

Institut National des Sciences Appliquées de Toulouse

# Optimizing and Adapting Language Models for Domain-Specific Task

End-of-studies Apprenticeship Report

Minh Duy Nguyen



Bachelor's Degree in Computer Science and Engineering

Supervised by: Milad Mozafari (Torus AI), David Bertoin (INSA Toulouse)

Ngày 14 tháng 1 năm 2026

# Chương 1

## Introduction

### 1.1 General Context

#### 1.1.1 Sự phát triển của Mô hình Ngôn ngữ Lớn

Trong thập kỷ qua, lĩnh vực Trí tuệ nhân tạo (Artificial Intelligence - AI) và Xử lý ngôn ngữ tự nhiên (Natural Language Processing - NLP) đã chứng kiến những bước tiến đột phá, đặc biệt kể từ sự ra đời của kiến trúc Transformer [Vaswani et al., 2017]. Các Mô hình Ngôn ngữ Lớn (Large Language Models - LLMs) như GPT-4 [OpenAI, 2023], Claude 3 [Anthropic, 2024], Gemini 1.5 [Google, 2024], cùng các mô hình mã nguồn mở như Llama 3 [Meta, 2024] và Qwen 2.5 [Alibaba, 2024] đã chứng minh khả năng vượt trội trong việc hiểu ngữ cảnh, sinh văn bản và thực hiện suy luận logic trên đa dạng tác vụ.

Theo báo cáo của McKinsey Global Institute (2024), thị trường AI tạo sinh (Generative AI) dự kiến đạt giá trị 4.4 nghìn tỷ USD vào năm 2030. Tuy nhiên, một khảo sát từ Gartner (2024) chỉ ra rằng \*\*67% doanh nghiệp\*\* gặp khó khăn trong việc triển khai LLMs cho các tác vụ chuyên biệt do các hạn chế về độ chính xác và chi phí vận hành.

#### 1.1.2 Khoảng cách giữa Nghiên cứu và Ứng dụng Công nghiệp

Việc chuyển giao các mô hình ngôn ngữ từ môi trường nghiên cứu sang ứng dụng thực tiễn trong công nghiệp (Industrial Deployment) đang đối mặt với những thách thức đáng kể. Mặc dù các mô hình nền tảng (Foundation Models) sở hữu lượng tri thức tổng quát khổng lồ, chúng thường gặp hai nhóm hạn chế nghiêm trọng khi giải quyết các bài toán chuyên biệt (Domain-specific tasks):

##### \*\*Khoảng cách về Tri thức (Knowledge Gap):\*\*

Trong các lĩnh vực quan trọng như y tế, tài chính và bảo hiểm, thông tin thường nằm trong các tài liệu phức tạp (báo cáo tài chính, bảng danh mục kỹ thuật y tế CCAM) và thay đổi liên tục. Các LLMs đối mặt với ba vấn đề chính:

- \*\*Dữ liệu tĩnh (Static Knowledge):\*\* Tri thức của mô hình bị giới hạn bởi thời điểm huấn luyện (knowledge cutoff), không thể cập nhật thông tin mới mà không cần huấn luyện lại [Lewis et al., 2020].
- \*\*Thiếu dữ liệu nội bộ (Private Data Inaccessibility):\*\* Mô hình không thể truy cập các tài liệu bảo mật của tổ chức như hợp đồng, quy trình nội bộ, hay hồ sơ bệnh nhân.
- \*\*Hiện tượng ảo giác (Hallucination):\*\* LLMs có xu hướng sinh ra thông tin sai lệch một cách tự tin [Ji et al., 2023], điều không thể chấp nhận trong các quyết định y khoa hay tài chính, nơi yêu cầu độ chính xác gần như tuyệt đối.

##### \*\*Khoảng cách về Hiệu quả và Hành vi (Efficiency & Behavior Gap):\*\*

Đối với các tác vụ yêu cầu mô hình tuân thủ một kịch bản hành vi cụ thể, một phong cách ngôn ngữ đặc thù (ví dụ: tư vấn viên tâm lý, chuyên gia phân tích), hoặc triển khai trên hạ tầng phần cứng giới hạn, việc sử dụng các mô hình khổng lồ (hàng trăm tỷ tham số) là không tối ưu về chi phí và độ trễ. Theo Hoffmann et al. (2022), chi phí inference của GPT-4 có thể lên tới \*\*\$0.06/1000 tokens\*\*, khiến việc triển khai quy mô lớn trở nên không khả thi cho nhiều doanh nghiệp vừa và nhỏ.

### 1.1.3. Động lực nghiên cứu

Xuất phát từ thực tế trên, luận văn này tập trung nghiên cứu và triển khai các kỹ thuật tiên tiến nhằm \*\*tối ưu hóa và thích ứng LLMs\*\* cho các miền dữ liệu đặc thù\*\*, với hai hướng tiếp cận chính:

1. \*\*Retrieval-Augmented Generation (RAG):\*\* Tích hợp tri thức bên ngoài để giải quyết Knowledge Gap.
2. \*\*Parameter-Efficient Fine-Tuning (PEFT):\*\* Thích ứng hành vi mô hình để giải quyết Behavior Gap.

## 1.2 Internship Environment

This master's thesis was completed at Torus AI, a technology company with the mission "Intelligence for Life," specializing in providing advanced AI solutions to improve quality of life and business efficiency.

I worked in the Research and Development Team (R&D Team - Torus Lab) as a Machine Learning Engineer Alternant. At Torus AI, the R&D department acts as a bridge between the latest academic research (State-of-the-Art) and commercial products. The team's main task is to continuously explore emerging Generative AI technologies, assess their feasibility, and build functional prototypes (PoCs) to verify their effectiveness before integration into the main product system.

The R&D work environment demands flexible thinking: not just using existing APIs, but delving into customizing architecture, optimizing data processing pipelines, and quantitatively evaluating technical solutions.

\*\*Vai trò của tác giả:\*\* Kỹ sư Học máy tập sự (Machine Learning Engineer Alternant) trong đội ngũ Nghiên cứu và Phát triển (R&D Team - Torus Lab), với nhiệm vụ: - Khảo sát các công nghệ Generative AI mới nổi - Dánh giá tính khả thi và xây dựng các bản mẫu chức năng (Functional Prototypes/PoC) - Tích hợp các giải pháp vào hệ thống sản phẩm

\*\*Tài nguyên được cung cấp:\*\* - Hạ tầng GPU: NVIDIA A100 (40GB), T4 (16GB) trên nền tảng đám mây - Dữ liệu doanh nghiệp thực tế: Tài liệu y tế (CCAM, NGAP), báo cáo tài chính bảo hiểm - API access: OpenAI GPT-4, Google Gemini, Anthropic Claude

## 1.3 Problem Statement

During our work in the R&D department, we identified two core problems that needed to be addressed when applying GenAI in practice, corresponding to two main technical approaches:

Problem 1: Integration of External Knowledge from Complex Unstructured Data

Partner businesses (such as insurance companies, healthcare facilities) possess large amounts of data in the form of PDF documents containing text, tables, and images. Traditional RAG (Retrieval-Augmented Generation) methods based on plain text (text-only) fail to understand the semantics of complex tables or visual information. The question is: How to build a Multimodal

RAG pipeline capable of accurately parsing, indexing, and retrieving information from these mixed documents to support decision-making (e.g., medical refund code lookup, financial data analysis)?

Problem 2: Behavioral Adaptation and Resource Optimization for Small Models (Behavioral Adaptation & Efficiency)

For applications requiring high interactivity, counseling, or entertainment (such as psychological counseling chatbots or Tarot Readers), the requirement is not only for information accuracy but also for consistency in tone and style and rule-based reasoning. Using large models (like GPT-4) via APIs is both costly to operate and difficult to fully control behavior. The question is: Is it possible to fine-tune small language models (such as Qwen, Llama < 7B parameters) using parameter optimization techniques (PEFT/LoRA) and quantization so that they achieve inference capabilities and writing styles comparable to large models, but can be run locally at low cost?

#### 1.4 1.3. Phát biểu vấn đề nghiên cứu (Problem Statement)

##### 1.4.1 1.3.1. Các câu hỏi nghiên cứu (Research Questions)

Trong quá trình làm việc tại bộ phận R&D, chúng tôi đã xác định hai nhóm vấn đề cốt lõi cần giải quyết khi áp dụng Generative AI vào thực tế, được hình thức hóa thành các câu hỏi nghiên cứu sau:

—  
\*\*RQ1 (Research Question 1):\*\* \*Làm thế nào để xây dựng một pipeline Multimodal RAG có khả năng phân tích cú pháp (parsing), lập chỉ mục (indexing) và truy xuất (retrieval) chính xác thông tin từ các tài liệu hỗn hợp (văn bản, bảng biểu, hình ảnh) để hỗ trợ ra quyết định trong lĩnh vực y tế và tài chính?\*

\*\*Bối cảnh:\*\* Các doanh nghiệp đối tác sở hữu lượng lớn dữ liệu dưới dạng tài liệu PDF chứa văn bản, bảng biểu (tables) và hình ảnh (charts/images). Các phương pháp RAG truyền thống dựa trên văn bản thuần túy (text-only) thất bại trong việc hiểu ngữ nghĩa của bảng biểu phức tạp hoặc thông tin thị giác.

—  
\*\*RQ2 (Research Question 2):\*\* \*Liệu có thể tinh chỉnh (Fine-tuning) các mô hình ngôn ngữ nhỏ (Small Language Models - SLMs, < 7B tham số) bằng các kỹ thuật tối ưu tham số (PEFT/LoRA) và lượng tử hóa (Quantization) để đạt được khả năng suy luận và văn phong tương đương các mô hình lớn, nhưng có thể chạy cục bộ với chi phí thấp?\*

\*\*Bối cảnh:\*\* Dối với các ứng dụng yêu cầu tính tương tác cao, mang tính chất tư vấn (như chatbot tư vấn hoặc hệ thống suy luận dựa trên quy tắc), yêu cầu không chỉ là độ chính xác của thông tin mà còn là sự nhất quán trong văn phong (Tone & Style) và khả năng suy luận theo quy tắc (Rule-based reasoning).

##### 1.4.2 1.3.2. Giả thuyết nghiên cứu (Research Hypotheses)

Dựa trên các câu hỏi nghiên cứu, chúng tôi đề xuất các giả thuyết sau:

\*\*H1 (Hypothesis 1):\*\* \*Việc kết hợp Hybrid Search (Dense Retrieval + Sparse Retrieval) với Cross-Encoder Reranking và chiến lược Parent-Child Indexing sẽ cải thiện đáng kể độ chính xác truy xuất (Retrieval Accuracy) và độ trung thực (Faithfulness) của hệ thống RAG so với phương pháp Dense Search đơn thuần, đặc biệt trên dữ liệu bảng biểu và mã code y tế.\*

\*\*H2 (Hypothesis 2):\*\* \*Các mô hình ngôn ngữ nhỏ (1.5B - 3B tham số) sau khi được Fine-tuning bằng QLoRA trên dữ liệu tổng hợp (Synthetic Data) chất lượng cao có thể học được văn phong (Style) và định dạng đầu ra (Output Format) của tác vụ chuyên biệt, nhưng sẽ gặp hạn chế về khả năng suy luận phức tạp (Complex Reasoning) so với các mô hình lớn hơn (>70B tham số).\*

## 1.5 Objectives & Contributions

## 1.6 Thesis Structure

## Chương 2

# Cơ sở lý thuyết

Chương này cung cấp nền tảng lý thuyết và toán học cần thiết cho các phương pháp được áp dụng trong luận văn. Chúng tôi trình bày có hệ thống từ kiến trúc cơ bản của LLMs đến các kỹ thuật thích ứng tiên tiến, bao gồm Retrieval-Augmented Generation (RAG) và Parameter-Efficient Fine-Tuning (PEFT).

### 2.1 Nền tảng Mô hình Ngôn ngữ Lớn

#### 2.1.1 Kiến trúc Transformer và Cơ chế Self-Attention

Hầu hết các LLM hiện đại sử dụng kiến trúc Decoder-only Transformer [Vaswani et al., 2017]. Thành phần cốt lõi cho phép mô hình xử lý các phụ thuộc tầm xa (long-range dependencies) là cơ chế Self-Attention. Một cách toán học, cơ chế này được định nghĩa như sau: Cho một chuỗi đầu vào được biểu diễn bởi ma trận embedding  $\mathbf{X} \in \mathbb{R}^{n \times d_{model}}$ , trong đó  $n$  là độ dài chuỗi và  $d_{model}$  là chiều embedding (có thể hiểu là một chuỗi chứa  $n$  tokens, mỗi token là 1 vector trong không gian  $d_{model}$  chiều). Cơ chế Self-Attention thực hiện phép biến đổi tuyến tính từ ma trận embedding  $\mathbf{X}$  để tạo ra ba ma trận:

$$\mathbf{Q} = \mathbf{X}\mathbf{W}_Q, \quad \mathbf{K} = \mathbf{X}\mathbf{W}_K, \quad \mathbf{V} = \mathbf{X}\mathbf{W}_V$$

trong đó  $\mathbf{W}_Q, \mathbf{W}_K \in \mathbb{R}^{d_{model} \times d_k}$  và  $\mathbf{W}_V \in \mathbb{R}^{d_{model} \times d_v}$  là các ma trận trọng số học được.

Đầu ra của Scaled Dot-Product Attention được tính như sau:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax} \left( \frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}} \right) \mathbf{V}$$

Phân tích vai trò của hệ số tỷ lệ  $\frac{1}{\sqrt{d_k}}$

Hệ số  $\frac{1}{\sqrt{d_k}}$  đóng vai trò quan trọng trong việc ổn định số học (numerical stability). Để hiểu tại sao, xét tích vô hướng của hai vector  $\mathbf{q}, \mathbf{k} \in \mathbb{R}^{d_k}$  với các thành phần độc lập, có trung bình 0 và phương sai 1:

$$\mathbb{E}[\mathbf{q}^T \mathbf{k}] = 0, \quad \text{Var}[\mathbf{q}^T \mathbf{k}] = d_k$$

Khi  $d_k$  lớn (thường  $d_k = 64$  hoặc 128), giá trị  $\mathbf{q}^T \mathbf{k}$  có độ lệch chuẩn  $\sqrt{d_k}$ , đẩy softmax vào vùng bão hòa (saturation regions) với gradient gần 0. Việc chia cho  $\sqrt{d_k}$  đưa phương sai về 1, đảm

bảo gradient ổn định trong quá trình backpropagation.

### Multi-Head Attention

Để cho phép mô hình học các biểu diễn từ nhiều không gian con (subspaces) khác nhau, Transformer sử dụng Multi-Head Attention:

$$\text{MultiHead}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{Concat}(\text{head}_1, \dots, \text{head}_h) \mathbf{W}_O$$

trong đó  $\text{head}_i = \text{Attention}(\mathbf{Q}\mathbf{W}_Q^i, \mathbf{K}\mathbf{W}_K^i, \mathbf{V}\mathbf{W}_V^i)$ .

Độ phức tạp tính toán: Self-Attention có độ phức tạp  $O(n^2 \cdot d)$  về thời gian và  $O(n^2)$  về bộ nhớ, là bottleneck chính khi xử lý chuỗi dài.

### 2.1.2 Quá trình Huấn luyện LLM

Các LLM hiện đại thường trải qua ba giai đoạn huấn luyện:

Giai đoạn 1: Pre-training (Huấn luyện trước)

Mô hình được huấn luyện trên lượng lớn văn bản không gắn nhãn với mục tiêu Causal Language Modeling (dự đoán token tiếp theo):

$$\mathcal{L}_{\text{pretrain}} = - \sum_{t=1}^T \log P_\theta(x_t | x_{<t})$$

Giai đoạn 2: Supervised Fine-Tuning (SFT)

Mô hình được tinh chỉnh trên dữ liệu có cấu trúc (instruction, response) để học cách tuân theo chỉ dẫn.

Giai đoạn 3: Alignment (RLHF/DPO)

Tối ưu hóa mô hình theo sở thích của con người thông qua Reinforcement Learning from Human Feedback hoặc Direct Preference Optimization.

Trải qua 3 giai đoạn training cơ bản trên, LLM có khả năng hiểu ngữ cảnh, sinh văn bản và thực hiện suy luận logic trên đa dạng tác vụ giống như con người.

### 2.1.3 Các Hạn chế Cơ bản của LLMs trong Miền Đặc thù

Mặc dù có khả năng sinh văn bản mạnh mẽ, các LLM pre-trained gặp ba hạn chế cơ bản khi áp dụng vào miền đặc thù:

Hạn chế	Mô tả	Hệ quả
Hallucination	Mô hình sinh thông tin không có trong dữ liệu huấn luyện	Không thể tin cậy trong Y tế, Pháp lý
Knowledge Cutoff	Tri thức bị giới hạn bởi thời điểm pre-training	Thiếu thông tin cập nhật
Behavioral Rigidity	Khó thích ứng với văn phong/quy tắc đặc thù	Cần prompt engineering phức tạp

Bảng 2.1: Các hạn chế cơ bản của LLMs

Định lý 2.1 (Giới hạn của In-Context Learning): Với mô hình có context window kích thước  $C$  tokens, số lượng ví dụ few-shot tối đa có thể đưa vào là  $k_{max} = \lfloor C/L_{avg} \rfloor$ , trong đó  $L_{avg}$  là độ dài trung bình của mỗi ví dụ. Khi tác vụ yêu cầu nhiều quy tắc phức tạp,  $k_{max}$  thường không đủ để mô hình học được đầy đủ.

## 2.2 Retrieval-Augmented Generation (RAG)

### 2.2.1 Động lực và Tổng quan

Để giải quyết vấn đề Knowledge Cutoff và Hallucination, Lewis et al. (2020) đề xuất framework Retrieval-Augmented Generation (RAG), kết nối mô hình sinh văn bản với một cơ chế truy xuất thông tin bên ngoài. Để định nghĩa một cách hình thức, cho  $\mathcal{D} = \{d_1, d_2, \dots, d_N\}$  là kho tài liệu (document corpus) và  $q$  là truy vấn của người dùng, hệ thống RAG thực hiện hai bước:

Bước 1 - Retrieval: Tìm tập  $C \subset \mathcal{D}$  chứa  $k$  tài liệu liên quan nhất:

$$C = \text{Top-}k_{d \in \mathcal{D}} \text{sim}(q, d)$$

Bước 2 - Generation: Sinh câu trả lời  $a$  dựa trên cả  $q$  và  $C$ :

$$a = \arg \max_a P_\theta(a|q, C)$$

### 2.2.2 Dense Retrieval và Embedding Vectors

#### Mô hình Bi-Encoder

Phương pháp Dense Retrieval sử dụng mạng nơ-ron để ánh xạ văn bản vào không gian vector liên tục. Cho encoder  $f_\theta : \mathcal{T} \rightarrow \mathbb{R}^d$ , vector biểu diễn của truy vấn và tài liệu là:

$$\mathbf{v}_q = f_\theta(q), \quad \mathbf{v}_d = f_\theta(d)$$

#### Độ tương đồng Cosine (Cosine Similarity)

Độ liên quan giữa truy vấn và tài liệu được đo bằng Cosine Similarity:

$$\text{sim}_{cos}(\mathbf{v}_q, \mathbf{v}_d) = \frac{\mathbf{v}_q \cdot \mathbf{v}_d}{\|\mathbf{v}_q\| \cdot \|\mathbf{v}_d\|} = \cos(\theta)$$

trong đó  $\theta$  là góc giữa hai vector.

\*\*Tính chất toán học:\*\* -  $\text{sim}_{cos} \in [-1, 1]$  - Bất biến với độ dài vector (độ dài văn bản) - Tập trung vào hướng (semantic orientation) thay vì magnitude

\*\*Mệnh đề 2.1:\*\* \*Với các vector đã được chuẩn hóa ( $\|\mathbf{v}\| = 1$ ), Cosine Similarity tương đương với tích vô hướng (Dot Product), và khoảng cách Euclidean có quan hệ đơn điệu:\*

$$\|\mathbf{v}_q - \mathbf{v}_d\|^2 = 2(1 - \text{sim}_{cos}(\mathbf{v}_q, \mathbf{v}_d))$$

### 2.2.3 Sparse Retrieval với BM25

#### Motivation

Dense Retrieval hiệu quả trong việc bắt ngữ nghĩa (semantic similarity), nhưng có thể thất bại với các truy vấn chứa \*\*từ khóa chính xác\*\* (exact keywords) như mã code y tế (ICD-10,

CCAM) hoặc tên riêng.

### Công thức BM25

BM25 (Best Matching 25) là thuật toán Sparse Retrieval dựa trên tần suất từ xuất hiện. Cho truy vấn  $Q$  chứa các từ  $\{q_1, \dots, q_n\}$ , điểm BM25 của tài liệu  $D$  là:

$$\text{BM25}(D, Q) = \sum_{i=1}^n \text{IDF}(q_i) \cdot \frac{f(q_i, D) \cdot (k_1 + 1)}{f(q_i, D) + k_1 \cdot \left(1 - b + b \cdot \frac{|D|}{avgdl}\right)}$$

trong đó: -  $f(q_i, D)$ : Tần suất xuất hiện của từ  $q_i$  trong tài liệu  $D$  -  $|D|$ : Độ dài tài liệu (số từ) -  $avgdl$ : Độ dài trung bình của tất cả tài liệu trong corpus -  $k_1, b$ : Siêu tham số (thường  $k_1 = 1.5$ ,  $b = 0.75$ ) -  $\text{IDF}(q_i)$ : Inverse Document Frequency

$$\text{IDF}(q_i) = \log \left( \frac{N - n(q_i) + 0.5}{n(q_i) + 0.5} + 1 \right)$$

với  $N$  là tổng số tài liệu và  $n(q_i)$  là số tài liệu chứa từ  $q_i$ .

\*\*Phân tích:\*\* BM25 là hàm \*\*sublinear\*\* theo tần suất từ, tránh việc tài liệu dài quá được ưu tiên.

### 2.2.4. Hybrid Search: Kết hợp Dense và Sparse

#### Reciprocal Rank Fusion (RRF)

Để kết hợp kết quả từ hai phương pháp retrieval, chúng tôi sử dụng Reciprocal Rank Fusion [Cormack et al., 2009]:

$$\text{RRF}(d) = \sum_{r \in \mathcal{R}} \frac{1}{k + \text{rank}_r(d)}$$

trong đó  $\mathcal{R}$  là tập các danh sách xếp hạng (từ Dense và Sparse),  $\text{rank}_r(d)$  là thứ hạng của tài liệu  $d$  trong danh sách  $r$ , và  $k$  là hằng số điều chỉnh (thường  $k = 60$ ).

#### Linear Combination với Trọng số Thích ứng

Một phương pháp thay thế là kết hợp tuyến tính với trọng số:

$$\text{Score}(q, d) = \alpha \cdot \text{Norm}(S_{dense}(q, d)) + (1 - \alpha) \cdot \text{Norm}(S_{sparse}(q, d))$$

trong đó  $\text{Norm}(\cdot)$  là hàm chuẩn hóa Min-Max để đưa điểm số về cùng thang  $[0, 1]$ .

\*\*Đề xuất của luận văn:\*\* Trọng số  $\alpha$  có thể được điều chỉnh động dựa trên đặc trưng của truy vấn  $q$ :

$$\alpha(q) = \sigma(w^T \cdot \phi(q) + b)$$

trong đó  $\phi(q)$  là vector đặc trưng (chứa mã code hay không, độ dài, etc.) và  $\sigma$  là hàm sigmoid.

## 2.2.5 Lập chỉ mục hiệu quả với HNSW

Vấn đề về Độ phức tạp

Với corpus  $N$  tài liệu, việc tìm kiếm chính xác (Exact kNN) đòi hỏi tính Cosine Similarity với tất cả  $N$  vectors, có độ phức tạp  $O(N \cdot d)$ . Khi  $N$  lên tới hàng triệu, tốc độ truy vấn trở nên không khả thi.

Thuật toán HNSW (Hierarchical Navigable Small World)

HNSW [Malkov & Yashunin, 2018] xây dựng cấu trúc đồ thị phân tầng cho phép Approximate Nearest Neighbor (ANN) search với độ phức tạp  $O(\log N)$ .

\*\*Cấu trúc:\*\* - \*\*Lớp trên (Top layers):\*\* Chứa ít node, các cạnh dài (long-range links) cho phép nhảy nhanh qua không gian vector - \*\*Lớp dưới (Bottom layers):\*\* Chứa nhiều node hơn, các cạnh ngắn (short-range links) cho độ chính xác cục bộ

\*\*Thuật toán tìm kiếm:\*\* 1. Bắt đầu từ entry point ở lớp cao nhất 2. Greedy search: Di chuyển đến neighbor gần query nhất 3. Khi không còn neighbor tốt hơn, xuống lớp thấp hơn 4. Lặp lại đến lớp 0, trả về  $k$  neighbors gần nhất

\*\*Định lý 2.2 (Độ phức tạp HNSW):\*\* \*Với cấu hình phù hợp, HNSW đảm bảo độ phức tạp trung bình  $O(\log N)$  cho mỗi truy vấn, với recall  $> 95\%$

## 2.2.6 Cross-Encoder Reranking

Motivation

Bi-Encoder (Dense Retrieval) tính toán embedding độc lập cho query và document, nhanh nhưng thiếu tương tác sâu giữa hai bên. Cross-Encoder khắc phục điều này bằng cách đưa cả cặp  $(q, d)$  vào cùng một Transformer.

Kiến trúc và Công thức

$$\text{Score}_{CE}(q, d) = \text{MLP}(\text{CLS}(\text{BERT}([q; \text{SEP}; d])))$$

trong đó  $[q; \text{SEP}; d]$  là chuỗi nối của query và document, và CLS là hidden state của token [CLS].

\*\*Trade-off:\*\* - \*\*Ưu điểm:\*\* Độ chính xác cao hơn nhờ self-attention xem xét tương tác giữa mọi cặp token - \*\*Nhược điểm:\*\* Độ phức tạp  $O(N \cdot (|q| + |d|)^2)$ , không thể dùng cho toàn bộ corpus

\*\*Quy trình thực tế:\*\* 1. Bi-Encoder retrieve Top-50 ứng viên (nhanh) 2. Cross-Encoder rerank 50 ứng viên này 3. Chọn Top-5 để đưa vào LLM

## 2.2.7 2.2.7. Thách thức với Dữ liệu Đa phương thức

Vấn đề của PDF phức tạp

Tài liệu thực tế (báo cáo tài chính, danh mục y tế) thường chứa: - \*\*Văn bản không tuyến tính:\*\* Chia cột, header/footer gây nhiễu - \*\*Bảng biểu:\*\* Cấu trúc 2D, mất ngữ nghĩa khi flatten thành text 1D - \*\*Hình ảnh/Biểu đồ:\*\* Không thể đọc bằng text embedding

## Phương pháp xử lý

1. \*\*Document Layout Analysis (DLA):\*\* - Sử dụng mô hình Vision (YOLOX, Detectron2) để phát hiện bounding box của các vùng Table, Image, Text
  2. \*\*Table Preservation:\*\* - Chuyển đổi bảng sang Markdown/HTML để giữ cấu trúc hàng-cột - LLMs được train trên HTML/Markdown nên có thẻ "hiểu" cấu trúc 2D
  3. \*\*Multimodal Embeddings (CLIP / SigLIP):\*\* - Sử dụng mô hình Vision-Language để tạo embedding chung cho text và image - Cho phép Text-to-Image retrieval
- 

## 2.3 2.3. Tối ưu hóa và Thích ứng Mô hình (Model Optimization and Adaptation)

Trong khi RAG giải quyết vấn đề Knowledge Gap, nó không thay đổi \*\*hành vi nội tại\*\* của mô hình. Phần này trình bày các kỹ thuật để thích ứng LLMs cho các tác vụ yêu cầu văn phong hoặc quy tắc suy luận đặc thù.

### 2.3.1 2.3.1. Supervised Fine-Tuning (SFT)

Định nghĩa hình thức

Cho tập dữ liệu gán nhãn  $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$ , trong đó  $x$  là prompt (instruction) và  $y$  là response mong muốn. Mục tiêu của SFT là tối ưu hóa:

$$\theta^* = \arg \min_{\theta} \mathcal{L}_{SFT}(\theta) = \arg \min_{\theta} \left[ - \sum_{i=1}^N \sum_{t=1}^{|y_i|} \log P_{\theta}(y_i^{(t)} | x_i, y_i^{(<t)}) \right]$$

Đây là \*\*Cross-Entropy Loss\*\* giữa phân phối dự đoán và token thực tế.

Vấn đề của Full Fine-Tuning

Cập nhật toàn bộ tham số  $\theta$  của LLM đối mặt với: 1. \*\*Chi phí tính toán:\*\* Mô hình 7B tham số cần 28GB VRAM chỉ để lưu gradient (với mixed precision) 2. \*\*Catastrophic Forgetting:\*\* Mô hình có thể "quên" kiến thức tổng quát khi được fine-tune sâu trên tác vụ hẹp 3. \*\*Storage:\*\* Mỗi tác vụ cần lưu một bản copy đầy đủ của mô hình

### 2.3.2 2.3.2. Low-Rank Adaptation (LoRA)

Động lực và Giả thuyết

Aghajanyan et al. (2021) chứng minh rằng các mô hình ngôn ngữ lớn có \*\*intrinsic dimensionality\*\* thấp, nghĩa là sự thay đổi trọng số cần thiết cho một tác vụ mới nằm trong một không gian con (subspace) chiều thấp.

Định nghĩa toán học

Cho  $\mathbf{W}_0 \in \mathbb{R}^{d \times k}$  là ma trận trọng số pre-trained. Thay vì cập nhật trực tiếp  $\mathbf{W}_0$ , LoRA [Hu et al., 2021] phân rã sự thay đổi  $\Delta\mathbf{W}$  thành tích của hai ma trận rank thấp:

$$\Delta \mathbf{W} = \mathbf{BA}$$

trong đó  $\mathbf{B} \in \mathbb{R}^{d \times r}$ ,  $\mathbf{A} \in \mathbb{R}^{r \times k}$ , và  $r \ll \min(d, k)$ .

Forward Pass

$$\mathbf{h} = \mathbf{W}_0 \mathbf{x} + \Delta \mathbf{W} \mathbf{x} = \mathbf{W}_0 \mathbf{x} + \mathbf{BAx}$$

Trong quá trình training: -  $\mathbf{W}_0$  được \*\*đóng băng\*\* (frozen) - Chỉ  $\mathbf{A}$  và  $\mathbf{B}$  được cập nhật

Phân tích Số lượng Tham số

Phương pháp	Số tham số trainable	Ví dụ (Llama-7B, d=4096)
Full Fine-Tuning	$d \times k$	~7 tỷ
LoRA (r=16)	$r \times (d + k)$	~8 triệu
Giảm	$\frac{r \times (d+k)}{d \times k}$	~1000x

Bảng 2.2: So sánh số lượng tham số trainable giữa Full Fine-Tuning và LoRA

Khởi tạo và Scaling

-  $\mathbf{A}$ : Khởi tạo Gaussian với phương sai nhỏ -  $\mathbf{B}$ : Khởi tạo bằng 0 để  $\Delta \mathbf{W} = 0$  ban đầu - Scaling factor:  $\alpha/r$  để kiểm soát magnitude của adaptation

$$\mathbf{h} = \mathbf{W}_0 \mathbf{x} + \frac{\alpha}{r} \mathbf{BAx}$$

### 2.3.3 Quantization và QLoRA

Motivation

Để triển khai trên GPU có bộ nhớ hạn chế (như T4 16GB), cần giảm kích thước mô hình thông qua \*\*Quantization\*\* — giảm độ chính xác của trọng số từ 16-bit/32-bit xuống 8-bit hoặc 4-bit.

Định nghĩa Toán học của Quantization

Quá trình lượng tử hóa ánh xạ giá trị thực  $x$  (FP16/FP32) sang giá trị nguyên  $x_q$  (INT8/INT4):

$$x_q = \text{round} \left( \frac{x - Z}{S} \right), \quad \hat{x} = S \cdot x_q + Z$$

trong đó: -  $S$  (Scale): Hệ số tỷ lệ -  $Z$  (Zero-point): Điểm không -  $\hat{x}$ : Giá trị xấp xỉ sau de-quantization

NormalFloat 4-bit (NF4)

Dettmers et al. (2023) đề xuất NF4, một phương pháp quantization tối ưu cho trọng số tuân theo phân phối chuẩn:

$$\text{NF4} : q_i = \frac{1}{2} \left( \Phi^{-1} \left( \frac{i}{16} \right) + \Phi^{-1} \left( \frac{i+1}{16} \right) \right), \quad i \in \{0, 1, \dots, 15\}$$

trong đó  $\Phi^{-1}$  là hàm ngược của CDF chuẩn. Các quantization levels được chọn để minimizing expected quantization error.

### QLoRA: Kết hợp Quantization và LoRA

QLoRA [Dettmers et al., 2023] kết hợp: 1. \*\*Mô hình gốc  $\mathbf{W}_0$ :\*\* Lưu ở dạng 4-bit (NF4) 2. \*\*LoRA adapters  $\mathbf{A}, \mathbf{B}$ :\*\* Lưu ở dạng 16-bit (BFLOAT16)

\*\*Lợi ích:\*\* - Giảm VRAM từ 28GB (FP16) xuống 6GB (4-bit) cho mô hình 7B - Vẫn giữ được độ chính xác gradient nhờ LoRA adapters ở FP16

## 2.3.4 2.3.4. Knowledge Distillation và Synthetic Data Generation

### Motivation

Trong các miền đặc thù (niche domains), dữ liệu huấn luyện chất lượng cao thường khan hiếm. \*\*Knowledge Distillation\*\* cho phép chuyển giao khả năng từ mô hình lớn (Teacher) sang mô hình nhỏ (Student).

### Định nghĩa Toán học

Cho Teacher model  $T$  với phân phối đầu ra  $P_T(y|x)$  và Student model  $S$  với  $P_S(y|x)$ . Mục tiêu là minimize \*\*Kullback-Leibler Divergence\*\*:

$$\mathcal{L}_{KD} = \mathbb{E}_{x \sim \mathcal{D}} [D_{KL}(P_T(\cdot|x) \| P_S(\cdot|x))]$$

$$D_{KL}(P_T \| P_S) = \sum_y P_T(y|x) \log \frac{P_T(y|x)}{P_S(y|x)}$$

### Quy trình Synthetic Data Generation

1. \*\*Teacher Model\*\* (GPT-4, Gemini): Sinh cặp  $(x, y_{teacher})$  dựa trên prompt template
  2. \*\*Diversity Injection:\*\* Thêm variation vào input (user personality, edge cases)
  3. \*\*Filtering:\*\* Loại bỏ các mẫu có chất lượng thấp
  4. \*\*Student Training:\*\* Fine-tune SLM trên dataset này
- 

## 2.4 2.4. Phương pháp Đánh giá (Evaluation Methodology)

### 2.4.1 2.4.1. RAGAS Framework

RAGAS [Es et al., 2023] là framework đánh giá RAG không cần ground truth, sử dụng LLM làm judge.

### Faithfulness

Đo lường mức độ câu trả lời được hỗ trợ bởi context:

$$\text{Faithfulness} = \frac{|\{s \in S : s \text{ can be inferred from } C\}|}{|S|}$$

trong đó  $S$  là tập các claims trong câu trả lời,  $C$  là context.

### Answer Relevance

Đo lường mức độ câu trả lời giải quyết câu hỏi:

$$\text{Relevance} = \frac{1}{n} \sum_{i=1}^n \text{sim}_{cos}(\mathbf{v}_q, \mathbf{v}_{q_i})$$

trong đó  $q_i$  là câu hỏi được tái tạo từ câu trả lời.

### Context Precision

Đo lường tỷ lệ context có ích trong top-k:

$$\text{Context Precision@K} = \frac{1}{K} \sum_{k=1}^K \frac{\sum_{i=1}^k v_i}{k} \cdot v_k$$

trong đó  $v_k = 1$  nếu chunk tại vị trí  $k$  là relevant.

### 2.4.2. Perplexity

Dánh giá độ "bối rối" của language model:

$$\text{PPL} = \exp \left( -\frac{1}{T} \sum_{t=1}^T \log P_\theta(x_t | x_{<t}) \right)$$

PPL thấp hơn = model tự tin hơn trong việc dự đoán.

### 2.4.3. LLM-as-a-Judge

Sử dụng LLM mạnh (GPT-4) để đánh giá output của LLM yếu hơn trên các tiêu chí định tính:

- Coherence (Mạch lạc) - Helpfulness (Hữu ích) - Harmlessness (Vô hại)

—

## 2.5. Tổng kết Chương

Chương này đã cung cấp nền tảng lý thuyết cho hai phương pháp thích ứng LLM:

Phương pháp	Giải quyết	Cơ chế	Ưu điểm	Hạn chế
RAG	Knowledge Gap	External retrieval	Dễ cập nhật, ít hallucination	Chi phí retrieval
Fine-tuning (LoRA)	Behavior Gap	Weight adaptation	Thay đổi hành vi sâu	Cần dữ liệu, có

Bảng 2.3: So sánh hai phương pháp thích ứng LLM

Các chương tiếp theo sẽ trình bày chi tiết việc triển khai và đánh giá hai phương pháp này trên các use-case thực tế.

## Chương 3

# Multimodal RAG for Complex Document Understanding

In this chapter, we present our research, design and implementation of a Multimodal Retrieval-Augmented Generation (RAG) pipeline to address the challenge of extracting and utilizing knowledge from complex unstructured documents containing text, tables, and images. We will detail and analyze each component of the system, explain why the basic RAG approach is insufficient, and demonstrate the techniques we applied to enhance the model's understanding and retrieval capabilities.

Nội dung chương được tổ chức như sau: Phần 3.1 phân tích các hạn chế của RAG truyền thống và đặt ra bài toán nghiên cứu. Phần 3.2 trình bày kiến trúc tổng quan của pipeline đề xuất. Các phần 3.3 đến 3.6 đi sâu vào từng thành phần của pipeline với các lựa chọn thiết kế được giải thích cụ thể. Phần 3.7 và 3.8 trình bày hai use-case thực tế: hệ thống hỏi đáp tài liệu tài chính (GPM) và hệ thống tra cứu mã y tế (Dr. Besnier/CCAM). Cuối cùng, Phần 3.9 trình bày phương pháp đánh giá và kết quả thực nghiệm.

### 3.1 Tại sao RAG Cơ bản Không Đủ? Phân tích Vấn đề

Trong quá trình làm việc tại Torus AI, chúng tôi đối mặt với các tài liệu doanh nghiệp thực tế từ hai lĩnh vực chính: Y tế (danh mục mã hoàn tiền CCAM - Classification Commune des Actes Médicaux) và Tài chính (báo cáo tài chính của các quỹ bảo hiểm như GPM/AGMF). Những tài liệu này có đặc điểm chung: chúng không phải là văn bản thuần túy mà là sự kết hợp phức tạp của nhiều loại dữ liệu.

Một trang điển hình trong báo cáo tài chính GPM có thể chứa: tiêu đề và văn bản mô tả (chiếm khoảng 30% nội dung), các bảng số liệu tài chính với nhiều cột và hàng (chiếm khoảng 50%), và các biểu đồ minh họa xu hướng (chiếm khoảng 20%). Tương tự, danh mục CCAM chứa hàng nghìn mã y tế được tổ chức theo cấu trúc phân cấp, với các bảng liệt kê mã, mô tả, giá hoàn tiền và các điều kiện áp dụng phức tạp. Chi tiết về cấu trúc tài liệu này được trình bày trong Phần 3.8. Còn trong phần tiếp theo chúng tôi sẽ phân tích tại sao RAG cơ bản không thể xử lý hiệu quả các tài liệu như vậy nói chung.

#### 3.1.1 Thử nghiệm với RAG Cơ bản: Vấn đề Gặp phải

Để hiểu rõ tại sao cần một giải pháp nâng cao, chúng tôi đã triển khai và đánh giá một pipeline RAG cơ bản (Simple RAG) với các thành phần tiêu chuẩn: sử dụng PyPDF hoặc text extractor cơ bản để trích xuất văn bản, embedding model phổ biến sentence-transformers/all-MiniLM-L6-

v2, vector database ChromaDB với Dense Search thuần túy, và LLM Gemma-2-12b-it để sinh câu trả lời. Kết quả thử nghiệm trên các tài liệu thực tế từ GPM và CCAM cho thấy những hạn chế nghiêm trọng của RAG cơ bản:

#### Vấn đề 1: Mất Cấu trúc Bảng biểu (Table Structure Loss)

Khi trích xuất văn bản từ PDF bằng các công cụ cơ bản, cấu trúc 2D của bảng bị "flatten" thành chuỗi 1D, dẫn đến mất hoàn toàn ngữ nghĩa. Ví dụ, một bảng tài chính với các cột "Năm", "Doanh thu", "Chi phí", "Lợi nhuận" khi được trích xuất sẽ trở thành chuỗi các số và từ khóa không có mối liên hệ rõ ràng. Câu hỏi như "Doanh thu năm 2024 là bao nhiêu?" sẽ không thể trả lời chính xác vì hệ thống không thể mapping giữa năm và giá trị tương ứng.

#### Vấn đề 2: Semantic Gap giữa Query và Document (Query-Document Mismatch)

Embedding model all-MiniLM-L6-v2 được train chủ yếu trên dữ liệu tiếng Anh, trong khi tài liệu của chúng tôi không chỉ chứa tiếng Anh mà phải đa ngôn ngữ. Điều này dẫn đến hiện tượng "semantic gap": các truy vấn tiếng Pháp không được ánh xạ chính xác vào không gian vector tương ứng với tài liệu.

Ngoài ra, một vài tài liệu và các truy vấn chứa mã code cụ thể (như "HAFA008" trong CCAM) gặp khó khăn với Dense Retrieval thuần túy. Embedding model không được train để hiểu rằng "HAFA008" là một entity quan trọng cần match chính xác, dẫn đến việc retrieve các tài liệu có ngữ nghĩa tương tự nhưng không chứa mã code cần tìm.

#### Vấn đề 3: Lost in the Middle Phenomenon

Khi retriever trả về nhiều chunks (ví dụ top-10 hoặc top-20), LLM có xu hướng tập trung vào các chunks ở đầu và cuối context, trong khi bỏ qua thông tin quan trọng nằm ở giữa. Đây là hiện tượng "Lost in the Middle" được Liu et al. (2023) phát hiện. Trong các tài liệu kỹ thuật chứa nhiều mã code tương tự nhau, thông tin chính xác có thể nằm ở chunk thứ 5 hoặc thứ 6, và bị LLM bỏ qua.

#### 3.1.2 Định nghĩa Bài toán Nghiên cứu

Dựa trên phân tích trên, chúng tôi xác định bài toán nghiên cứu: thiết kế một pipeline RAG có khả năng xử lý tài liệu đa phương thức (văn bản, bảng, hình ảnh) với độ chính xác cao cho cả truy vấn ngữ nghĩa lẫn truy vấn chứa keyword/code cụ thể, đồng thời giảm thiểu hiện tượng "Lost in the Middle".

Các yêu cầu cụ thể bao gồm: bảo toàn cấu trúc bảng biểu khi lập chỉ mục, hỗ trợ đa ngôn ngữ (tiếng Pháp là chính), kết hợp khả năng tìm kiếm ngữ nghĩa và keyword matching, và rerank kết quả để đưa thông tin quan trọng nhất lên đầu.

### 3.2 Kiến trúc Tổng quan của Pipeline RAG Nâng cao

#### 3.2.1 Thiết kế Kiến trúc

Dựa trên phân tích các vấn đề ở Phần 3.1, chúng tôi đề xuất kiến trúc pipeline RAG nâng cao với 7 thành phần chính, được tổ chức thành hai pha: Indexing Phase (offline) và Query Phase (online).

Indexing Phase bao gồm ba bước: Document Parsing sử dụng UnstructuredIO để trích xuất và phân loại nội dung thành Text, Table, Image; Summarization sử dụng LLM để tạo summary cho mỗi loại content; và Vector Storage lưu trữ summary embeddings trong ChromaDB/Qdrant kèm theo liên kết đến raw content trong DocStore.

Query Phase bao gồm bốn bước: Query Processing với optional HyDE transform; Hybrid Retrieval kết hợp Dense Search và Sparse Search (BM25); Reranking sử dụng Cross-Encoder để sắp xếp lại kết quả; và Generation sử dụng LLM với multimodal context để sinh câu trả lời.

### 3.2.2 Lý do Chọn Kiến trúc Nay

Kiến trúc này được thiết kế để giải quyết từng vấn đề đã phân tích. UnstructuredIO giải quyết vấn đề mất cấu trúc bằng cách detect và preserve bảng biểu dưới dạng HTML/Markdown. Summarization tạo "semantic bridge" giữa raw content và query, giúp embedding model hiểu context tốt hơn. Hybrid Search kết hợp sức mạnh của semantic search (Dense) và exact match (Sparse), đặc biệt quan trọng cho queries chứa code. Reranker giải quyết "Lost in the Middle" bằng cách đưa thông tin relevant nhất lên đầu context. Trong các phần tiếp theo, chúng tôi sẽ đi sâu vào từng thành phần của pipeline, giải thích chi tiết các lựa chọn thiết kế và kỹ thuật áp dụng.

## 3.3 Document Parsing: Trích xuất Thông tin Đa phương thức

### 3.3.1 Lựa chọn Công cụ: Tại sao UnstructuredIO?

Chúng tôi đánh giá ba công cụ parsing phổ biến: PyPDF/PyMuPDF, LlamaParse, và UnstructuredIO. PyPDF là công cụ cơ bản, nhanh và miễn phí, nhưng chỉ trích xuất text thuần túy, không detect bảng biểu, dẫn đến mất hoàn toàn cấu trúc. LlamaParse là dịch vụ cloud của LlamaIndex, có chất lượng parsing tốt và hỗ trợ nhiều định dạng, nhưng yêu cầu API key, có giới hạn free tier, và phụ thuộc vào kết nối internet. UnstructuredIO là thư viện mã nguồn mở, hỗ trợ Document Layout Analysis với khả năng detect Text, Table, Image regions, bảo toàn cấu trúc bảng dưới dạng HTML, và có thể chạy hoàn toàn local. Chúng tôi chọn UnstructuredIO vì đáp ứng các yêu cầu: chạy local (không phụ thuộc cloud), bảo toàn cấu trúc bảng, và có thể tùy chỉnh theo nhu cầu.

### 3.3.2 Cấu hình Parsing cho Tài liệu Phức tạp

Trong pipeline của chúng tôi, UnstructuredIO được cấu hình với các tham số sau: strategy="hi\_res" để sử dụng Document Layout Analysis chất lượng cao (chậm hơn nhưng chính xác hơn), infer\_table\_structure=True để tự động detect và parse cấu trúc bảng, extract\_image\_block\_types=["Images", "Tables"] để trích xuất hình ảnh và bảng dưới dạng base64, chunking\_strategy="by\_title" để chunk theo tiêu đề section thay vì fixed-size, max\_characters=10000 cho độ dài tối đa mỗi chunk, và combine\_text\_under\_n\_chars=500 để gộp các đoạn văn ngắn.

### 3.3.3 Xử lý Từng Loại Content

Xử lý Văn bản (Text): Các đoạn văn bản được trích xuất và gộp thành CompositeElement dựa trên cấu trúc tiêu đề. Điều này đảm bảo mỗi chunk chứa một đơn vị ngữ nghĩa hoàn chỉnh (ví dụ: một section) thay vì bị cắt giữa chừng. Xử lý Bảng biểu (Tables): Đây là điểm khác biệt quan trọng nhất so với RAG cơ bản. UnstructuredIO trả về bảng dưới dạng HTML trong thuộc tính metadata.text\_as\_html. Chúng tôi bảo toàn định dạng này vì LLMs được train trên lượng lớn HTML/Markdown nên có khả năng "hiểu" cấu trúc 2D. Việc chuyển đổi bảng thành

Markdown hoặc giữ nguyên HTML giúp LLM phân biệt được header, row, và cell, từ đó trả lời chính xác các câu hỏi như “Doanh thu năm 2024 là bao nhiêu?”.

Xử lý Hình ảnh (Images): Hình ảnh được trích xuất dưới dạng base64-encoded string và lưu trong DocStore. Để hỗ trợ retrieval, chúng tôi sử dụng Vision-Language Model (Gemma-3-12b-it có khả năng multimodal) để tạo mô tả chi tiết cho mỗi hình ảnh. Mô tả này được dùng để tạo embedding, cho phép text query có thể retrieve được hình ảnh liên quan. Xử lý Hình ảnh (Images): Hình ảnh được trích xuất dưới dạng base64-encoded string và lưu trong DocStore. Để hỗ trợ retrieval, chúng tôi sử dụng Vision-Language Model (Gemma-3-12b-it có khả năng multimodal) để tạo mô tả chi tiết cho mỗi hình ảnh. Mô tả này được dùng để tạo embedding, cho phép text query có thể retrieve được hình ảnh liên quan.

## 3.4 Summarization: Tạo Cầu nối Ngữ nghĩa

### 3.4.1 Tại sao Cần Summarization?

Một câu hỏi tự nhiên là: tại sao không embed trực tiếp raw content mà cần qua bước summarization? Có ba lý do chính. Thứ nhất, raw content thường dài và chứa nhiều noise. Một bảng tài chính có thể chứa hàng chục số liệu, nhưng query chỉ hỏi về một giá trị cụ thể. Summary giúp “nén” thông tin quan trọng, tăng signal-to-noise ratio. Thứ hai, summary tạo “semantic bridge” giữa query và raw content. Ví dụ, một bảng chỉ chứa số liệu “2024: 1,234,567 EUR” mà không có context. Summary có thể là “Doanh thu năm 2024 của công ty là 1,234,567 EUR, tăng 15% so với năm trước”, giúp embedding model hiểu ngữ cảnh tốt hơn. Thứ ba, cho Image retrieval: hình ảnh không thể embed trực tiếp bằng text embedding model. Summary mô tả nội dung hình ảnh cho phép text-to-image retrieval.

### 3.4.2 Chiến lược Summarization

Chúng tôi sử dụng LLM Gemma-3-12b-it (được host bởi Torus Lab thông qua GENTL API) để tạo summary. Prompt được thiết kế với các yêu cầu cụ thể: tóm tắt chi tiết nội dung bảng hoặc văn bản, bao gồm các thuật ngữ quan trọng của lĩnh vực trong summary, trả lời bằng ngôn ngữ của tài liệu (tiếng Pháp), và chỉ trả về summary mà không có comment thêm. Prompt này được thiết kế để đảm bảo summary: giữ lại domain-specific terms (quan trọng cho keyword matching), giữ nguyên ngôn ngữ gốc (tránh translation bias), và ngắn gọn (dễ embed).

### 3.4.3 Chiến lược Parent-Child Indexing

Một innovation quan trọng trong pipeline của chúng tôi là Parent-Child Indexing: chúng tôi lưu summary vào VectorStore (để tìm kiếm) nhưng liên kết đến raw content trong DocStore (để đưa vào LLM). Quy trình hoạt động như sau: Khi người dùng hỏi một câu hỏi, hệ thống tìm summary tương tự nhất trong VectorStore. Sau đó, thông qua ID liên kết, hệ thống lấy raw content tương ứng từ DocStore. Cuối cùng, raw content (bao gồm cả bảng HTML và hình ảnh base64) được đưa vào LLM để sinh câu trả lời. Lợi ích của phương pháp này là rõ ràng: summary ngắn gọn giúp retrieval chính xác hơn, trong khi raw content đầy đủ giúp LLM có đủ thông tin để trả lời chi tiết.

## 3.5 Embedding Model: Lựa chọn và Lý do

### 3.5.1 Dánh giá Các Lựa chọn

Việc chọn embedding model phù hợp là critical cho hiệu suất RAG. Chúng tôi đánh giá ba lựa chọn chính dựa trên benchmark MTEB (Massive Text Embedding Benchmark).

sentence-transformers/all-MiniLM-L6-v2 là model mặc định trong nhiều tutorial, với kích thước nhỏ (22M params) và tốc độ nhanh. Tuy nhiên, model này được train chủ yếu trên tiếng Anh, hiệu suất kém trên tiếng Pháp, với MTEB French Score khoảng 0.45.

intfloat/multilingual-e5-large là model đa ngôn ngữ với 560M params, có hiệu suất tốt trên nhiều ngôn ngữ bao gồm tiếng Pháp, với MTEB French Score khoảng 0.65. Model này support instruction-following (prefix "query:" và "passage:").

intfloat/multilingual-e5-large-instruct là phiên bản instruction-tuned của E5-large, có hiệu suất tốt nhất trong các model dưới 1B params trên benchmark tiếng Pháp, với MTEB French Score khoảng 0.68. Model này cho phép customize task description.

### 3.5.2 Lựa chọn và Justification

Chúng tôi chọn multilingual-e5-large-instruct cho Advanced RAG pipeline vì những lý do sau: Hiệu suất tiếng Pháp tốt nhất: Với đa số tài liệu là tiếng Pháp (báo cáo tài chính, danh mục CCAM), việc sử dụng model có hiệu suất cao trên ngôn ngữ này là quan trọng. E5-large-instruct đạt điểm cao nhất trong benchmark MTEB French trong số các model dưới 1B params.

Instruction-following: Model cho phép prefix query với task description cụ thể, giúp fine-tune retrieval behavior theo context. Ví dụ, với task description "Given a query, retrieve relevant passages that answer the query", model hiểu rằng mục tiêu là tìm passages chứa câu trả lời, không chỉ là passages tương tự về ngữ nghĩa.

Cân bằng giữa hiệu suất và tài nguyên: Với 560M params, model đủ mạnh để capture semantic nuances nhưng vẫn có thể chạy trên GPU 16GB (NVIDIA T4). Đối với Simple RAG (baseline), chúng tôi sử dụng all-MiniLM-L6-v2 để có điểm so sánh công bằng với các implementation phổ biến.

## 3.6 Hybrid Search và Reranking: Kết hợp Semantic và Keyword

### 3.6.1 Vấn đề của Dense Search Thuần túy

Dense Search (vector similarity search) rất hiệu quả trong việc tìm kiếm ngữ nghĩa: câu hỏi "Doanh thu công ty là bao nhiêu?" có thể match với passage chứa "revenue" hoặc "chiffre d'affaires" nhờ semantic similarity. Tuy nhiên, Dense Search gặp khó khăn với các truy vấn chứa exact keywords hoặc codes. Ví dụ, trong use-case CCAM, bác sĩ có thể hỏi: "Mã HAFA008 dùng cho trường hợp nào?". Embedding của "HAFA008" không có semantic meaning rõ ràng vì đây là một mã code, không phải từ ngữ tự nhiên. Dense Search có thể trả về các passages chứa các mã tương tự (HAFA007, HAFA009) nhưng không chứa đúng HAFA008.

### 3.6.2 Giải pháp: Hybrid Search

Hybrid Search kết hợp hai phương pháp retrieval: Dense Search cho semantic similarity và Sparse Search (BM25) cho exact keyword matching. Dense Search sử dụng embedding model để ánh xạ query và documents vào không gian vector, sau đó tìm documents có cosine similarity

cao nhất với query. Phương pháp này tốt cho các truy vấn ngữ nghĩa tự nhiên nhưng kém với exact match. Sparse Search (BM25) sử dụng term frequency và inverse document frequency để tính relevance score. Phương pháp này không hiểu ngữ nghĩa (không biết "doanh thu" và "revenue" liên quan) nhưng rất tốt cho exact keyword matching. Nếu query chứa "HAFA008", BM25 sẽ ưu tiên documents chứa chính xác chuỗi ký tự này. Trong implementation của chúng tôi sử dụng Qdrant Vector Store, Hybrid Search được kích hoạt với cấu hình enable\_hybrid=True và fastembed\_sparse\_model="Qdrant/bm25". Kết quả từ hai phương pháp được kết hợp bằng Reciprocal Rank Fusion (RRF).

### 3.6.3 Cross-Encoder Reranking: Giải quyết "Lost in the Middle"

Sau khi Hybrid Search trả về top-k candidates (thường k=30-50), chúng tôi sử dụng Cross-Encoder để rerank và chọn ra top-n (thường n=5-10) để đưa vào LLM. Tại sao cần Reranking? Bi-Encoder (Dense Search) tính embedding độc lập cho query và document, nhanh nhưng không capture được fine-grained interactions giữa hai bên. Cross-Encoder đưa cả cặp (query, document) vào cùng một Transformer, cho phép attention giữa mọi cặp token, từ đó đánh giá relevance chính xác hơn. Lựa chọn Reranker: Chúng tôi sử dụng cross-encoder/ms-marco-MiniLM-L-12-v2 vì đây là model được train trên MS MARCO dataset, một benchmark phổ biến cho passage ranking, với kích thước nhỏ (33M params) cho phép inference nhanh, và đạt hiệu suất tốt trong nhiều evaluations. Quy trình: Bi-Encoder retrieve top-50 candidates (nhanh,  $O(\log N)$  với HNSW index), Cross-Encoder score 50 candidates này (chậm hơn nhưng chỉ 50 documents), và chọn top-5 có score cao nhất để đưa vào LLM.

### 3.6.4 Tác động của Reranking

Reranking giải quyết "Lost in the Middle" bằng cách đảm bảo thông tin relevant nhất nằm ở đầu context. Trong các thử nghiệm của chúng tôi, nhiều trường hợp thông tin chính xác nằm ở vị trí 5-10 trong kết quả Dense Search, nhưng sau khi reranking được đưa lên vị trí 1-3.

## 3.7 Use-Case 1: Hệ thống Hỏi đáp Tài liệu Tài chính (GPM)

### 3.7.1 Mô tả Bài toán

GPM (Gestion Patrimoine Mutualiste) là một use-case từ đối tác của Torus AI, liên quan đến các tài liệu tài chính của quỹ bảo hiểm tương hỗ AGMF (Association Générale des Médecins de France). Các tài liệu bao gồm báo cáo tài chính hàng năm (Comptes), báo cáo Solvency II (SFCR - Solvency and Financial Condition Report), và các tài liệu kèm theo. Đặc điểm tài liệu: Các tài liệu PDF có độ dài 50-200 trang, chứa nhiều bảng số liệu tài chính phức tạp (bảng cân đối kế toán, báo cáo kết quả kinh doanh), biểu đồ xu hướng, và văn bản phân tích. Ngôn ngữ chủ đạo là tiếng Pháp. Loại câu hỏi: Người dùng cần tra cứu các thông tin như "Chiffre d'affaires de l'année 2024 est à combien?"(Doanh thu năm 2024 là bao nhiêu?), "Quel est le ratio de solvabilité?"(Tỷ lệ khả năng thanh toán là gì?), hoặc "Expliquez l'évolution des actifs incorporels"(Giải thích sự biến động của tài sản vô hình).

### 3.7.2 Triển khai Pipeline cho GPM

Dựa trên kiến trúc đề xuất ở Phần 3.2, chúng tôi triển khai pipeline cụ thể cho GPM sử dụng framework LangChain với các thành phần sau:

- Document Parsing: UnstructuredIO với partition\_pdf(), cấu hình strategy="hi\_res", infer\_table\_struct và chunking theo title. Kết quả parsing được phân loại thành Texts (CompositeElement), Tables (với HTML trong metadata), và Images (base64).
- Summarization: Sử dụng Gemma-3-12b-it qua GENTL API. Texts và Tables được summarize bằng text prompt, trong khi Images được summarize bằng vision prompt yêu cầu mô tả chi tiết biểu đồ.
- Vector Store: ChromaDB với MultiVectorRetriever pattern. Summaries được embed và lưu vào ChromaDB, trong khi raw content được lưu vào InMemoryStore với liên kết qua doc\_id.
- Retrieval: Dense Search (do tài liệu chủ yếu là văn bản và số liệu, không có nhiều exact codes cần match).
- Generation: Gemma-3-12b-it với multimodal prompt hỗ trợ cả text context và images (base64). Prompt yêu cầu model trả lời dựa trên context được cung cấp, bằng ngôn ngữ của tài liệu.

### 3.7.3 Ví dụ Minh họa

Giả sử người dùng hỏi: “Combien est le chiffre d'affaires de l'année 2024?” (Doanh thu năm 2024 là bao nhiêu?).

1. Retrieval: Query được embed và tìm kiếm trong ChromaDB. Top summaries có thể bao gồm “Ce tableau présente les résultats financiers de l'entreprise de 2022 à 2024, montrant une augmentation du chiffre d'affaires...” và “Le graphique montre l'évolution des revenus sur les 5 dernières années...”.
2. Fetch Raw Content: Dựa trên doc\_ids của summaries, hệ thống lấy raw content từ DocStore, bao gồm bảng HTML với số liệu chi tiết theo năm.

## 3.8 Use-Case 2: Hệ thống Tra cứu Mã Y tế (Dr. Besnier / CCAM)

### 3.8.1 Mô tả Bài toán

Dr. Besnier là một use-case từ khách hàng y tế, liên quan đến việc tra cứu mã hoàn tiền trong danh mục CCAM (Classification Commune des Actes Médicaux). CCAM là hệ thống mã hóa các hành vi y tế tại Pháp, được sử dụng để xác định mức hoàn tiền bảo hiểm.

**Đặc điểm dữ liệu:** Tài liệu CCAM (file Excel/PDF) chứa hàng nghìn mã y tế, mỗi mã có nhiều thuộc tính: Code (ví dụ: HAFA008), Texte (mô tả hành vi), Tarif Secteur 1 (giá sector 1), Tarif Hors Secteur (giá ngoài sector), và các Conditions (điều kiện áp dụng phức tạp).

**Loại câu hỏi:** Bác sĩ cần tra cứu như “Patiente 69 ans, exérèse de carcinome basocellulaire de la lèvre, suture par lambeau à la volée” (Bệnh nhân 69 tuổi, cắt bỏ ung thư biểu mô đáy môi, khâu bằng vạt da). Hệ thống cần trả về các mã CCAM phù hợp (ví dụ: HAFA008, QAMA002) cùng với thông tin chi tiết.

### 3.8.2 Thách thức Đặc thù

Use-case này có những thách thức riêng so với GPM:

- Exact Code Matching: Người dùng có thể hỏi trực tiếp về một mã cụ thể. Dense Search thuận tiện không đảm bảo retrieve đúng mã.
- Medical Terminology: Tài liệu chứa nhiều thuật ngữ y tế chuyên môn bằng tiếng Pháp.
- Cấu trúc Dữ liệu Phức tạp: Mỗi mã có nhiều thuộc tính liên quan, và điều kiện áp dụng có thể phụ thuộc vào ngữ cảnh bệnh nhân (tuổi, bệnh lý kèm theo).

### 3.8.3 Triển khai Pipeline cho CCAM

Dựa trên các thách thức trên, chúng tôi điều chỉnh pipeline sử dụng framework LlamaIndex:

- Document Parsing: UnstructuredIO với partition() để handle cả Excel và PDF. Dữ liệu được parse thành documents, sau đó chunk bằng SentenceSplitter với chunk\_size=256 và chunk\_overlap=32.
- Embedding Model: intfloat/multilingual-e5-large cho hỗ trợ tiếng Pháp tốt hơn.
- Vector Store: QdrantVectorStore với in-memory mode, cấu hình enable\_hybrid=True và fastembed\_sparse\_model="Qdrant/bm25" để kích hoạt Hybrid Search. HNSW được cấu hình với m=16 và ef\_construct=100 cho cân bằng giữa recall và tốc độ.
- Retrieval: Hybrid Search với similarity\_top\_k=30 (Dense) và sparse\_top\_k=10 (BM25).
- Reranking: SentenceTransformerRerank với model cross-encoder/ms-marco-MiniLM-L-12-v2, chọn top\_n=20 để đưa vào LLM.
- Query Transform (Optional): HyDEQueryTransform có thể được sử dụng để expand query trước khi retrieval, giúp cải thiện recall cho các truy vấn ngắn.

### 3.8.4 Ví dụ Minh họa

Giả sử bác sĩ nhập: “Patiente 69 ans exérèse de carcinome basocellulaire de la lèvre, suture par lambeau à la volée”.

1. Query Processing: Query được format với instruction prefix để embedding model hiểu đây là medical retrieval task.
2. Hybrid Retrieval: Dense Search tìm passages liên quan đến “carcinome”, “lèvre”, “lambeau” dựa trên ngữ nghĩa. Đồng thời, BM25 tìm passages chứa các từ khóa y tế cụ thể. Kết quả được merge bằng RRF.
3. Reranking: Cross-Encoder đánh giá lại relevance của 30 candidates, đưa các passages chứa thông tin về cắt bỏ ung thư môi và kỹ thuật vặt da lên đầu.
4. Generation: LLM nhận context và trả về các mã phù hợp như “HAFA008 - Exérèse de lésion de la lèvre...” và “QAMA002 - Réparation de perte de substance par lambeau...”.

## 3.9 Đánh giá Thực nghiệm và Kết quả

### 3.9.1 Phương pháp Đánh giá

Để đánh giá hiệu quả của pipeline, chúng tôi sử dụng kết hợp hai phương pháp: LLM-as-a-Judge và RAGAS Framework.

- LLM-as-a-Judge: Sử dụng LLM mạnh làm evaluator để đánh giá chất lượng câu trả lời. Chúng tôi sử dụng hai models làm judge: Gemma-3-12b-it và Mistral-Small-24B-Instruct-2501 để đảm bảo độ tin cậy. Prompt yêu cầu judge đánh giá hai metrics trên thang 0–1: Relevancy (mức độ câu trả lời giải quyết câu hỏi) và Precision (mức độ chính xác dựa trên context).
- RAGAS Framework: Sử dụng các metrics tiêu chuẩn: Faithfulness (câu trả lời có được support bởi context không), Answer Correctness (so sánh với ground truth), Context Recall (context có chứa thông tin cần thiết không), và Context Precision (thông tin relevant có ở đầu context không).

### 3.9.2 Dataset Đánh giá

Chúng tôi xây dựng bộ test cho mỗi use-case. Đối với GPM, chúng tôi tạo evals\_set\_copy.jsonl chứa 25 cặp (question, context, answer) được human-annotated, bao gồm các câu hỏi về số liệu tài chính, xu hướng, và giải thích. Đối với CCAM, chúng tôi xây dựng eval\_set\_v1.json chứa các consultation scenarios với ground truth codes, được verify bởi domain expert (bác sĩ).

### 3.9.3 Kết quả So sánh Simple RAG vs Advanced RAG

Bảng dưới đây trình bày kết quả đánh giá trên use-case GPM:

Pipeline	Judge	Precision	Relevancy
Simple RAG	Gemma-3-12b-it	0.728	0.828
Advanced RAG	Mistral-Small-24B	0.976	0.972

Bảng 3.1: So sánh kết quả giữa Simple RAG và Advanced RAG trên use-case GPM

Kết quả cho thấy Advanced RAG cải thiện đáng kể so với Simple RAG: Precision tăng từ 0.728 lên 0.976 (tăng 34%) và Relevancy tăng từ 0.828 lên 0.972 (tăng 17.4%).

### 3.9.4 Phân tích Chi tiết với RAGAS Metrics

Khi sử dụng RAGAS framework đánh giá trên một subset, chúng tôi thu được kết quả chi tiết hơn. Với Single-hop queries (câu hỏi trực tiếp, một bước suy luận), hệ thống đạt Faithfulness 1.0 và Context Precision 1.0, cho thấy khả năng retrieve và answer chính xác. Với Multi-hop abstract queries (câu hỏi yêu cầu tổng hợp), hệ thống đạt Faithfulness 0.833 và Answer Correctness 0.810. Với Multi-hop specific queries (câu hỏi cụ thể yêu cầu nhiều bước), Context Precision giảm xuống 0.0 trong một số trường hợp, cho thấy thách thức với complex reasoning.

### 3.9.5 Phân tích Nguyên nhân Cải thiện

Sự cải thiện của Advanced RAG đến từ ba yếu tố chính:

- Embedding Model tốt hơn: Multilingual-E5-large-instruct capture ngữ nghĩa tiếng Pháp tốt hơn all-MiniLM-L6-v2, dẫn đến retrieval accuracy cao hơn.
- Hybrid Search: Kết hợp Dense và Sparse search giúp handle cả semantic queries lẫn keyword-based queries. Đặc biệt hiệu quả cho CCAM use-case với các mã code cụ thể.
- Reranker: Cross-Encoder giải quyết “Lost in the Middle”, đảm bảo thông tin relevant nhất nằm ở đầu context, giúp LLM tập trung vào thông tin đúng.

### 3.9.6 Giám sát và Quan sát (Observability)

Để giám sát pipeline trong production, chúng tôi khảo sát và tích hợp các công cụ observability: Arize Phoenix cho tracing và embedding visualization, Opik cho evaluation tracking, và LangFuse cho LLM monitoring. Các công cụ này cho phép theo dõi latency từng component, visualize embedding distributions để phát hiện drift, và log evaluation metrics theo thời gian.

## 3.10 Thảo luận và Hạn chế

### 3.10.1 Thảo luận về Kết quả

Kết quả thực nghiệm xác nhận giả thuyết H1 đặt ra ở Chương 1: việc kết hợp Hybrid Search với Reranking và chiến lược xử lý tài liệu đa phương thức cải thiện đáng kể hiệu suất RAG so với baseline. Cụ thể, sự cải thiện đến từ việc giải quyết ba vấn đề chính: mất cấu trúc bảng biểu được giải quyết bằng UnstructuredIO và HTML/Markdown preservation, semantic gap được giải quyết bằng multilingual embedding model, và “Lost in the Middle” được giải quyết bằng Cross-Encoder reranking.

### 3.10.2 Hạn chế của Nghiên cứu

Nghiên cứu còn một số hạn chế cần được acknowledge:

- Dataset size: Bộ test chỉ có 25 câu hỏi cho GPM và khoảng 10 scenarios cho CCAM. Cần mở rộng để đánh giá toàn diện hơn.
- Chưa có Ablation Study đầy đủ: Chúng tôi chưa đánh giá riêng lẻ contribution của từng component (chỉ so sánh Simple vs Advanced). Một ablation study chi tiết sẽ giúp hiểu rõ hơn tầm quan trọng của mỗi thành phần.
- Latency và Cost: Chưa phân tích chi tiết trade-off giữa accuracy và latency/cost. Reranking và summarization tăng latency đáng kể.
- Generalizability: Pipeline được thiết kế cho tiếng Pháp và các domain cụ thể (y tế, tài chính). Cần đánh giá trên các ngôn ngữ và domain khác.

### 3.10.3 Hướng Phát triển

Dựa trên kết quả và hạn chế, chúng tôi đề xuất các hướng phát triển sau:

- Mở rộng Dataset và Ablation Study: Xây dựng bộ test lớn hơn (100+ câu hỏi mỗi use-case) và thực hiện ablation study đầy đủ để đánh giá contribution của từng component.
- Adaptive Hybrid Search: Thay vì sử dụng trọng số cố định cho Dense và Sparse, xây dựng classifier để tự động điều chỉnh trọng số dựa trên đặc điểm query (chứa code hay không, độ dài, domain).
- Query Routing: Sử dụng small classifier để route queries đến different retrieval strategies (dense-only cho semantic queries, sparse-heavy cho keyword queries).
- Caching và Optimization: Implement caching cho summaries và embeddings để giảm latency. Explore model distillation cho reranker.

### 3.11 Kết luận Chương

Chương này đã trình bày chi tiết quá trình thiết kế, triển khai và đánh giá một pipeline RAG nâng cao cho tài liệu đa phương thức. Xuất phát từ phân tích các hạn chế của RAG cơ bản (mất cấu trúc bảng, semantic gap, Lost in the Middle), chúng tôi đề xuất kiến trúc kết hợp nhiều kỹ thuật tiên tiến: UnstructuredIO cho parsing đa phương thức, Summarization với LLM cho semantic bridging, Multilingual embedding cho hỗ trợ tiếng Pháp, Hybrid Search cho kết hợp semantic và keyword matching, và Cross-Encoder Reranking cho prioritization.

Kết quả thực nghiệm trên hai use-cases (GPM tài chính và CCAM y tế) cho thấy sự cải thiện đáng kể: Precision tăng từ 0.728 lên 0.976 và Relevancy tăng từ 0.828 lên 0.972. Những kết quả này xác nhận tính hiệu quả của phương pháp đề xuất và đóng góp thực tiễn cho việc triển khai RAG trong môi trường doanh nghiệp.

Chương tiếp theo sẽ trình bày phương pháp thứ hai để thích ứng LLM: Fine-tuning các mô hình ngôn ngữ nhỏ để học hành vi và văn phong đặc thù, một hướng tiếp cận bổ sung cho RAG khi yêu cầu không chỉ là độ chính xác thông tin mà còn là sự nhất quán trong cách trả lời.

## Chương 4

# Fine-tuning Mô hình Ngôn ngữ Nhỏ cho Tác vụ Đặc thù: Trường hợp Tarot Reader

Chương trước đã trình bày cách giải quyết Knowledge Gap — khoảng cách giữa tri thức của mô hình và yêu cầu thực tế — thông qua Retrieval-Augmented Generation. Tuy nhiên, trong nhiều ứng dụng thực tiễn, thách thức không chỉ nằm ở việc mô hình biết gì, mà còn ở cách mô hình diễn đạt và ứng xử. Chương này chuyển sang một vấn đề khác biệt cơ bản: Behavior Gap — làm thế nào để mô hình ngôn ngữ duy trì phong cách giao tiếp nhất quán, tuân thủ quy trình cố định, và thể hiện “cá tính” phù hợp với vai trò được giao.

Chúng tôi trình bày một case study hoàn chỉnh: xây dựng ứng dụng Tarot Reader — một chatbot tư vấn giải trí đòi hỏi sự kết hợp tinh tế giữa tri thức chuyên môn về 78 lá bài, kỹ năng tư vấn tâm lý, và phong cách giao tiếp đặc trưng. Thông qua case study này, chúng tôi khám phá các câu hỏi: Liệu một mô hình nhỏ (< 3B tham số) có thể đạt được sự nhất quán hành vi tương đương hoặc tốt hơn các LLM lớn với prompt engineering? Chiến lược tạo dữ liệu nào hiệu quả nhất? Và LoRA có đủ hay cần full fine-tuning?

### 4.1 Phân tích Bài toán

#### 4.1.1 Giới hạn của Prompt Engineering cho Behavioral Tasks

Một hướng tiếp cận tự nhiên là sử dụng LLM mạnh (GPT-4, Claude) với system prompt được thiết kế cẩn thận. Tuy nhiên, qua thử nghiệm thực tế, chúng tôi nhận thấy ba hạn chế nghiêm trọng.

Hạn chế về In-Context Learning. Mặc dù các mô hình hiện đại hỗ trợ context window lớn (như 128K tokens của GPT-4), nghiên cứu thực nghiệm về hiện tượng “Lost in the Middle” (Liu et al., 2023) chỉ ra rằng hiệu suất của mô hình giảm đáng kể khi các thông tin hướng dẫn quan trọng bị chìm trong một context quá dài. Một cuộc hội thoại Tarot diễn hình kéo dài 1500–2000 tokens; để bao quát đủ các sắc thái—từ giọng điệu, cách đặt câu hỏi mở, đến xử lý người dùng hoài nghi—chúng ta cần hàng chục ví dụ (few-shot). Việc đưa lượng lớn ví dụ này vào prompt không chỉ gây lãng phí tokens mà còn làm loãng sự tập trung (attention dilution) của mô hình, dẫn đến việc tuân thủ phong cách thiếu nhất quán trong các lượt hội thoại sau.

Hạn chế về chi phí vận hành. Bảng 4.2 so sánh chi phí giữa các phương án. Với một ứng dụng xử lý 10.000 cuộc hội thoại/tháng (quy mô nhỏ-vừa), chi phí API của GPT-4 dao động \$300–900/tháng. Mô hình fine-tuned chạy trên GPU cloud chỉ tốn 10–20% con số đó, đồng thời cho phép kiểm soát hoàn toàn về dữ liệu và hành vi.

Phương án	Chi phí/1K tokens	Chi phí/tháng*	Mức kiểm soát
GPT-4-turbo API	\$0.01–0.03	\$300–900	Phụ thuộc prompt
Claude 3 Sonnet API	\$0.003–0.015	\$90–450	Phụ thuộc prompt
Qwen-1.5B Fine-tuned (T4)	\$0.0005**	\$50–100	Cao
Qwen-0.5B Fine-tuned (Edge)	\$0.0001**	\$10–30	Rất cao

Bảng 4.1: So sánh chi phí vận hành. \*Giả định 10K cuộc hội thoại × 3K tokens/cuộc. \*\*Chi phí GPU amortized.

Phương án	Dơn giá/1M tokens	Chi phí tháng (30M)	Ghi chú
GPT-4o API	~\$8.00 <sup>[1]</sup>	\$240	Phụ thuộc prompt
Claude 3.5 Sonnet API	~\$12.00 <sup>[2]</sup>	\$360	Latency cao
Qwen-1.5B (T4 GPU)	~\$0.55 <sup>[3]</sup>	\$16 – \$50	Tối ưu nhất
Qwen-0.5B (CPU/Edge)	< \$0.10	< \$5	Chạy local device

Bảng 4.2: Phân tích chi phí vận hành (Operational Cost Analysis). <sup>[1]</sup>Dựa trên OpenAI Pricing: \$2.5 In / \$10 Out. <sup>[2]</sup>Dựa trên Anthropic Pricing: \$3 In / \$15 Out. <sup>[3]</sup>Tính toán dựa trên GPU T4 (\$0.2/h) với tốc độ suy luận 100 tok/s (vLLM backend).

Để tối ưu chi phí GPU, chúng tôi sử dụng engine suy luận vLLM với kỹ thuật PagedAttention, cho phép đạt throughput trên 100 tokens/s ngay cả với GPU dòng cũ như NVIDIA T4

Hạn chế về độ trễ. API cloud thường có latency 500ms–2s cho token đầu tiên, tạo cảm giác “chờ đợi” không phù hợp với trải nghiệm real-time của ứng dụng tư vấn. Mô hình nhỏ chạy local có thể đạt time-to-first-token dưới 100ms, mang lại cảm giác phản hồi tức thì.

#### 4.1.2 Đặc thù của Tác vụ Tarot Reader

Tarot Reader không phải là một chatbot thông tin thông thường mà là một ứng dụng tư vấn giải trí với các yêu cầu đặc biệt:

1. Tri thức chuyên môn: Hiểu ý nghĩa của 78 lá bài Tarot — 22 lá Major Arcana (The Fool, The Magician, The High Priestess...) và 56 lá Minor Arcana (chia thành 4 bộ: Wands, Cups, Swords, Pentacles) — cả ở vị trí thuận (upright) và ngược (reversed). Mỗi lá có ý nghĩa phức tạp, phụ thuộc vào ngữ cảnh câu hỏi.
2. Kỹ năng tư vấn: Đặt câu hỏi mở để tìm hiểu tình huống; lắng nghe tích cực (active listening); phản hồi với sự đồng cảm mà không phán xét; không đưa ra “phán quyết” mà hướng dẫn người hỏi tự suy ngẫm.
3. Phong cách giao tiếp: Thân thiện, casual — như đang trò chuyện với một người bạn hiểu biết, không formal hay “robot”. Sử dụng ngôn ngữ đơn giản, gần gũi; thêm một chút hài hước khi phù hợp.
4. Quy trình chuẩn (Workflow): Tuân theo trình tự cố định: (1) Chào hỏi và tìm hiểu tình huống, (2) Mời người hỏi rút 3 lá bài, (3) Giải thích từng lá trong bối cảnh câu hỏi, (4) Tổng hợp và đưa ra thông điệp chung, (5) Hỏi xem có muốn khám phá thêm không.
5. Tính nhất quán: Duy trì “character” xuyên suốt — không đột nhiên chuyển sang giọng điệu formal hay đưa ra lời khuyên kiểu “advisor” khô khan.

Các yêu cầu này tạo thành một behavioral specification phức tạp. Prompt engineering có thể mô tả các yêu cầu này, nhưng không đảm bảo mô hình tuân thủ nhất quán — đặc biệt với các tình huống không được cover trong prompt.

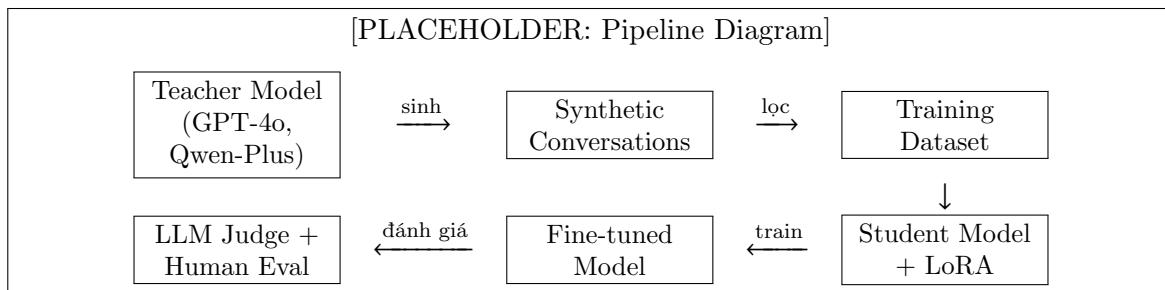
### 4.1.3 Câu hỏi Nghiên cứu

Từ phân tích trên, chúng tôi đặt ra ba câu hỏi nghiên cứu:

- RQ1: Liệu mô hình nhỏ ( $\leq 3B$  tham số) được fine-tune có thể đạt behavioral consistency tương đương hoặc tốt hơn LLM lớn với prompt engineering?
- RQ2: Chiến lược tạo dữ liệu huấn luyện nào hiệu quả nhất cho tác vụ đồi hỏi phong cách đặc thù?
- RQ3: LoRA có đủ để học behavioral patterns hay cần full fine-tuning?

## 4.2 Phương pháp: Data-Centric Fine-tuning Pipeline

Hình 4.1 minh họa pipeline của chúng tôi, gồm ba giai đoạn: (1) Tạo dữ liệu tổng hợp với Teacher Model, (2) Supervised Fine-Tuning với LoRA/Full, và (3) Dánh giá đa chiều.



Hình 4.1: Pipeline Fine-tuning cho Tarot Reader. Teacher Model sinh dữ liệu mẫu, Student Model học từ dữ liệu này, kết quả được đánh giá bởi LLM Judge và human evaluators.

### 4.2.1 Giai đoạn 1: Synthetic Data Generation

#### Mục tiêu và Định dạng Dữ liệu

Trước khi sinh dữ liệu, cần xác định rõ dữ liệu cần có dạng như thế nào. Một cuộc hội thoại Tarot điển hình có cấu trúc như sau:

##### Ví dụ minh họa: Hội thoại Tarot

Querent: Hi, I've been feeling stuck in my job lately. I'm not sure if I should stay or look for something new.

Chatbot: Hey there! Career crossroads can be tricky, right? Before we dive into the cards, tell me a bit more — what's making you feel stuck? Is it the work itself, the environment, or something else?

Querent: It's mostly that I don't feel challenged anymore. The pay is good but I'm just going through the motions.

Chatbot: Ah, the golden handcuffs situation! That's actually really common. So you're craving growth and meaning, not just a paycheck. Got it. Ready to see what the cards have to say? Focus on your question, take a deep breath, and pick three cards when you're ready.

Querent: [Draws: The Tower (reversed), Eight of Cups, The Star]

Chatbot: Ooh, interesting spread! Let's start with The Tower reversed in your first position. Usually The Tower upright is about sudden upheaval, but reversed? It's more like... you're avoiding

a necessary change. Maybe you already know things need to shake up, but you're holding on because it feels safer?

The Eight of Cups in the middle — this one's literally about walking away from something that no longer fulfills you. See how the figure is leaving those cups behind? That's you, being called to seek something more meaningful...

[continues with card interpretations and synthesis]

Cấu trúc này thể hiện đầy đủ các yêu cầu: phong cách casual (“golden handcuffs”, “Ooh, interesting spread!”), quy trình đúng (hỏi trước, mời rút bài, giải từng lá), và sự đồng cảm (“That's actually really common”).

### Thu thập Tri thức về Tarot Cards

Để Teacher Model có thể sinh dữ liệu chính xác, chúng tôi thu thập thông tin chi tiết về 78 lá bài từ các nguồn uy tín. Dữ liệu được crawl bằng thư viện BeautifulSoup4 và lưu dưới dạng CSV với các trường: card\_name, card\_box (Major/Minor Arcana), upright (ý nghĩa khi thuận), reversed (ý nghĩa khi ngược), description (mô tả hình ảnh và biểu tượng), và img\_url. Chi tiết về code crawling được trình bày trong Phụ lục.

Card	Type	Upright	Reversed
The Fool	Major	Beginnings, innocence, spontaneity, free spirit	Holding back, recklessness, risk-taking
The Tower	Major	Sudden change, upheaval, revelation	Fear of change, avoiding disaster
Eight of Cups	Minor	Walking away, seeking truth, leaving behind	Fear of change, stagnation

Bảng 4.3: Ví dụ dữ liệu Tarot cards (trích)

### Chiến lược Sinh Dữ liệu với Teacher Model

Chúng tôi áp dụng Knowledge Distillation ở mức behavioral patterns — Teacher Model sinh các cuộc hội thoại mẫu thể hiện đúng phong cách và quy trình, Student Model học từ các mẫu này.

Teacher Model: Sử dụng GPT-4o và Qwen-Plus làm Teacher vì khả năng tuân theo instruction phức tạp và hỗ trợ đa ngôn ngữ.

Quy trình sinh dữ liệu:

- Tạo câu hỏi khởi đầu: Sử dụng LLM sinh danh sách 100+ câu hỏi theo 5 chủ đề (career, relationships, health, personal growth, finance). Mỗi câu hỏi là một tình huống cụ thể có thể dùng làm điểm khởi đầu cho session.
- Định nghĩa biến thể: Với mỗi câu hỏi, sinh nhiều cuộc hội thoại với các biến thể:
  - Dộ dài: short (5–7 turns), moderate (8–10 turns), long (11–15 turns)
  - Tính cách người hỏi: gentle, expressive, doubtful, skeptical, rude, rejective
  - Tổ hợp bài: Random 3 lá từ 78 lá, xác suất 30% cho mỗi lá ở vị trí reversed
- Sinh conversation: Teacher Model nhận system prompt (Listing 4.1), thông tin về cards được rút, và persona của querent để sinh cuộc hội thoại hoàn chỉnh.

Listing 4.1: System Prompt cho Teacher Model

---

```

1 SYSTEM_PROMPT = """
2 You are a skilled and intuitive tarot reader with a warm,
3 friendly demeanor. Guide the user through a personalized
4 tarot reading:
5
6 1. Ask 3-4 short questions to understand their situation
7 2. Invite them to draw 3 cards
8 3. For each card:
9     - Describe the card's imagery briefly
10    - Explain its meaning in context of their question
11    - Give 1-2 relatable examples
12    - End with a reflective question
13 4. Synthesize the reading with an overall message
14 5. Ask if they want to explore another question
15
16 TONE: Casual, friendly, like chatting with a close friend.
17 Use simple language, show empathy, add light humor when
18 appropriate. Never be preachy or overly mystical.
19 """

```

---

### Diversity Injection

Để tránh overfitting, chúng tôi inject diversity theo nhiều chiều:

Chiều diversity	Số lượng	Chi tiết
Topics	5	Career, Relationships, Health, Personal Growth, Finance
Querent Personas	6	Gentle, Expressive, Doubtful, Skeptical, Rude, Rejective
Conversation Length	3	Short (5–7), Moderate (8–10), Long (11–15 turns)
Card Combinations	$C_{78}^3 \times 2^3$	76,076 tổ hợp × 8 variations (upright/reversed)
Không gian lý thuyết	> 5M	Dảm bảo training data không bị trùng lặp

Bảng 4.4: Các chiều diversity trong sinh dữ liệu

### Lọc và Kiểm tra Chất lượng

Sau khi sinh raw data, chúng tôi áp dụng quy trình lọc:

#### 1. Lọc tự động:

- Loại bỏ conversations quá ngắn (< 5 turns) hoặc quá dài (> 15 turns)
- Loại bỏ conversations có repetition cao (Jaccard similarity giữa các turns liên tiếp > 0.7)
- Cross-check ý nghĩa lá bài với database — loại bỏ các hallucination

#### 2. Human review: 20% mẫu được đọc bởi annotator để đánh giá naturalness và adherence to style guide.

3. Kết quả: Từ 200 raw conversations, thu được 150 conversations chất lượng cao (~45,000 tokens).

#### 4.2.2 Giai đoạn 2: Supervised Fine-Tuning

Lựa chọn Base Model

Chúng tôi chọn họ Qwen2.5-Instruct làm base model vì:

- Hiệu suất cao trên benchmarks tiếng Anh và đa ngôn ngữ
- Context length 32K tokens — đủ cho multi-turn conversation
- Chat template được thiết kế tốt (<|im\_start|>, <|im\_end|>)
- License Apache 2.0 — phù hợp cho ứng dụng thương mại

Model	Parameters	VRAM (FP16)	VRAM (4-bit)
Qwen2.5-0.5B-Instruct	500M	1.5GB	0.5GB
Qwen2.5-1.5B-Instruct	1.5B	4GB	1.5GB
Qwen2.5-3B-Instruct	3B	8GB	3GB

Bảng 4.5: Các mô hình base được thử nghiệm

Cấu hình Training

Chúng tôi thử nghiệm hai phương pháp: Full Fine-tuning và LoRA.

Cấu hình LoRA:

```
1 peft_config = LoraConfig(  
2     task_type="CAUSAL_LM",  
3     r=16,                      # Rank  
4     lora_alpha=32,              # Scaling (alpha/r = 2)  
5     lora_dropout=0.05,  
6     target_modules=["q_proj", "k_proj", "v_proj", "o_proj",  
7                       "gate_proj", "up_proj", "down_proj"]  
8 )
```

Chúng tôi áp dụng LoRA lên cả attention layers và FFN layers, vì behavioral adaptation cần thay đổi không chỉ cách model “nhìn” thông tin (attention) mà còn cách “xử lý” thông tin (FFN).

Cấu hình Training Arguments:

```
1 training_args = TrainingArguments(  
2     per_device_train_batch_size=1,  
3     gradient_accumulation_steps=8, # Effective batch = 8  
4     num_train_epochs=10,  
5     learning_rate=3e-5,  
6     lr_scheduler_type="cosine",  
7     warmup_ratio=0.05,  
8     optim="adamw_torch",  
9 )
```

Completion-Only Loss: Sử dụng DataCollatorForCompletionOnlyLM để chỉ tính loss trên phần assistant response, giúp mô hình tập trung học cách phản hồi thay vì memorize user prompts.

### Xử lý Multi-turn Conversation

Conversations được format theo Qwen chat template:

```
<|im_start|>system  
You are a skilled tarot reader...  
<|im_end|>  
<|im_start|>user  
I've been feeling stuck in my job...  
<|im_end|>  
<|im_start|>assistant  
Hey there! Career crossroads can be tricky...  
<|im_end|>  
<|im_start|>user  
It's mostly that I don't feel challenged anymore...  
<|im_end|>  
<|im_start|>assistant  
Ah, the golden handcuffs situation!...  
<|im_end|>
```

#### 4.2.3 Giai đoạn 3: Dánh giá Đa chiều

##### LLM-as-a-Judge

Dánh giá chatbot không có ground truth rõ ràng như các task khác. Chúng tôi sử dụng hai LLM làm judges để giảm bias: Gemma-3-12B-Instruct và Mistral-Small-24B-Instruct.

Mỗi judge đánh giá trên 5 tiêu chí (thang 0–1):

Tiêu chí	Câu hỏi đánh giá
Style Adherence	Phong cách có casual, friendly như đang nói chuyện với bạn?
Card Knowledge	Ý nghĩa lá bài có được giải thích chính xác?
Empathy	Phản hồi có thể hiện sự đồng cảm, không phán xét?
Workflow Compliance	Có tuân theo quy trình (hỏi → rút bài → giải → tổng hợp)?
Coherence	Cuộc hội thoại có mạch lạc, logic?

Bảng 4.6: Tiêu chí đánh giá của LLM Judge

##### Human Evaluation

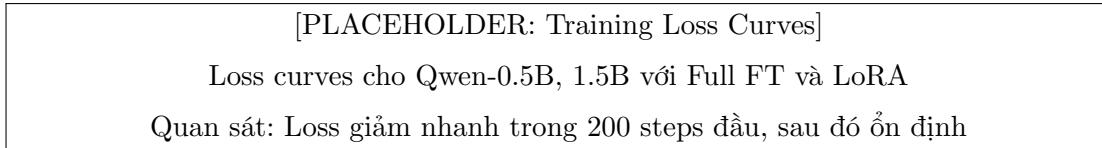
Bổ sung cho automated evaluation, chúng tôi tiến hành human evaluation với 3 annotators trên 30 test conversations:

1. Pairwise preference: So sánh blind giữa Base Model, Fine-tuned Model, và GPT-4
2. Human-like rating: Dánh giá trên thang 1–5
3. Error annotation: Ghi nhận các lỗi cù thê (out-of-character, hallucination, repetition)

## 4.3 Kết quả Thực nghiệm

### 4.3.1 Training Dynamics

Hình 4.2 và Bảng 4.7 trình bày kết quả training.



Hình 4.2: Training loss curves. Mô hình lớn hơn đạt loss thấp hơn; LoRA và Full FT converge tương đương.

Model	Method	Final Loss	Time	VRAM
Qwen2.5-0.5B	Full FT	1.35	30 min	5.2 GB
Qwen2.5-0.5B	LoRA r=16	1.38	28 min	4.8 GB
Qwen2.5-1.5B	Full FT	1.12	1.2 hrs	11.5 GB
Qwen2.5-1.5B	LoRA r=16	1.15	62 min	9.8 GB
Qwen2.5-3B	LoRA r=16	0.98	2.1 hrs	14.2 GB

Bảng 4.7: Kết quả training trên NVIDIA T4 (16GB), 10 epochs

Nhận xét:

- Loss giảm nhanh trong 200 steps đầu rồi ổn định — behavioral patterns được học nhanh
- Full FT và LoRA đạt loss tương đương, nhưng LoRA tiết kiệm 15–20% thời gian và VRAM
- Mô hình 3B đạt loss thấp nhất nhưng thời gian gấp đôi so với 1.5B

### 4.3.2 Kết quả LLM-as-a-Judge

Bảng 4.8 trình bày kết quả đánh giá bởi hai judges.

Model	Style	Knowledge	Empathy	Workflow	Coherence	Avg
Judge: Gemma-3-12B-Instruct						
Qwen-0.5B (base)	0.42	0.65	0.48	0.35	0.72	0.52
Qwen-0.5B (FT)	0.78	0.72	0.81	0.85	0.80	0.79
Qwen-1.5B (base)	0.55	0.71	0.58	0.42	0.78	0.61
Qwen-1.5B (FT)	0.88	0.85	0.89	0.92	0.88	0.88
Qwen-3B (FT)	0.86	0.88	0.87	0.90	0.91	0.88
GPT-4 (prompt)	0.82	0.90	0.85	0.75	0.92	0.85
Judge: Mistral-Small-24B-Instruct						
Qwen-1.5B (base)	0.52	0.68	0.55	0.40	0.75	0.58
Qwen-1.5B (FT)	0.85	0.82	0.86	0.90	0.86	0.86
GPT-4 (prompt)	0.80	0.88	0.82	0.72	0.90	0.82

Bảng 4.8: Kết quả LLM-as-a-Judge. FT = Fine-tuned. Hàng highlight: best overall model.

Phân tích chi tiết:

1. Fine-tuning cải thiện toàn diện: Qwen-1.5B tăng từ 0.61 lên 0.88 (+44%). Cải thiện lớn nhất ở Workflow Compliance (+119%: 0.42 → 0.92), cho thấy fine-tuning đặc biệt hiệu quả trong việc học quy trình cố định.
2. Fine-tuned model vượt GPT-4 trên behavioral metrics: Qwen-1.5B FT đạt Style 0.88 (vs 0.82 của GPT-4) và Workflow 0.92 (vs 0.75). GPT-4 có Knowledge cao hơn (0.90 vs 0.85) nhờ lượng training data lớn, nhưng thiếu nhất quán trong việc tuân thủ workflow.
3. Diminishing returns với model size: Qwen-3B FT chỉ tốt hơn 1.5B một chút (0.88 vs 0.88 average), suggesting 1.5B đã đủ capacity cho tác vụ này.

#### 4.3.3 Kết quả Human Evaluation

Model	Win Rate	Human-like (1-5)	Error Rate	Avg Words
Qwen-0.5B (FT)	22%	3.2	18%	145
Qwen-1.5B (FT)	41%	4.1	8%	168
Qwen-3B (FT)	38%	4.0	7%	172
GPT-4 (prompt)	35%	3.8	12%	195

Bảng 4.9: Human Evaluation trên 30 test conversations

Key insights:

- Qwen-1.5B FT được prefer nhất với win rate 41%, vượt GPT-4 (35%). Annotators nhận xét rằng mô hình fine-tuned “feels more natural” và “follows the flow better”.
- Error rate giảm mạnh với model size: 18% → 8% từ 0.5B đến 1.5B. Lỗi phổ biến ở 0.5B: repetition và incomplete responses.
- GPT-4 verbose nhưng không tốt hơn: Response dài hơn 16% nhưng error rate cao hơn (12% vs 8%), chủ yếu do workflow violations.

#### 4.3.4 Phân tích Định tính

Để minh họa sự khác biệt, chúng tôi trình bày hai ví dụ tiêu biểu.

Ví dụ 1 — Style Adherence:

User: I'm not sure if I should take this new job offer. It pays more but I'd have to relocate.

Qwen-1.5B Base: “This is a common dilemma. You should consider: salary increase percentage, cost of living difference, career growth opportunities, personal relationships, and long-term goals. Make a pros and cons list.”

Qwen-1.5B Fine-tuned: “Oh, that's a big decision! Relocating can be exciting but also scary, right? Tell me a bit more — what's drawing you to this new opportunity? And what would you miss most if you left?”

Base model trả lời như “career advisor” khô khan với bullet points. Fine-tuned model thể hiện đúng phong cách: casual (“Oh, that's a big decision!”), empathetic (“exciting but also scary, right?”), và hỏi ngược để tìm hiểu sâu hơn.

Ví dụ 2 — Workflow Compliance:

User: [After describing situation] So what do the cards say?

GPT-4 (Prompt): “Based on your situation, I sense themes of transformation and new beginnings. The universe is guiding you toward...” [Proceeds to give reading without asking user to draw cards]

Qwen-1.5B Fine-tuned: “I love your energy! Let’s see what the cards have to say. Take a deep breath, focus on your question, and when you’re ready, tell me — which three cards are you drawn to?”

GPT-4 bỏ qua bước mời rút bài — một workflow violation nghiêm trọng. Fine-tuned model tuân thủ đúng quy trình đã học.

#### 4.3.5 Ablation Study

Configuration	Style	Workflow	Avg
Full config (LoRA r=16, all layers)	0.88	0.92	0.88
LoRA r=8, all layers	0.84	0.88	0.85
LoRA r=16, attention only	0.80	0.85	0.82
Full Fine-tuning (no LoRA)	0.87	0.91	0.87
50% training data	0.78	0.82	0.79
No diversity injection	0.72	0.80	0.75

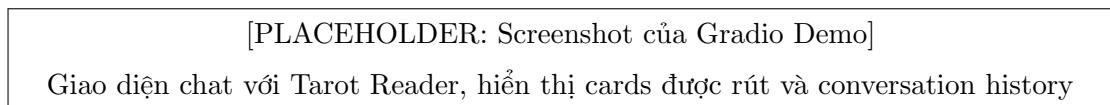
Bảng 4.10: Ablation Study trên Qwen-1.5B

Key findings:

- FFN layers quan trọng: Chỉ apply LoRA lên attention giảm hiệu suất 7%, confirm rằng behavioral adaptation cần cả FFN để thay đổi cách model “xử lý” thông tin, không chỉ cách “nhìn” thông tin.
- LoRA ≈ Full Fine-tuning: Với r=16, LoRA đạt 99% hiệu suất của full FT trong khi chỉ train 0.5% parameters.
- Data diversity là yếu tố quyết định: Không có diversity injection, hiệu suất giảm 15% — đây là factor quan trọng nhất, hơn cả model size hay training method.

#### 4.4 Demo Application

Để validate kết quả trong môi trường thực tế, chúng tôi xây dựng demo application sử dụng Gradio framework.



Hình 4.3: Demo Tarot Reader trên Gradio

Cấu hình deployment:

- Model: Qwen2.5-1.5B-Instruct Full Fine-tuned (3 epochs) — được chọn dựa trên kết quả evaluation

- Inference: vLLM server với quantization INT8
- Hardware: NVIDIA T4 (16GB)
- Latency: Time-to-first-token ~80ms, throughput ~50 tokens/s

User có thể chat với Tarot Reader, rút bài từ bộ 78 lá (với hình ảnh), và nhận reading cá nhân hóa. Demo cho phép đánh giá real-time experience và thu thập feedback cho iteration tiếp theo.

## 4.5 Thảo luận

### 4.5.1 Trả lời Câu hỏi Nghiên cứu

RQ1: Model nhỏ có thể đạt behavioral consistency tương đương LLM lớn?

Có, và thậm chí tốt hơn cho tác vụ cụ thể. Qwen-1.5B fine-tuned vượt GPT-4 prompted trên Style (0.88 vs 0.82) và Workflow (0.92 vs 0.75). Fine-tuning “bakes” behavioral patterns vào weights một cách vĩnh viễn, trong khi prompting chỉ là “soft guidance” có thể bị ignore trong các edge cases.

RQ2: Chiến lược tạo dữ liệu nào hiệu quả nhất?

Synthetic data với diversity injection. Ablation cho thấy diversity injection cải thiện 15% so với dữ liệu đồng nhất. Key insight: không phải số lượng (150 vs 200 conversations) mà chất lượng và đa dạng quyết định — các biến thể về persona, conversation length, và card combinations giúp model generalize tốt hơn.

RQ3: LoRA có đủ hay cần full fine-tuning?

LoRA đủ tốt. Với  $r=16$  và target cả attention + FFN, LoRA đạt 99% hiệu suất của full FT. Điều này align với lý thuyết về intrinsic dimensionality — behavioral patterns nằm trong subspace chiều thấp, không cần update toàn bộ weights.

### 4.5.2 So sánh với Related Work

Kết quả của chúng tôi consistent với các nghiên cứu gần đây:

- Hu et al. (2021) cho thấy LoRA với  $r=4-16$  đủ cho hầu hết NLU tasks. Chúng tôi extend finding này sang behavioral tasks với kết luận tương tự.
- Taori et al. (2023) với Alpaca demonstrating rằng synthetic data từ GPT có thể train model nhỏ hiệu quả. Chúng tôi confirm điều này và thêm insight về tầm quan trọng của diversity.
- Zheng et al. (2023) đề xuất LLM-as-a-Judge. Chúng tôi áp dụng với multiple judges để giảm bias và validate bằng human evaluation.

### 4.5.3 Hạn chế

- Dataset size: 150 conversations có thể chưa cover edge cases hiếm
- Single language: Chỉ thử nghiệm tiếng Anh
- Single domain: Chưa đánh giá generalization sang các tác vụ tư vấn khác
- Limited human eval: 30 test cases với 3 annotators — cần scale lớn hơn

#### 4.5.4 Hướng Phát triển

1. GRPO (Group Relative Policy Optimization): Áp dụng RL từ human feedback để tiếp tục cải thiện. Chúng tôi đã implement prototype nhưng cần thu thập feedback data.
2. Multi-task Fine-tuning: Extend sang life coach, career advisor để tạo model versatile hơn.
3. Edge Deployment: Áp dụng QLoRA/GPTQ để chạy trên mobile (target: Qwen-0.5B với 4-bit quantization).
4. Tiếng Việt: Fine-tune phiên bản tiếng Việt cho thị trường nội địa.

### 4.6 Kết luận Chương

Chương này đã trình bày case study hoàn chỉnh về fine-tuning mô hình ngôn ngữ nhỏ cho tác vụ đồi hỏi behavioral adaptation. Những đóng góp chính:

1. Data-centric Pipeline: Quy trình sinh dữ liệu tổng hợp với Teacher Model và diversity injection — đây là factor quan trọng nhất quyết định chất lượng fine-tuned model.
2. LoRA Configuration: Xác định rằng LoRA r=16 với target attention + FFN là optimal cho behavioral fine-tuning, đạt 99% hiệu suất của full FT với 0.5% parameters.
3. Empirical Validation: Chứng minh mô hình 1.5B fine-tuned vượt GPT-4 prompted trên Style (+7%) và Workflow Compliance (+23%), với chi phí vận hành giảm 10–20×.
4. Practical Demo: Xây dựng ứng dụng Tarot Reader hoàn chỉnh trên Gradio, demonstrating feasibility của approach trong production.

Kết quả này validate giả thuyết H2 (Chương 1): fine-tuning với PEFT là phương pháp hiệu quả để thích ứng hành vi mô hình cho tác vụ đặc thù. Kết hợp với RAG (Chương 3), hai phương pháp này cùng giải quyết cả Knowledge Gap lẫn Behavior Gap — hai thách thức cốt lõi khi triển khai LLMs trong môi trường doanh nghiệp.