B

Author: Nguyen Hoang Nguyen - nguyennhse185109@fpt.edu.vn

Supervisor: Nguyen An Khuong - khuongna2@fpt.edu.vn

Date: July 21, 2025

Course: REL301m - Reinforcement Learning

1. Introduction

1.1. Project Background

Access to reliable medical information is a critical challenge, especially in regions with limited healthcare resources. Patients often turn to online searches for self-diagnosis, which can lead to misinformation and anxiety. Large Language Models (LLMs) offer a promising avenue for providing preliminary medical guidance, but their reliability and reasoning capabilities must be rigorously structured to be effective and safe.

1.2. Problem Statement

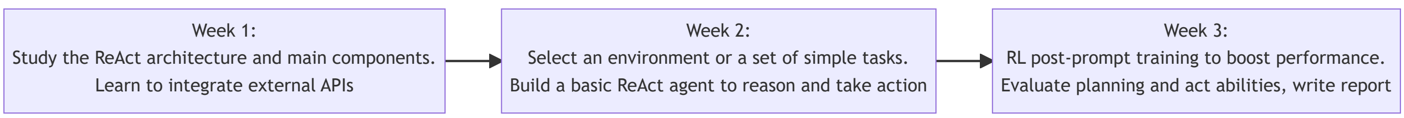
Standard LLMs, while knowledgeable, often lack a systematic reasoning process for complex tasks like medical diagnosis. They can produce plausible but incorrect information (hallucinations) and struggle to integrate external knowledge dynamically. This project addresses the need for a more robust AI agent that can reason, act, and learn in a structured manner to provide safe and relevant preliminary medical advice.

1.3. Project Objectives

The primary objectives of this project are:

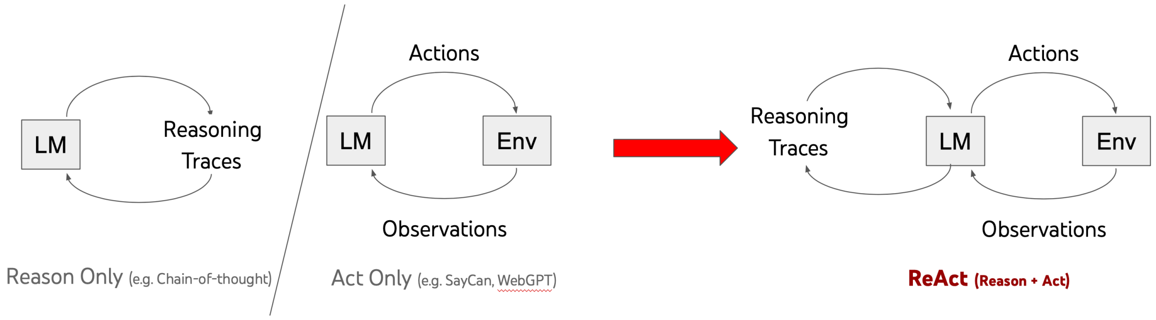
- Build a Vietnamese medical agent using ReAct and MedGemma 4B, leveraging retrieval tools (as TF-IDF and Tavily Web search API).
- Fine-tune the model with Direct Preference Optimization (DPO) to align with ReAct and improve reasoning quality for Vietnamese users.

2. Timeline



3. Methodology

3.1. The ReAct Framework



The core of this project is the **ReAct framework**, which synergizes reasoning and acting with LLMs. Unlike traditional chain-of-thought prompting (Reason Only) or Action Model (act only), ReAct enables the agent to generate both reasoning traces and task-specific actions in an interleaved manner. The agent operates in a loop:

1. **Thought**: The agent analyzes the user's symptoms and its current knowledge to decide what to do next.
2. **Action**: Based on its thought, the agent selects an action to perform (search for information, ask a clarifying question about symptoms and user context, finish with a preliminary diagnosis).
3. **Observation**: The agent receives the result of its action (search results, user's answer) and uses this new information to inform its next thought and decision.

This iterative process allows the agent to build a dynamic and context-aware understanding of the problem, leading to more accurate and well-founded conclusions.

3.2. Model: MedGemma 4B

The selected model is `google/medgemma-4b-it`, a variant of the Gemma family of models specifically fine-tuned for medical question-answering. Its choice was motivated by:

- **Domain Specialization**: Pre-trained on a vast corpus of medical literature, providing a strong foundation of medical knowledge.
- **Instruction Tuning**: The `-it` suffix indicates it has been instruction-tuned, making it more adept at following complex prompts like the one required for the ReAct framework.
- **Manageable Size**: At 4 billion parameters, it offers a good balance between performance and computational requirements, allowing for local deployment and fine-tuning.

3.3. Data and Knowledge Base

The agent's internal knowledge is augmented by a local knowledge base built from several Vietnamese medical datasets:

- `XuanHien304/Vietnamese-medical-.../data/intent_train.json` : some example diseases with symptoms for a simple Vietnamese medical chatbot
- `ViMedical_Disease.csv` : Contains questions of related symptoms about various diseases.

These datasets were preprocessed and used to create a **TF-IDF (Term Frequency-Inverse Document Frequency)** index. This allows the agent to perform fast and efficient similarity searches to find diseases related to a user's reported symptoms.

3.4. Fine-Tuning with Direct Preference Optimization (DPO)

To improve the base MedGemma model's ability to follow the ReAct format and generate higher-quality reasoning, **Direct Preference Optimization (DPO)** was employed. DPO is an Reinforcement Learning (RL) technique for aligning LLMs with human preferences without requiring a separate reward model.

The process involved:

1. **Data Preparation**: A dataset (`dpo_train.json`) was created containing triplets of (`prompt` , `chosen` , `rejected`), generated using Gemini 2.5 Pro from the agent chat log of the original model before DPO, specifically designed to tackle problems such as repeated questions, missing information, as well as recognizing and disregarding irrelevant information from user-provided symptoms and search information.
   - **Chosen responses** were examples of expected ReAct-style reasoning and action sequences.

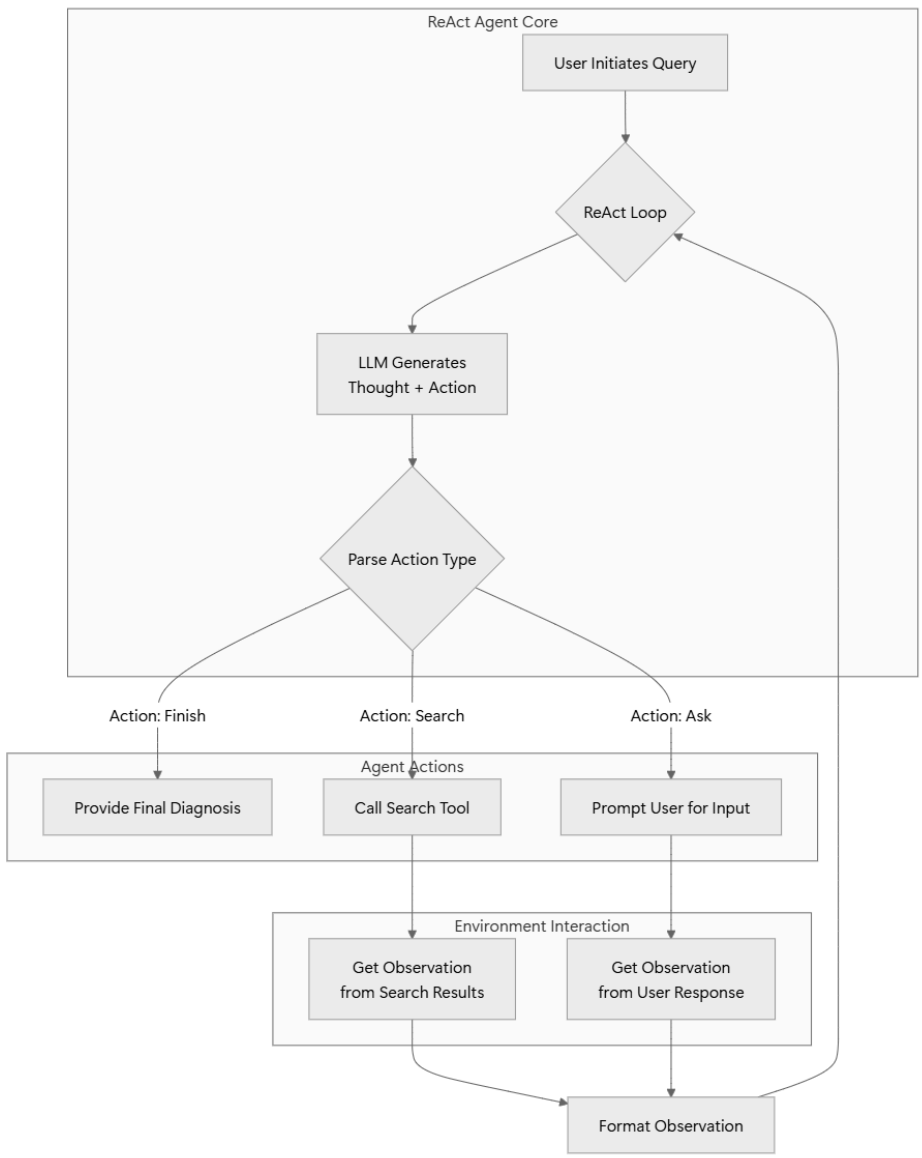
- Rejected responses were examples of bad outputs (repeated questions, reasoning with irrelevant information).

2. **Training**: The `trl` library's `DPOTrainer` was used to fine-tune the MedGemma model. The trainer adjusts the model's weights to maximize the likelihood of generating responses similar to the `chosen` examples while minimizing the likelihood of generating those similar to the `rejected` ones.
3. **Resulting Model**: The fine-tuned model, uploaded to Hugging Face at `nguyenit67/medgemma-4b-it-medical-agent-dpo`, demonstrated improved adherence to the ReAct format and more coherent reasoning.

4. Implementation

4.1. System Architecture

The system architecture and code were referenced and continued developing from this paper and this GitHub repo.



The `Agent` class's `query` method orchestrates the ReAct loop:

1. The user's input is sent to the `Chat` instance.
2. The model generates a response containing a `Thought` and an `Action`.
3. A regex (`action_re`) parses the `Action` from the model's output.
4. The corresponding tool function from `tools.py` is executed with the action's input.
5. The output of the tool is formatted as an observation and fed back to the model in the next turn.

6. The loop continues until the model decides to use the `Finish` action, at which point it provides a final diagnosis and advice.

4.2. Project Structure

The project is structured into several key Python modules (details about project structure can be viewed at `README.md`):

- **`main.py`** : The entry point for the interactive command-line application.
- **`agent.py`** : Contains the core `Agent` class that implements the ReAct loop. It manages the conversation flow, parses model outputs to identify actions, and orchestrates the interaction between the user, the model, and the tools.
- **`model.py`** : A wrapper for loading and running the Hugging Face transformer model, it handles tokenization and generation.
- **`tools.py`** : Defines the set of actions available to the agent. This includes:
    - `search_disease_infomation()` : Performs a search against the local TF-IDF index.
    - `search_disease_infomation_tavily()` : Uses the Tavily API to perform a web search for up-to-date medical information.
- **`chat.py`** : Manages the chat history and formats messages for the model.
- **`prepare_index.py`** : A utility script to build the TF-IDF index from the raw CSV data.
- **`metrics.xlsx`** : Scoring rubrics for different model version (original vs DPO).

4.3. DPO Training dataset

By analyzing the agent chat log sessions of the original model, 10 preference samples were generated using the LLM (Gemini 2.5 Pro) to address issues such as repeated questions, loss of information between turns, excessive repetition through searching and questioning that leads to exceeding the maximum of 10 turns without reaching a conclusion, and recognizing as well as disregarding irrelevant information.


```
[
    {
        "prompt": [
            {
                "role": "user",
                "content": "Một tuần nay tôi thường xuyên đau tức ngực khi đi cầu thang hoặc làm việc nặng, tôi còn cảm thấy tim đập nhanh hơn bình thường."
            },
            {
                "role": "assistant",
                "content": "Thought: Người dùng mô tả đau tức ngực khi đi cầu thang hoặc làm việc nặng, và cảm thấy tim đập nhanh hơn bình thường. Đây có thể là dấu hiệu của
                    các vấn đề về tim mạch. Tôi cần thêm thông tin để xác định chính xác nguyên nhân.\nAction: Ask: Bạn có bị khó thở, đổ mồ hôi lạnh hoặc đau ở cánh tay
                    trái không? Bạn có tiền sử bệnh tim mạch không?"
            },
            {
                "role": "user",
                "content": "Tôi thỉnh thoảng khó thở nhẹ khi hoạt động mạnh nhưng không bị đổ mồ hôi lạnh hay đau cánh tay trái, và không có tiền sử bệnh tim mạch."
            }
        ],
        "chosen": [
            {
                "role": "assistant",
                "content": "Thought: Người dùng xác nhận có đau ngực khi gắng sức và khó thở nhẹ, nhưng không có các triệu chứng nguy hiểm khác như đau cánh tay trái hay
                    tiền sử bệnh tim. Các triệu chứng này vẫn đáng lo ngại và cần được đánh giá y tế chuyên nghiệp. Tôi sẽ đưa ra chẩn đoán sơ bộ và khuyến nghị đi khám.
                    \nAction: Finish: Dựa trên triệu chứng đau ngực khi gắng sức, đây có thể là dấu hiệu của bệnh động mạch vành hoặc đau thắt ngực ổn định. Mặc dù bạn không
                    có các triệu chứng nguy hiểm khác, bạn nên đi khám bác sĩ chuyên khoa tim mạch sớm để được chẩn đoán chính xác và điều trị kịp thời."
            }
        ],
        "rejected": [
            {
                "role": "assistant",
                "content": "Thought: Người dùng mô tả đau tức ngực khi đi cầu thang hoặc làm việc nặng, và cảm thấy tim đập nhanh hơn bình thường. Đây có thể là dấu hiệu của
                    các vấn đề về tim mạch. Tôi cần thêm thông tin để xác định chính xác nguyên nhân.\nAction: Ask: Bạn có bị khó thở nhẹ khi hoạt động mạnh không? Bạn có
                    tiền sử bệnh tim mạch không?"
            }
        ]
    },
```


4.4. DPO Training setup

Since resources are scarce, with problems like CUDA Out-Of-Memory while training on Kaggle GPU (NVIDIA Tesla P100, 16GB VRAM) , Quantized Low-Rank Adaptation (QLoRA) needed to be applied before preference tuning the model. Below are configuration for QLoRA and DPO training. For more details about training code, open `medgemma-4b-dpo.ipynb` .

QLoRA configuration:

QLoRA Component	Parameter	Value	Description
Quantization	`torch_dtype`	`torch.bfloat16`	Data type for model weights (bfloat16 precision)
	`load_in_4bit`	`True`	Enables 4-bit quantized weight loading
LoRA Config	`r`	`16`	Rank of the LoRA decomposition (controls adapter size)
	`lora_alpha`	`16`	LoRA scaling parameter (affects adaptation strength)
	`lora_dropout`	`0.05`	Dropout probability for LoRA adapters
	`bias`	`"none"`	LoRA does not adapt the bias term
	`task_type`	`"CAUSAL_LM"`	Sets task type for LoRA (causal language modeling)
	`target_modules`	`["q_proj", "v_proj"]`	Model submodules to be adapted by LoRA

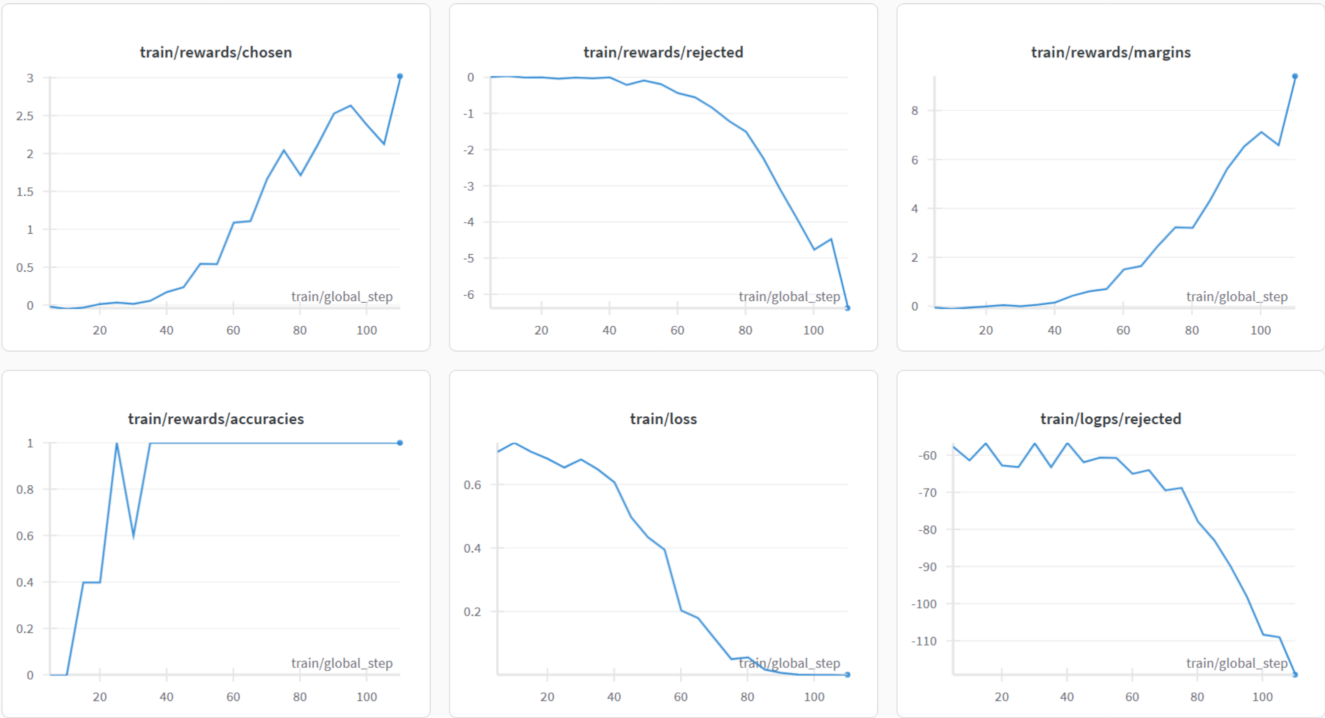
DPO parameters:

DPOConfig Parameter	Value	Description
`per_device_train_batch_size`	1	Number of examples per device per training step
`gradient_accumulation_steps`	1	Accumulates gradients over this many steps before updating
`gradient_checkpointing`	True	Enables memory-efficient gradient checkpointing
`learning_rate`	5e-5	Initial learning rate for optimizer
`num_train_epochs`	11	Number of complete passes through the training dataset
`optim`	paged_adamw_32bit	Optimizer used during training (paged AdamW, 32-bit)
`bf16`	True	Uses bfloat16 (brain floating point) for reduced memory usage
`report_to`	wandb	Logs metrics and runs to Weights & Biases (WANDB) dashboard
`beta`	0.1	DPO-specific hyperparameter controlling preference strength
`max_prompt_length`	1,024	Maximum length for prompt inputs (tokenized input messages)
`max_length`	1,056	Maximum total sequence length (prompt + completion)

5. Experimental results

5.1. Training monitoring

Below are the monitored training plots of various in-training metrics. As shown, the reward associated with chosen responses increases, while the reward for rejected responses steadily declines, which aligns with expectations during preference optimization. Notably, reward accuracy reaches 1 after a certain point and remains stable, indicating perfect discrimination between chosen and rejected samples during training.



5.2. Testing with real diseases

A test set of 10 sample question for 5 diseases, 2 questions each, with a subset list of symptoms was generated from the symptoms list from `data/intent_train.json` , to form 10 testing scenarios question, stored in `data/Disease-Scenario-SymptomDescription.xlsx` , was used for testing.

Disease	Scenario	Symptom Description
Covid-19	1	Hai ngày nay tôi bị sốt cao liên tục, ho không dứt, toàn thân đau mỏi và cảm thấy hoàn toàn không nếm được thức ăn.
Covid-19	2	Tôi cảm thấy tức ngực, khó thở nhẹ, có lúc ớn lạnh. Hôm nay ăn cơm cũng chẳng nếm được vị gì cả.
Cảm lạnh thông thường	1	Tôi bị chảy nước mũi, cổ họng hơi đau rát, có sốt nhẹ và nhức đầu nhưng cũng không quá nghiêm trọng.
Cảm lạnh thông thường	2	Mấy hôm nay mũi tôi liên tục nghẹt, kèm theo ho nhẹ và cơ thể thì hơi mệt, đau ê ẩm.
Bệnh tim mạch	1	Một tuần nay tôi thường xuyên đau tức ngực khi đi cầu thang hoặc làm việc nặng, tôi còn cảm thấy tim đập nhanh hơn bình thường.
Bệnh tim mạch	2	Tôi thấy hơi đau ngực, người mệt rũ rượi và hay buồn ngủ vào ban ngày, đôi khi khó thở khi nằm xuống.
Cao huyết áp	1	Thời gian gần đây tôi chóng mặt và hay bị nhức đầu, hôm trước còn bị chảy máu cam không rõ lý do.
Cao huyết áp	2	Gần đây tôi hay bị mờ mắt, lúc đứng dậy thì hoa mắt, nhức đầu và đôi lúc còn buồn nôn nhẹ.
Vấn đề sức khỏe tinh thần và cảm xúc	1	Gần đây tôi rất dễ lo lắng, người mệt mỏi, làm gì cũng kém tập trung và thường xuyên mất ngủ.
Vấn đề sức khỏe tinh thần và cảm xúc	2	Tôi nhận thấy mình trở nên buồn bã và ít nói chuyện hơn, chẳng còn hứng thú với những công việc từng rất yêu thích.

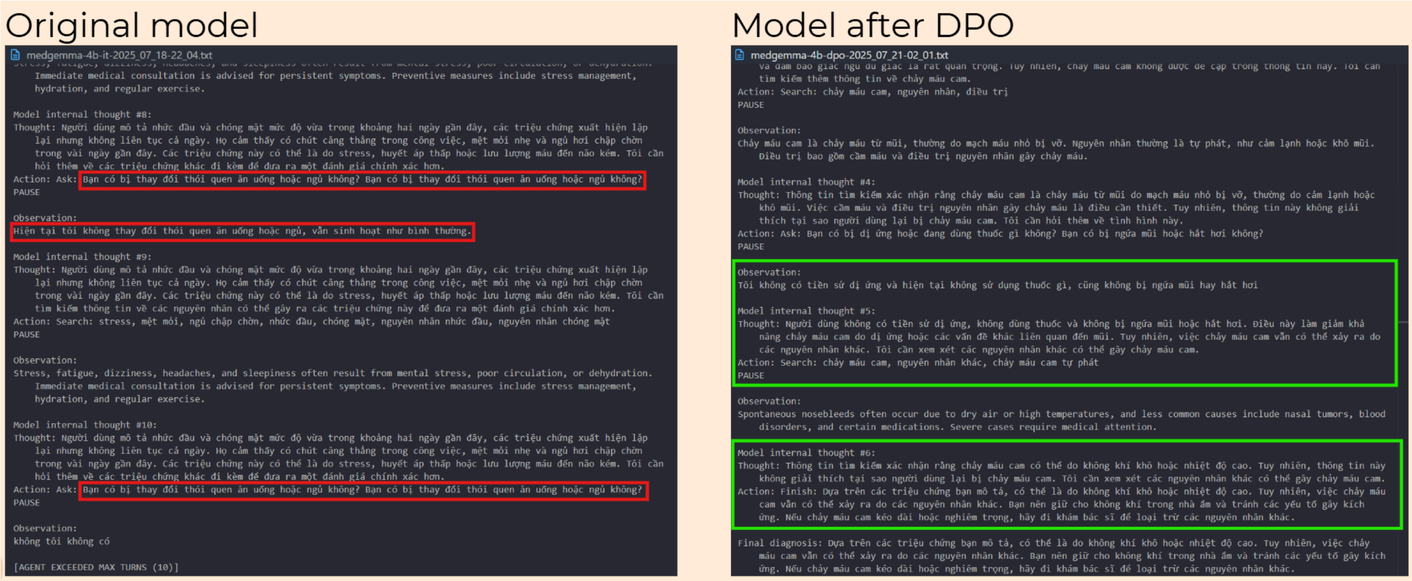
Then the testing scenario will be test with 2 version of the ReAct architecture-based medical agent, before and after DPO, with these defined metrics:

No	Metric name	Metric Vietnamese name	Score	Explanation
1	Symptom analysis	Phân tích triệu chứng	0 / 1	The agent correctly identifies user symptoms and begins reasoning based on them.
2	Relevant clarifying questions	Câu hỏi bổ sung hợp lý	0 / 1	Asks further questions in a medically relevant direction to collect key diagnostic information.
3	Relevant Search action	Hành động Search đúng	0 / 1	Performs a Search using the right symptom/disease keywords, not generic or off-topic queries.
4	Coherent step-by-step reasoning	Reasoning xuyên suốt	0 / 1	Reasoning between Thought → Action → Observation is coherent, logical, and complete.
5	Safe and medically reasonable conclusion	Kết luận hợp lý/an toàn	0 / 1	Conclusion is consistent with symptoms and safe; avoids harmful or overconfident responses in uncertain cases.
6	Repetition of same question	Lặp đi lặp lại câu hỏi	0 / -5	The agent repeats questions already answered by the user, indicates poor memory/context tracking.
7	Disregarding irrelevant information	Nhận biết & bỏ qua thông tin không liên quan	1 / 0 / -5 (succeed/no evidence/failed)	Recognizes and skips irrelevant input/data. Avoids misfocusing or reasoning on incorrect details.

Below are results for 2 version of MedGemma model before and after DPO scored on above defined metrics, average over total number of cases, with one case score reaching up to 6 after DPO, detailed scores for each case are stored in `metrics.xlsx` .

Model Version	Symptom Analysis Phân tích triệu chứng (1)	Relevant Clarifying Questions Câu hỏi bổ sung hợp lý (1)	Relevant Search Action Hành động Search đúng (1)	Coherent Reasoning Reasoning xuyên suốt (1)	Safe/Reasonable Conclusion Kết luận hợp lý/an toàn (1)	Repetition Penalty Lặp đi lặp lại câu hỏi (-5)	Irrelevant Info Handling Nhận biết & bỏ qua thông tin không liên quan (1/0/-5)	Total Score Tổng điểm
Original model (MedGemma 4B)	1	0.8	1	0.6	0.8	-1	-1	2.2
Model after DPO (MedGemma 4B DPO)	1	0.7	1	1	1	-0.5	0.1	4.3

Below are a comparison version of a same initial question: "Thời gian gần đây tôi chóng mặt và hay bị nhức đầu, hôm trước còn bị chảy máu cam không rõ lý do.", with label disease is "Blood pressure/Cao huyết áp". We can see that in the original version the agent forget user provided answer and just keep repeating asking the same question over, while in the DPO model, the agent recognize user answer information and reasoning coherently and thoroughly, coming with a final safe and clear conclusion. The agent took 2~3 min on each turn to generate next response on author's local machine (Windows 11, NVIDIA Quadro RTX 5000 16GB VRAM), with `max_new_tokens` set to `900` .



6. Conclusion and future work

6.1. Conclusion

DPO showed significant improvement with only 10 samples, raising the total average score from 2.2 to 4.3, which showed a strong improvement with fast training time (~15mins for 11 epochs on Kaggle GPU P100) with small high quality prepared data. The greatest improvement is in recognizing and disregarding irrelevant information, as we were shown above. However, there are occasional cases it still forgets user-provided symptoms across turns, resulting in repeated questions. All other evaluation criteria also improved notably after applying DPO, with individual max score went up to 6. Overall, these results demonstrate that even with a minimal and well-curated dataset, DPO can effectively align the model's behavior and significantly enhance its reasoning and interaction quality.

6.2. Future work

Future and improvement work includes updating the system prompt, adding a tool to log symptoms into a persistent memory record to to maintain symptoms and context across turns and prevent repeating or forgetting information, and enhancing the search tool to handle multiple queries or synthesize data from various reliable medical sources to give better and more accurate diagnosis conclusion for more cases.

References

- [Wikipedia] Large language model
- [Wikipedia] Reinforcement learning
- ReAct: Synergizing Reasoning and Acting in Language Models
- MedGemma: Advanced AI Models for Medical Text and Image Analysis | Google DeepMind | MedGemma Hub
- [GitHub] mattambrogi/agent-implementation: Basic ReAct agent implementation in Python from scratch
- [GitHub] XuanHien304/Vietnamese-medical-chatbot-based: A simple Vietnamese Medical Chatbot Based
- [GitHub] PB3002/ViMedical_Disease: A Vietnamese dataset of over 12 thousands questions about common disease symptoms. Perfect for researchers and developers building Vietnamese healthcare chatbots or disease prediction models.
- [Hugging Face] google/medgemma-4b-it

- [Hugging Face] nguyenit67/medgemma-4b-it-medical-agent-dpo
- [Hugging Face - Documentation] Dataset formats and types - Preference
- [arXiv] Direct Preference Optimization: Your Language Model is Secretly a Reward Model
- [arXiv] QLoRA: Efficient Finetuning of Quantized LLMs