Project Report: ReAct-based Medical Pre-Diagnosis Agent

Author: NguyenNHSE185109 **Date**: July 20, 2025 **Course**: REL301m - Reinforcement Learning

1. Introduction

1.1. 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 assistant that can reason, act, and learn in a structured manner to provide safe and relevant preliminary medical advice.

1.3. Objectives

The primary objectives of this project are:

- To implement a medical pre-diagnosis agent using the **ReAct (Reasoning and Acting)** framework.
- To leverage a specialized medical LLM, **MedGemma-4B-IT**, for its domain-specific knowledge.
- To enhance the agent's capabilities by integrating external tools for information retrieval (local TF-IDF search and web search using Tavily

API).
- To fine-tune the base model using **Direct Preference Optimization (DPO)** to align its behavior with the ReAct framework and improve its reasoning quality.
- To create a system that interacts with users in Vietnamese, making it accessible to a local audience.

2. Methodology

2.1. 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, 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 query and its current knowledge to form a plan.
2. **Action**: Based on its thought, the agent selects an action to perform (e.g., search for information, ask a clarifying question).
3. **Observation**: The agent receives the result of its action (e.g., search results, user's answer) and uses this new information to inform its next thought.

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.

2.2. Model: MedGemma-4B-IT

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.

2.3. Data and Knowledge Base

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

- `ViMedical_Disease.csv` : Contains descriptions of various diseases.
- `symptoms.csv` : A comprehensive list of symptoms.

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.

2.4. 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 a 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_response` , `rejected_response`).
    - **Chosen responses** were examples of high-quality ReAct-style reasoning and action sequences.
    - **Rejected responses** were examples of less desirable outputs (e.g., unstructured thoughts, incorrect actions).
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, `nguyenit67/medgemma-4b-it-medical-agent-dpo`, demonstrated improved adherence to the ReAct format and more coherent reasoning.

3. Implementation

3.1. System Architecture

The project is structured into several key Python modules:

- **`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, generation, and quantization settings (e.g., 4-bit/8-bit loading) to manage memory usage.
- **`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.
- **`main.py`** : The entry point for the interactive command-line application.

3.2. The ReAct Loop in Practice

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 or recommendation.

4. Results and Evaluation

4.1. Qualitative Analysis

Analysis of conversation logs (e.g., `logs/medgemma-4b-it-2025_07_20-02_46.txt`) reveals the agent's effectiveness. In a typical session, the agent demonstrates a clear, logical progression:

- It correctly identifies and consolidates user-reported symptoms.
- It uses the `Search` tool to gather information about potential conditions.
- **Crucially, it demonstrates the ability to filter and ignore irrelevant information from search results.** For example, when a user reports symptoms without a fever, the agent correctly dismisses search results related to influenza or COVID-19, even if they are returned by the search tool.
- It uses the `Ask` tool to gather missing information, such as symptom duration or pre-existing conditions.
- The final `Finish` action provides a well-reasoned summary and a safe recommendation (e.g., "consult a doctor for an accurate diagnosis").

4.2. Impact of DPO Fine-Tuning

The fine-tuned model (`...-medical-agent-dpo`) showed marked improvements over the base `medgemma-4b-it` model:

- **Better Format Adherence**: The DPO model was more consistent in generating responses that strictly followed the `Thought: ... Action: ...` format.

- **Improved Reasoning**: The thoughts generated were more coherent and directly relevant to the user's query and the observations from the tools.
- **Reduced Hallucinations**: The fine-tuning process appeared to ground the model more effectively, reducing the incidence of irrelevant or factually incorrect statements in its reasoning process.

5. Conclusion and Future Work

5.1. Conclusion

This project successfully implemented a ReAct-based medical pre-diagnosis agent capable of sophisticated reasoning and tool use. By combining the specialized knowledge of the MedGemma-4B-IT model with the structured reasoning of the ReAct framework and the power of DPO fine-tuning, the agent provides relevant, context-aware, and safe preliminary medical guidance in Vietnamese. The agent's ability to dynamically search for and filter information marks a significant step towards more reliable AI-powered healthcare assistants.

5.2. Future Work

- **Expansion of Knowledge Base**: Incorporate more diverse and comprehensive Vietnamese medical datasets.
- **Multi-turn Tool Use**: Enhance the agent to use multiple tools within a single turn for more complex queries.
- **Integration of Structured Data**: Allow the agent to query structured databases (e.g., drug interaction databases).
- **Clinical Evaluation**: Conduct a formal evaluation of the agent's recommendations against those of medical professionals.
- **UI Development**: Create a user-friendly web or mobile interface to improve accessibility.