**MINISTRY OF EDUCATION AND TRAINING**

**FPT UNIVERSITY**

DeepSeek-based Chatbot system supports work management

by

Dinh Cong Bang

A thesis submitted in conformity with the requirements  
for the degree of Master of Software Engineering

© Copyright by Dinh Cong Bang 2025

**MINISTRY OF EDUCATION AND TRAINING**

**FPT UNIVERSITY**

DeepSeek-based Chatbot system supports work management

by

Dinh Cong Bang

A thesis submitted in conformity with the requirements  
for the degree of Master of Software Engineering

Supervisor:

Assoc. Prof. Phan Duy Hung

© Copyright by Dinh Cong Bang 2025

DeepSeek-based Chatbot system supports work management

Dinh Cong Bang

Degree Master of Software Engineering

FPT University

2025

Abstract

Trong bối cảnh chuyển đổi số hiện nay, việc kết hợp AI vào các hệ thống là một nhu cầu tất yếu. do đó luận văn này phát triển một hệ thống quản lý công việc nhằm tự động hóa việc tạo báo cáo, hướng dẫn người mới, tra cứu đầu mối. Hệ thống nâng cao năng suất thông qua việc tự động tạo báo cáo, đề xuất nhiệm vụ và hướng dẫn nhân viên mới. Mô hình AI được huấn luyện trên dữ liệu từ Jira và dữ liệu các nhóm chat nội bộ, giúp hệ thống nắm bắt ngữ cảnh thực tế và quy trình nghiệp vụ.

Cốt lõi của hệ thống là mô hình DeepSeek, được tinh chỉnh bằng kỹ thuật LoRA (Low-Rank Adaptation) kết hợp Multi-Stage Fine-Tuning. Kỹ thuật LoRA cố định trọng số gốc và chỉ huấn luyện thêm một lượng nhỏ trọng số hạng thấp, giúp giảm đến hàng nghìn lần số tham số cần cập nhật so với tinh chỉnh toàn bộ mô hình [1]. Mô hình sẽ được huấn luyện nhiều vòng, mỗi vòng các ma trận được ép trực giao với nhau tuỳ theo chiến lược và mục tiêu của từng vòng, qua đó tăng khả năng tiếp nhận và xử lý thông tin của mô hình.

Acknowledgments

Tôi xin bày tỏ lòng biết ơn sâu sắc tới PGS. TS. Phan Duy Hùng – người đã hướng dẫn tôi từ những bước đầu trong việc định hình đề tài và định hướng nghiên cứu. Công trình này sẽ không thể hoàn thành nếu thiếu đi sự hỗ trợ tận tình và bền bỉ từ thầy.

Tôi cũng xin gửi lời cảm ơn chân thành tới các đồng nghiệp của mình đã hỗ trợ tôi trong vấn đề cung cấp dữ liệu và hạ tầng training AI.

Cuối cùng, tôi xin dành lời tri ân sâu sắc đến gia đình – những người luôn bên cạnh, yêu thương và ủng hộ tôi vô điều kiện trong suốt chặng đường nghiên cứu này.

# List of Figures

[**Figure 1.** Overall architecture 16](#_Toc201755893)

[**Figure 2.** Data taken from Jira 18](#_Toc201755894)

[**Figure 3.** Data taken from Jira 21](#_Toc201755895)

[**Figure 4.** Data generate from Chat GPT 21](#_Toc201755896)

# List of Table

[**Table 1.** Data in Hugging Face Chat Template format 20](#_Toc201701301)

Table of Contents

[Acknowledgments 2](#_Toc201786896)

[List of Figures 3](#_Toc201786897)

[List of Table 4](#_Toc201786898)

[Table of Contents 5](#_Toc201786899)

[Chapter 1 Introduction 6](#_Toc201786900)

[1.1. Problem And Motivation 6](#_Toc201786901)

[1.2. Related Works 7](#_Toc201786902)

[1.3. Contribution 9](#_Toc201786903)

[Chapter 2 Background Study 10](#_Toc201786904)

[2.1. Business Workflow in Enterprises 10](#_Toc201786905)

[2.2. DeepSeek model 11](#_Toc201786906)

[2.3. Low-Rank Adaptation 12](#_Toc201786907)

[2.4. Orthogonal trong Low-Rank Adaptation 13](#_Toc201786908)

[2.5. Chain of LoRa 14](#_Toc201786909)

[Chapter 3 System Design 16](#_Toc201786910)

[3.1. System Architecture 16](#_Toc201786911)

[3.2. Data Preprocessing 17](#_Toc201786912)

[3.2.1. Data extraction 17](#_Toc201786913)

[3.2.2. Data Formatting for Instruction-Tuning 19](#_Toc201786914)

[3.3. Modeling Module 22](#_Toc201786915)

[Chapter 4 Experiments And Results 23](#_Toc201786916)

[4.1. Data collection 23](#_Toc201786917)

[4.2. Experiments 23](#_Toc201786918)

[4.3. Results 23](#_Toc201786919)

[4.4. Conclusion And Future Work 23](#_Toc201786920)

[References 24](#_Toc201786921)

# Introduction

## Problem And Motivation

Trong môi trường doanh nghiệp hiện tại, việc theo dõi, nhu cầu phân bổ và đánh giá tiến độ công việc hiệu quả ngày càng lớn, đặc biệt với sự gia tăng về quy mô dự án và độ phức tạp trong quy trình vận hành. Mặc dù nhiều công cụ như Jira hay Trello đã được áp dụng rộng rãi, chúng vẫn chủ yếu mang tính chất lưu trữ và theo dõi thủ công, đòi hỏi sự can thiệp thường xuyên từ người dùng. Điều này khiến quá trình tổng hợp báo cáo, đề xuất công việc hoặc hướng dẫn nhân viên mới trở nên tốn thời gian và thiếu nhất quán.

Song song với đó, sự phát triển nhanh chóng của các mô hình ngôn ngữ lớn (LLMs) như ChatGPT, Grok, DeepSeek mở ra cơ hội mới trong việc tự động hóa các tác vụ quản lý công việc. DeepSeek là một dòng mô hình ngôn ngữ mã nguồn mở được huấn luyện trên tập dữ liệu quy mô hàng nghìn tỷ token, với năng lực suy luận được tăng cường thông qua các chiến lược huấn luyện nhiều giai đoạn [4]. Tuy nhiên, để tận dụng hiệu quả các mô hình này trong môi trường doanh nghiệp cụ thể, việc tinh chỉnh (fine-tuning) dựa trên dữ liệu nội bộ là điều thiết yếu. Điều này đặt ra thách thức về hiệu quả tài nguyên, khả năng mở rộng và nguy cơ làm mất đi tri thức nền của mô hình gốc nếu không được huấn luyện đúng cách [2].

Từ nhu cầu thực tiễn, bài viết này này tập trung phát triển một hệ thống trí tuệ nhân tạo hỗ trợ doanh nghiệp trong việc theo dõi, phân tích và tối ưu hóa hoạt động vận hành nội bộ. Hệ thống dựa trên DeepSeek, một trong những mô hình LLM mã nguồn mở nổi bật hiện nay, được huấn luyện trên dữ liệu quy mô lớn, sở hữu khả năng suy luận ngữ cảnh tốt và có hỗ trợ tiếng việt. Tuy nhiên, để khai thác hiệu quả mô hình trong môi trường doanh nghiệp cụ thể, nơi có ngôn ngữ nội bộ, thuật ngữ chuyên ngành và đặc thù quy trình riêng biệt, việc fine-tuning là yếu tố bắt buộc. Fine-tune không chỉ giúp thích nghi với dữ liệu đặc thù của tổ chức mà còn mở ra khả năng tự động hóa hàng loạt quy trình như: tạo báo cáo tiến độ công việc, đề xuất nhiệm vụ phù hợp theo bối cảnh và định hướng nhân viên mới thông qua các hướng dẫn được cá nhân hóa.

Để giải quyết các bài toán trên mà vẫn đảm bảo hiệu quả tài nguyên và độ ổn định mô hình, bài viết đề xuất kỹ thuật LoRA (Low-Rank Adaptation) trong quá trình tinh chỉnh mô hình DeepSeek. LoRA là một phương pháp được đề xuất bởi [1]. cho phép giảm thiểu đáng kể số lượng trọng số cần cập nhật bằng cách chèn thêm các ma trận hạng thấp vào trong cấu trúc mô hình đã huấn luyện sẵn. Nhờ đó, quá trình fine-tune có thể được thực hiện với chi phí bộ nhớ và tính toán thấp hơn hàng chục lần so với tinh chỉnh toàn bộ mô hình, nhưng vẫn giữ nguyên được hiệu năng ở mức cao. Vì các trọng số gốc được giữ nguyên, mô hình sau tinh chỉnh bằng LoRA có khả năng duy trì tốt các tri thức nền, qua đó hạn chế hiệu ứng catastrophic forgetting [2].

LoRA cũng mang lại tính linh hoạt cao cho doanh nghiệp: chỉ cần lưu trữ phần trọng số được chèn thêm (adapter) thay vì toàn bộ mô hình sau khi huấn luyện, giúp giảm chi phí lưu trữ và dễ dàng triển khai đa phiên bản cho các bộ phận khác nhau. Trong bài viết này, dữ liệu huấn luyện được lấy từ các nền tảng Jira, cùng với các tệp log nội bộ ghi nhận tiến trình làm việc, phản hồi nhân viên và lịch sử nhiệm vụ. Dữ liệu này được xử lý và cấu trúc lại theo định dạng chuẩn đầu vào của mô hình, từ đó phục vụ cho việc huấn luyện các adapter LoRA theo từng tác vụ chuyên biệt: tạo báo cáo tổng hợp công việc, đề xuất nhiệm vụ tương thích với vai trò, và hỗ trợ định hướng và hướng dẫn xử lý task cho nhân viên mới.

Việc tích hợp mô hình DeepSeek đã được tinh chỉnh vào hệ thống quản lý công việc sẽ không chỉ giúp tự động hóa nhiều khâu quan trọng, mà còn đóng vai trò như một “trợ lý nội bộ”, có khả năng hiểu ngữ cảnh, đề xuất hành động và hỗ trợ ra quyết định theo thời gian thực. Đây là bước tiến cần thiết nhằm nâng cao năng lực vận hành trong bối cảnh doanh nghiệp đang đối mặt với áp lực chuyển đổi số và tối ưu nguồn lực.

## Related Works

Trong vài năm gần đây, việc ứng dụng các mô hình ngôn ngữ lớn (LLM) vào hệ thống hỗ trợ doanh nghiệp đã nhận được nhiều sự quan tâm từ. Các mô hình như GPT, LLaMA, và gần đây là DeepSeek đã cho thấy tiềm năng trong việc xử lý ngôn ngữ tự nhiên một cách linh hoạt và chính xác, mở ra hướng phát triển cho các hệ thống tự động như trợ lý công việc, công cụ tổng hợp báo cáo, và gợi ý tác vụ thông minh. Tuy nhiên, để triển khai các mô hình này một cách hiệu quả trong môi trường doanh nghiệp cụ thể, yêu cầu về tinh chỉnh mô hình (fine-tuning) theo dữ liệu nội bộ trở nên cần thiết.

Bên cạnh hiệu quả tính toán, một vấn đề khác trong fine-tuning là **hiện tượng catastrophic forgetting**,khi mô hình được tinh chỉnh nhiều lần hoặc cập nhật liên tục theo thời gian. [2] đã nghiên cứu mối quan hệ giữa số lượng bước huấn luyện và mức độ suy giảm kiến thức gốc, chỉ ra rằng ngay cả với các kỹ thuật như LoRA, nếu không có chiến lược quản lý thông tin hợp lý thì mô hình vẫn bị mất kiến thức nền nghiêm trọng. [6] cung cấp một khảo sát toàn diện về các phương pháp continual learning cho LLMs, bao gồm rehearsal, regularization, parameter isolation, và kết hợp các kỹ thuật như adapter hoặc LoRA để duy trì hiệu năng lâu dài.

Một trong những cách giúp tối ưu chi phí và hạn chế **catastrophic forgetting** là nghiên cứu [1], giới thiệu kỹ thuật **LoRA – Low-Rank Adaptation**. Phương pháp này giữ lại toàn bộ trọng số của mô hình gốc và chỉ huấn luyện thêm hai ma trận hạng thấp tại các lớp attention, giúp giảm mạnh chi phí bộ nhớ và tính toán trong quá trình fine-tune. LoRA được chứng minh có thể đạt chất lượng tương đương với fine-tuning toàn phần trong nhiều tác vụ NLP, đồng thời giảm nguy cơ làm mất đi các kiến thức nền đã học trước đó. Nhờ tính chất nhẹ và linh hoạt, LoRA đã trở thành nền tảng cho nhiều phương pháp mở rộng như QLoRA [5], áp dụng LoRA trên mô hình lượng tử hóa 4-bit để tinh chỉnh hiệu quả trên GPU bộ nhớ thấp.

Đối với những model LLM mã nguồn mở, DeepSeek là mô hình có tiềm năng cao, được thiết kế theo hướng hỗ trợ cộng đồng nghiên cứu triển khai mô hình hiệu quả trên dữ liệu tùy biến. Trong báo cáo kỹ thuật mới nhất, nhóm DeepSeek [4] đã huấn luyện các mô hình từ 1.3B đến 67B tham số với dữ liệu chất lượng cao, hỗ trợ đa ngôn ngữ và tối ưu cho khả năng suy luận logic. DeepSeek-R1, phiên bản fine-tuned reasoning model của họ, đạt kết quả tương đương với các mô hình thương mại như OpenAI GPT-3.5 trong nhiều tác vụ đòi hỏi lập luận nhiều bước [7]. Đáng chú ý, DeepSeek được phát hành với giấy phép mã nguồn mở đầy đủ, tạo điều kiện thuận lợi cho việc áp dụng trong môi trường doanh nghiệp không có khả năng truy cập mô hình thương mại đắt đỏ.

Một số tài liệu thực nghiệm cũng đã chứng minh khả năng ứng dụng của DeepSeek vào các bài toán đặc thù. Ví dụ, trong tài liệu hướng dẫn của DataCamp [8] các tác giả đã thực hiện tinh chỉnh DeepSeek-R1 Distill (8B) bằng kỹ thuật LoRA để xây dựng chatbot y khoa có khả năng suy luận theo chuỗi (chain-of-thought). Mặc dù không liên quan trực tiếp đến tác vụ quản lý công việc, nhưng phương pháp tiền xử lý dữ liệu và chiến lược fine-tuning trong nghiên cứu này hoàn toàn có thể áp dụng cho các bài toán tương tự như tạo báo cáo, phân tích tiến độ, hoặc đề xuất nhiệm vụ trong doanh nghiệp.

Các công trình hiện tại đã xây dựng nền móng quan trọng cho việc áp dụng LLM vào các hệ thống trợ lý doanh nghiệp. Tuy nhiên, vẫn còn thiếu những nghiên cứu cụ thể hóa hướng tiếp cận trong bài toán **quản lý công việc**, nơi đòi hỏi khả năng tích hợp dữ liệu nội bộ (như Jira, hệ thống chat), giữ ổn định tri thức nền và hoạt động hiệu quả trên nền tảng tài nguyên hạn chế. Bài viết này kế thừa các hướng tiếp cận đã được xác lập, đồng thời mở rộng bằng cách kết hợp DeepSeek, và các biến thể của kỹ thuật LoRA và dữ liệu công việc thực tế để xây dựng một hệ thống hỗ trợ quản lý phù hợp với nhu cầu của doanh nghiệp hiện đại.

## Contribution

Bài viết tập trung vào việc thiết kế và phát triển một hệ thống quản lý công việc dựa trên mô hình ngôn ngữ lớn DeepSeek, với các thành phần được tinh chỉnh bằng kỹ thuật Low-Rank Adaptation (LoRA) nhằm đảm bảo hiệu quả triển khai trong môi trường doanh nghiệp có tài nguyên hạn chế.

Bài viết đề xuất một kiến trúc hệ thống quản lý công việc AI phù hợp với bối cảnh doanh nghiệp vừa và nhỏ, trong đó dữ liệu được lấy từ những hệ thống quản lý công việc như Trello, Jira và các nguồn tài liệu nội bộ để tổng hợp thành dataset. Từ dữ liệu được cung cấp, mô hình sẽ huấn luyện với kỹ thuật Chain of LoRA, với mỗi vòng huấn luyện các vector trong ma trận A hoặc ma trận B được ép trực giao mềm với nhau, tuỳ theo mục tiêu của vòng huấn luyện đó, mô hình có thể tuỳ chỉnh trực giao. Từ đó tăng khả năng học dữ liệu mới trong những tập dữ liệu hạn chế, khai thác được những dữ liệu khó học, đảm bảo các khả năng chia sẻ kiến thức.

# Background Study

Để thiết kế và triển khai thành công một hệ thống quản lý công việc sử dụng trí tuệ nhân tạo, phù hợp với nhu cầu đặc thù của doanh nghiệp, cần xây dựng nền tảng kiến thức vững chắc về cả quy trình vận hành hiện tại và các tiến bộ kỹ thuật trong lĩnh vực mô hình ngôn ngữ lớn. Trước tiên, cấu trúc và đặc điểm dữ liệu của các phương pháp quản lý công việc phổ biến sẽ được phân tích nhằm xác định các điểm có thể tích hợp mô hình AI. Sau đó, các nền tảng kỹ thuật cốt lõi như mô hình DeepSeek, kỹ thuật tinh chỉnh LoRA và chiến lược huấn luyện nhiều giai đoạn, đây cơ sở cho việc lựa chọn công nghệ phù hợp để xây dựng hệ thống hiệu quả, nhẹ và dễ cập nhật theo môi trường thực tế.

## Business Workflow in Enterprises

Trong môi trường doanh nghiệp hiện đại, việc quản lý công việc hiệu quả đóng vai trò quan trọng trong việc đảm bảo năng suất và sự phối hợp giữa các bộ phận. Các tổ chức thường triển khai các hệ thống quản lý quy trình công việc (Workflow Management Systems - WfMS) để tự động hóa và giám sát các quy trình vận hành phức tạp như Jira, Trello. Nên việc thu thập thông tin trao đổi giữa các bộ phận là một việc khả thi.

Tuy nhiên việc lấy dữ liệu từ các hệ thống quản lý quy trình công việc phát sinh một vấn đề. Các công việc thường có sự liên quan nhất định đến sản phẩm, dịch vụ mà công ty cung cấp, nhưng đồng thời cũng có sự tách biệt về thông tin trao đổi trong nội bộ từng đơn vị, phòng ban. Dẫn đến việc dữ liệu khi thu thập vừa có tính tương đồng, nhưng cũng có tính khác biệt trong chi tiết từng công việc. Ví dụ cùng nói về 1 phần mềm, cùng là những thuật ngữ chuyên ngành, nhưng bộ phận BA sẽ cần thu thập cách tương tác với phần mềm; nhưng bộ phận Dev lại có những yêu cầu chuyên sâu về việc hiểu từng chức năng của hệ thống qua code. Từ đó đặt ra yêu cầu về việc mô hình phải hiểu rõ được chính xác sự liên quan khi các bộ phận cùng đề cập đến 1 chức năng, nhưng cũng phải hiểu rõ yêu cầu chuyên sâu của chức năng đó với từng bộ phận là gì.

## DeepSeek model

The DeepSeek-R1-Distill-Qwen-1.5B là một mô hình ngôn ngữ lớn (LLM) với 1.5 tỷ tham số, được phát triển dựa trên kiến trúc Qwen và tinh chỉnh từ phiên bản DeepSeek-R1. Mô hình này được thiết kế nhằm đạt hiệu suất cao trong các tác vụ suy luận logic, lập trình và toán học, đồng thời tối ưu hóa cho việc triển khai trên các hệ thống có tài nguyên hạn chế.

Mô hình sử dụng kiến trúc decoder-only Transformer với các đặc điểm kỹ thuật sau:

* Số lớp (n\_layers): 30
* Kích thước mô hình (d\_model): 4096
* Kích thước lớp trung gian (d\_intermediate): 11008
* Số đầu attention (n\_heads): 32
* Số đầu attention cho key-value (n\_kv\_heads): 32
* Kích thước từ vựng: 102,400 (byte-level BPE)
* Chiều dài ngữ cảnh tối đa: 4096 token

DeepSeek-R1-Distill-Qwen-7B có thể được triển khai trên các hệ thống với cấu hình phần cứng vừa phải [10]:

* GPU đề xuất: NVIDIA RTX 3070 với VRAM tối thiểu 8GB
* Yêu cầu VRAM: Khoảng 3.3GB
* Hỗ trợ lượng tử hóa: Có thể giảm yêu cầu VRAM xuống còn khoảng 4GB khi sử dụng lượng tử hóa 4-bit

Điều này cho phép các doanh nghiệp triển khai mô hình trên các máy trạm tiêu chuẩn hoặc thậm chí trên các máy tính cá nhân với cấu hình phù hợp. DeepSeek-R1-Distill-Qwen-1.5B đã được đánh giá cao trong các bài kiểm tra về khả năng suy luận và lập trình, đạt kết quả tương đương với các mô hình lớn hơn như Qwen3-235B-thinking. Mô hình này tận dụng kỹ thuật chain-of-thought từ DeepSeek-R1 để cải thiện khả năng suy luận, đồng thời duy trì hiệu suất cao trong khi giảm thiểu yêu cầu về tài nguyên tính toán. Với sự cân bằng giữa hiệu suất và khả năng triển khai, DeepSeek-R1-Distill-Qwen-1.5B là lựa chọn phù hợp cho các ứng dụng doanh nghiệp yêu cầu mô hình ngôn ngữ lớn có khả năng suy luận mạnh mẽ nhưng vẫn có thể triển khai trên hạ tầng phần cứng hiện có.

## Low-Rank Adaptation

LoRA (Low-Rank Adaptation) là một kỹ thuật parameter efficient fine tuning, được đề xuất bởi [1], nhằm giảm thiểu số lượng tham số cần cập nhật khi tinh chỉnh các mô hình ngôn ngữ lớn. Trong phương pháp tinh chỉnh truyền thống, toàn bộ ma trận trọng số W của mô hình sẽ được cập nhật trong quá trình huấn luyện lại, điều này đòi hỏi rất nhiều tài nguyên tính toán và bộ nhớ – đặc biệt khi mô hình có hàng tỷ tham số [1]. Khác với cách tiếp cận này, LoRA giữ nguyên các trọng số gốc và chỉ thêm vào các ma trận có hạng thấp, giúp giảm đáng kể chi phí tính toán và bộ nhớ cần thiết trong quá trình huấn luyện.

Thay vì việc cập nhật trực tiếp ma trận trọng số gốc *W*, LoRA giữ nguyên *W* và chèn thêm một phần hiệu chỉnh có hạng thấp, ký hiệu là ΔW. Với mô hình có một matrận trọng số W∈R*d×k* – nghĩa là có d hàng và k cột, LoRA sẽ thêm vào một ma trận mới được tính từ tích của hai ma trận nhỏ hơn theo công thức

Trong đó:

* A∈Rr×k
* B∈Rd×r
* r≪min(d,k): độ nén của ma trận hiệu chỉnh

Ma trận A đóng vai trò chiếu thông tin đầu vào xuống ma trận hạng thấp, đại diện cho các hướng học mới của trọng số gốc W. Ma trận B: tái mở rộng thông tin từ không gian thấp lên không gian đầu ra. Do đó, toàn bộ thông tin mới được học trong quá trình fine-tuning được mã hóa vào ΔW với chi phí tham số rất thấp.

Cuối cùng, trọng số được sử dụng trong mô hình sau khi áp dụng LoRA là:

Trong đó, α là hệ số điều chỉnh, sử dụng như một scaling factor. Nhằm đảm bảo rằng tổng mức ảnh hưởng của phần hiệu chỉnh ΔW là phù hợp, bất kể giá trị r được chọn là bao nhiêu. Điều này cho phép việc thay đổi r (độ nén của ma trận hiệu chỉnh) mà không làm mất cân bằng mức độ tác động của phần tinh chỉnh lên mô hình gốc [1].

Từ thực tiễn, có một vài cách để chọn α tối ưu cho mô hình; Determined AI khuyến nghị bắt đầu với α=r, sau đó giữ ổn định giá trị này khi thay đổi rank mà không cần tinh chỉnh lại α [11], theo tài liệu AutoTrain Hugging Face đưa ra giá trị mặc định α=32 với rank = 16, và cảnh báo việc đặt α≫r có thể gây overfitting [12]. Sebastian Raschka cũng đưa ra nguyên tắcα / r = 2 [13] là một tỷ lệ hợp lý để giúp ΔW không quá mạnh cũng không quá yếu so với trọng số gốc. Tuy nhiên, theo nghiên cứu của [14] việc sử dụng γr=α/r có thể gây ra hiện tượng suy giảm tốc độ học và hiệu suất tinh chỉnh, đặc biệt khi sử dụng các giá trị r lớn, tác giả đã đề xuất một phương pháp mới gọi là rsLoRA (rank-stabilized LoRA), trong đó hệ số điều chỉnh được thay bằng γr=α/√r để giúp giữ cho độ lớn của phần hiệu chỉnh ΔW ổn định hơn khi r tăng, từ đó cải thiện độ hội tụ và hiệu quả huấn luyện mà không làm ảnh hưởng đến khả năng suy luận của mô hình. Kalajdzievski chứng minh rằng rsLoRA không chỉ cải thiện hiệu suất trên các bài toán benchmark mà còn giảm yêu cầu về tài nguyên tính toán khi huấn luyện các mô hình ngôn ngữ lớn [14].

Việc sử dụng ma trận hạng thấp giúp mô hình áp dụng kỹ thuật LoRA đạt hiệu suất tinh chỉnh tương đương với các phương pháp truyền thống, nhưng chỉ cần 0.1% đến 1% số thamsố được huấn luyện so với fine-tuning đầy đủ [1]. Điều này khiến LoRA trở nên đặc biệt hữu ích trong các môi trường hạn chế tài nguyên, chẳng hạn như khi huấn luyện trên CPU hoặc GPU dung lượng thấp, giúp các doanh nghiệp vừa và nhỏ cũng có thể dễ dàng huấn luyện các mô hình AI của riêng họ.

## Orthogonal trong Low-Rank Adaptation

Orthogonalized trong fine-tuning trong Large Language Models là một kỹ thuật để ép các vector hàng hoặc cột của ma trận vuông góc góc với nhau và có độ dài bằng 1. Điều này sẽ ép mô hình phải học thêm thông tin mới và tránh việc học chung hướng với những dữ liệu đã có sẵn khiến mô hình bị catastrophic forgetting [15] [16].

Các nghiên cứu như Orthogonal Low-Rank Adaptation [15] chỉ ra bằng cách chia sẵn dữ liệu thành các task với chủ đề hoặc mục tiêu riêng biệt, và huấn luyện tuần tự từng task đó theo nguyên tắc:

* Ma trận A phải trực giao mềm với ma trận A của những task trước đó
* Ma trận A trực giao với các hàng khác trong chính nó.

Bằng cách ép các hướng học (ma trận A) của task mới **t**rực giao với các task trước, O-LoRA đảm bảo mô hình không ghi đè kiến thức đã học, từ đó duy trì hiệu suất tốt cho các tác vụ cũ mà không cần truy cập lại dữ liệu gốc. Mỗi task được ánh xạ vào một không gian con khác biệt nhờ tính chất trực giao của các vector học, giúp mô hình phân biệt rõ ràng giữa các nhiệm vụ khác nhau, tránh nhiễu loạn trong quá trình huấn luyện đa nhiệm. Nhờ không gian học bị giới hạn bởi trực giao, mô hình bị "bắt buộc" phải học các hướng mới thay vì khai thác quá mức vào các trọng số cũ, điều này giúp giảm nguy cơ overfit trên các task ít dữ liệu.

Tuy trực giao mang lại hiệu quả cao trong việc bảo toàn tri thức cũ và tránh nhiễu giữa các tác vụ thông qua ràng buộc trực giao, nhưng việc ép ma trận học A của mỗi task phải vuông góc với các task trước cũng có thể vô tình giới hạn không gian biểu diễn mà mô hình có thể học. Điều này đặc biệt đúng trong bối cảnh các task có liên quan cao hoặc chia sẻ nhiều đặc trưng chung – khi đó, việc bắt buộc học theo các hướng hoàn toàn mới có thể khiến mô hình không tận dụng được các biểu diễn hiệu quả sẵn có. [16] [17] chỉ ra các mô hình chỉ được cập nhật trong một không gian biểu diễn bị giới hạn tuy hiệu quả trong việc giảm nhiễu và tránh interference, nhưng lại làm giảm đáng kể khả năng học linh hoạt của mô hìnhdo không thể điều chỉnh đầy đủ các thành phần trọng số.

## Chain of LoRa

Chain of LoRA là một kỹ thuật fine-tuning theo hướng residual learning, được đề xuất bởi [18] để khắc phục hạn chế về khả năng tổng quát của phương pháp LoRA gốc. LoRA mặc dù hiệu quả về mặt tài nguyên, nhưng vẫn kém hơn fine-tuning toàn phần trong một số bài toán do chỉ cập nhật một phần nhỏ trọng số dưới dạng ma trận hạng thấp.

Chain of LoRA lấy cảm hứng từ thuật toán Frank-Wolfe trong tối ưu hóa hàm phi lồi, cho phép tăng dần hạng của phần cập nhật trọng số mà không làm tăng đáng kể chi phí tính toán hay bộ nhớ. Chain of LoRA giải quyết vấn đề giới hạn khả năng học bằng cách xây dựng chuỗi các LoRA module liên tiếp, mỗi module học phần dư còn lại của trọng số chưa được điều chỉnh, từ đó dần dần xấp xỉ tốt hơn với cập nhật tối ưu. Chain of LoRA triển khai mô hình theo nhiều giai đoạn:

* Tune LoRA: Huấn luyện một module LoRA mới trên trọng số hiện tại.
* Tie a knot: Hợp nhất trọng số vừa học vào backbone.
* Extend the chain: Khởi tạo adapter LoRA mới và tiếp tục học phần dư thừa còn lại.

Quá trình này lặp lại cho đến khi hoàn thành chuỗi nhiệm vụ. Chain of LoRA sử dụng residual learning để từng bước nâng cấp mô hình - mỗi adapter mới tương đương một bước fine-tune trong kỹ thuật multi-stage - nhưng không làm gia tăng chi phí tính toán hay bộ nhớ. Chain of LoRA còn đề ra chiến lược giảm dần rank giữa các vòng học, cho phép tận dụng tri thức học được trước đó một cách hiệu quả, đồng thời tối ưu hoá chi phí huấn luyện ở các vòng sau.

Chain of LoRA nâng cao khả năng tổng quát hóa của mô hình so với phương pháp LoRA truyền thống bằng cách học cập nhật trọng số theo từng phần dư (residual), từ đó xấp xỉ dần với trọng số tối ưu. Nhờ cơ chế học theo chuỗi, mỗi vòng lặp Chain of LoRA tập trung vào phần thông tin chưa được học từ các vòng trước, giúp mô hình khai thác triệt để không gian biểu diễn mà không ghi đè các biểu diễn đã học.

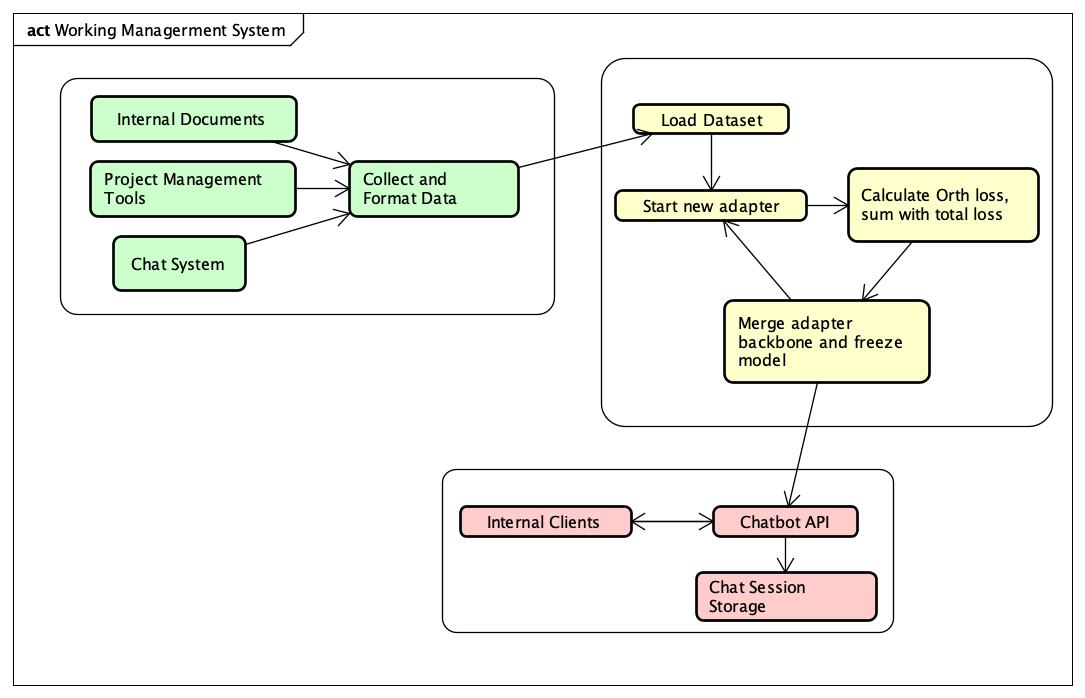
Việc hợp nhất (merge) các adapter vào mô hình backbone sau mỗi vòng giúp giữ nguyên chi phí bộ nhớ, tránh tích lũy các module dư thừa. Đồng thời, chiến lược giảm dần rank trong các vòng sau cho phép mô hình sử dụng ít tham số hơn mà vẫn duy trì chất lượng đầu ra, từ đó tối ưu chi phí tính toán.

Đặc biệt, ở các tác vụ phức tạp hoặc với bộ dữ liệu nhỏ, việc chia nhỏ mục tiêu học thành nhiều bước dễ tối ưu hơn giúp Chain of LoRA tăng độ ổn định trong quá trình huấn luyện. Thực nghiệm trên các mô hình như OPT-1.3B và LLaMA-2-7B cho thấy Chain of LoRA cải thiện độ chính xác trên tất cả các benchmark thử nghiệm (ví dụ: tăng 6.47% trên task WSC) mà không làm tăng chi phí huấn luyện so với LoRA thông thường.

# System Design

## System Architecture

Hệ thống chatbot hỗ trợ hỏi đáp nội bộ được thiết kế theo kiến trúc ba tầng chính: Thu thập dữ liệu, Xử lý & Huấn luyện mô hình, và Triển khai inference thông qua API. Kiến trúc tổng thể của hệ thống được minh họa trong Hình 1.



**Figure 1.** Overall architecture

Ở tầng đầu tiên, dữ liệu được thu thập từ ba nguồn chính: tài liệu hướng dẫn nội bộ, các hệ thống quản lý công việc như Jira hoặc Trello, và thông tin khảo sát/phỏng vấn từ nhân sự trong tổ chức. Sau khi được thu thập, dữ liệu được chuẩn hóa thành các cặp câu hỏi–trả lời theo định dạng instruction-tuning phù hợp cho các mô hình ngôn ngữ lớn.

Tầng huấn luyện mô hìnháp dụng kỹ thuật LoRA kết hợp với Chain of LoRA. Mô hình ngôn ngữ gốc được sử dụng là DeepSeek-R1-Distill-Qwen-1.5B, một phiên bản nhẹ có hiệu suất cao và hỗ trợ tiếng Việt. Quá trình huấn luyện được thực hiện theo từng vòng (round), với mỗi vòng bổ sung một adapter LoRA mới nhằm học phần thông tin dư chưa được tối ưu ở vòng trước. Trong mỗi vòng, hệ thống tính orthogonal loss cộng thêm vào total loss, để đảm bảo các hướng học mới không trùng với các vector khác, giúp hạn chế hiện tượng catastrophic forgetting và gia tăng khả năng học nhiều hướng khác.

Cuối cùng, mô hình sau huấn luyện được lưu trữ và triển khai dưới dạng API thông qua nền tảng Hugging Face Transformers. API này cho phép các hệ thống nội bộ gọi đến mô hình chatbot để nhận phản hồi theo ngữ cảnh, đồng thời lưu trữ lịch sử phiên trò chuyện để tăng khả năng hiểu ngữ cảnh trong các tương tác tiếp theo. Để đảm bảo hiệu suất và kiểm soát độ dài ngữ cảnh, hệ thống chỉ sử dụng một số lượt hội thoại gần nhất (5–10 lượt) để tạo prompt đầu vào cho mỗi phiên inference

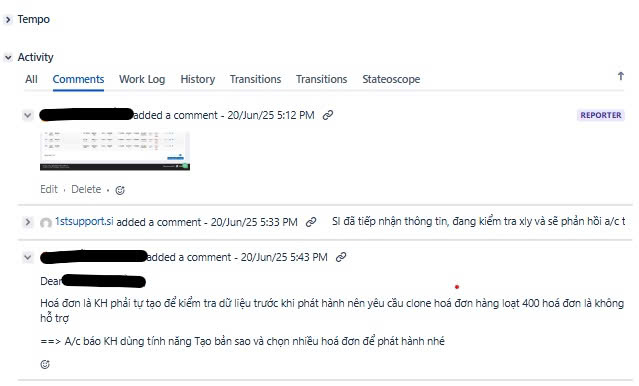
## Data Preprocessing

### Data extraction

Dữ liệu được thu thập dựa trên 3 nguồn chính:

* Dữ liệu lấy từ các hệ thống quản lý công việc như Jira hoặc Trello
* Dữ liệu dựa trên các tài liệu hướng dẫn nội bộ và public
* Dữ liệu từ các nhóm hỗ trợ khách hàng

Với dữ liệu của những hệ thống quản lý công việc, đây là những dữ liệu có độ clean rất cao, chỉ cần các bước clean những dòng dữ liệu trung gian và thay đổi lại cách xưng hô cho phù hợp với mô hình là có thể trở thành dữ liệu train trực tiếp cho AI. Tuy nhiên với những doanh nghiệp có nhiều cấp trung gian, dữ liệu có thể gây nhiễu với comment của các cấp trung gian như trong Figure 2.



**Figure 2.** Data taken from Jira

Comment của các cấp này thường có các từ lặp đi lặp lại, nhiều comment do hệ thống tự tạo hoặc có 1 format chung trong đó các từ dạng: nhờ.\*hỗ trợ", xử lý.\*sớm,… chiếm quá nửa nội dung của comment, những dữ liệu này hoàn toàn có thể lọc qua query hoặc regex.Với những dữ liệu ảnh, sẽ có những kỹ thuật để chuyển ảnh thành các dữ liệu dạng text, bài viết sẽ không đề cập đến cách cụ thể mà chỉ đề xuất hướng xử lý cho các dạng dữ liệu này. Với những dữ liệu dạng form nhập hoặc form thông tin có thể dùng các mô hình Tesseract+Layout hoặc LayoutLM để lấy dữ liệu theo form, với các ảnh có dạng chữ báo lỗi hoặc hiển thị thông tin Tesseract OCR là đủ để lấy dữ liệu. Một hạn chế lớn của dữ liệu lấy từ hệ thống quản lý công việc là nhiều luồng xử lý nằm ngoài Jira, như dữ liệu của các file code, các thao tác trực tiếp trên hệ thống khác, Jira chỉ là nơi truyền và thông báo kết quả, vì vậy lượng dữ liệu thực sự có thể đưa thành dữ liệu train chỉ được 20% trong tổng số dữ liệu.

Ngoài những dữ liệu từ các hệ thống như Jira, Trello. Các công ty khi triển khai một sản phẩm dịch vụ thường có các file excel dạng FAQ với khách hàng hoặc người được hướng dẫn. Đây là những dữ liệu chất lượng nhất, khi các câu hỏi được trả lời thường đã được lọc sẵn những câu hỏi tương tự nhau, và câu trả lời cũng rất chi tiết. Tuy nhiên hạn chế của những dữ liệu này là sẽ có những dữ liệu mang tính thời điểm, như bao giờ triển khai chức năng A hay B, để hạn chế việc này, khi đưa vào làm dữ liệu training, hệ thống có thể tìm và nhận biết những câu hỏi có dạng: khi nào có, bao giờ triển khai,… Để dữ liệu được clean hơn.

Với dữ liệu dạng tài liệu hướng dẫn, các công văn văn bản, đây là tài liệu có tính chuẩn xác cao nhất, nhất là những tài liệu dạng luật, văn bản, nghị định. Tuy nhiên thách thức là các tài liệu này cần khả năng tổng quát cao và chưa phải là một dạng format trao đổi công việc. Phương pháp đơn giản nhất là dùng các mô hình ngôn ngữ lớn như ChatGPT, Claude, Gemini để sinh ra các dữ liệu dạng jsonl. Dựa trên kinh nghiệm thực tế khi yêu cầu ChatGPT phân tích tài liệu và sinh ra bộ câu hỏi trả lời, bài viết đề xuất nên cho mô hình đọc trước file tài liệu để tổng quát được lượng kiến thức. Khi yêu cầu tạo câu hỏi nên có 1 prompt cố định với các yêu cầu cụ thể như format dạng jsonl, mỗi dòng gồm 1 bộ dữ liệu 1 câu hỏi, 1 câu trả lời và đưa ra yêu cầu về 1 mảng cố định trong tài liệu, lượng câu hỏi mỗi session nên là 25-50 để đảm bảo chất lượng các câu hỏi.

Với dữ liệu trên các nhóm chat nội bộ, do vấn đề về mặt bảo mật và quyền riêng tư, bài viết không đề cập đến việc lấy dữ liệu từ đây.

### Data Formatting for Instruction-Tuning

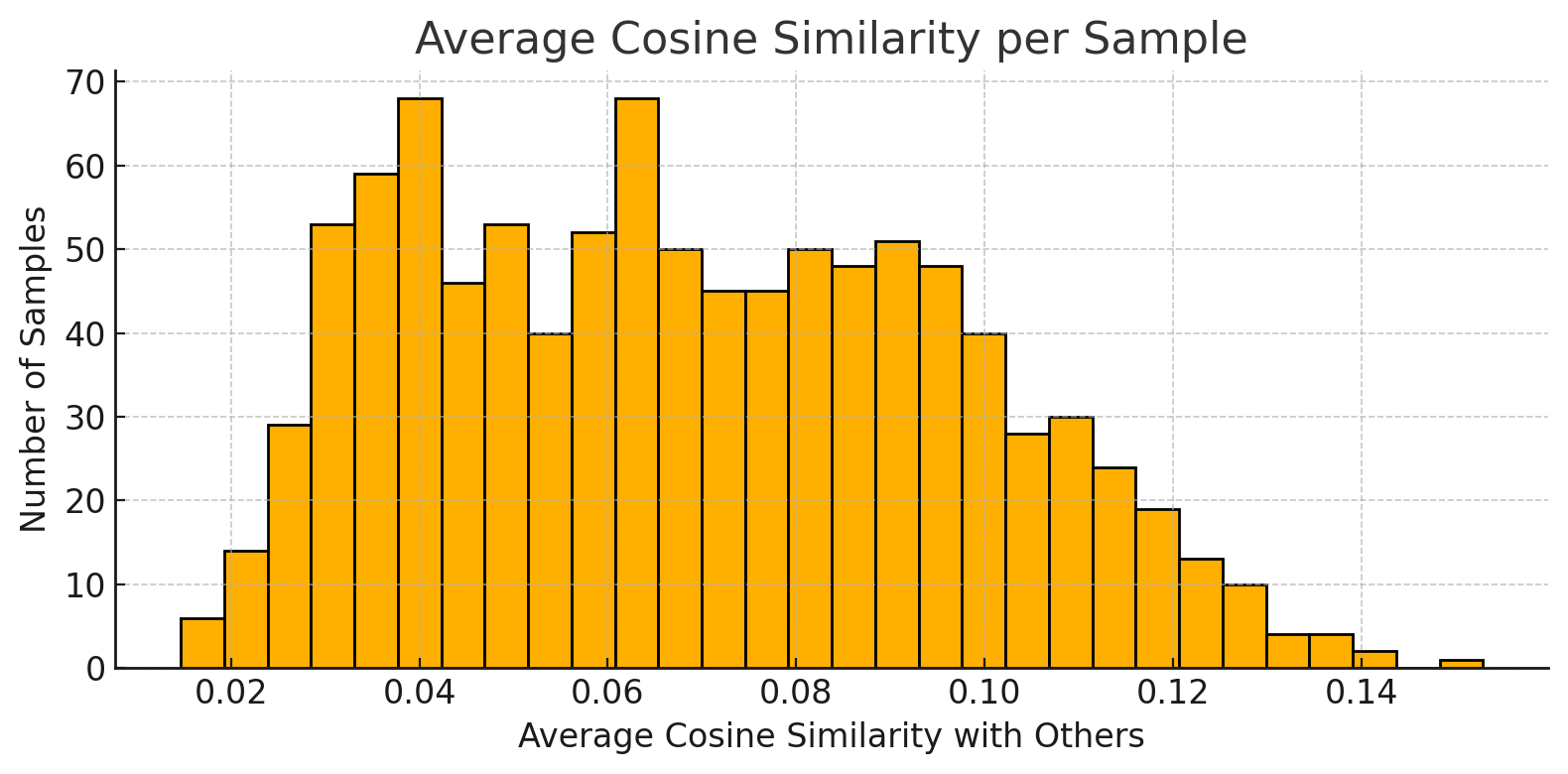
Sau khi đã lấy được dữ liệu theo dạng cặp câu hỏi và câu trả lời, dữ liệu cần được format để huấn luyện. Bài viết đề xuất format dữ liệu theo chuẩn Hugging Face Chat Template [17]. Hugging Face Chat Template Là định dạng dữ liệu hội thoại nhiều lượt (multi-turn) được sử dụng bởi các mô hình dạng "assistant". Dữ liệu được tổ chức theo danh sách các cặp {role, content}, sau đó được biến đổi thành prompt huấn luyện thông qua hàm template được định nghĩa sẵn trong tokenizer. Hàm chat\_template sẽ chịu trách nhiệm ghép các lượt hội thoại thành một chuỗi đầu vào duy nhất. Định dạng này giúp mô hình học được cấu trúc cuộc trò chuyện tự nhiên, giữ ngữ cảnh tốt hơn và hỗ trợ các bài toán như trả lời truy vấn, giải thích dữ liệu, function calling. Đây là định dạng chuẩn hiện được sử dụng bởi các mô hình như LLaMA2-chat, Mistral, DeepSeek, và được hỗ trợ trực tiếp bởi thư viện transformers của Hugging Face khi huấn luyện hoặc sinh dữ liệu đầu vào.

**Table 1.** Data in Hugging Face Chat Template format

|  |
| --- |
| {"messages": [{"role": "user", "content": "How can I reset my password?"}, {"role": "assistant", "content": "To reset your password, click on 'Forgot password' on the login page and follow the instructions."}]}  {"messages": [{"role": "user", "content": "What is the return policy of your store?"}, {"role": "assistant", "content": "Our return policy allows returns within 30 days of purchase, with the original receipt and packaging."}]}  {"messages": [{"role": "user", "content": "Can you explain how invoices work in this system?"}, {"role": "assistant", "content": "Invoices record transactions and are automatically generated when a purchase is made. You can view them under the 'Billing' section."}]}  {"messages": [{"role": "user", "content": "Is there an API to check invoice status?"}, {"role": "assistant", "content": "Yes, you can use the `getInvoiceStatus` API by providing the invoice ID as a parameter."}]}  {"messages": [{"role": "user", "content": "Show me an example of a valid XML invoice payload."}, {"role": "assistant", "content": "<Invoice><ID>12345</ID><Date>2025-06-24</Date><Amount>100.00</Amount></Invoice>"}]} |

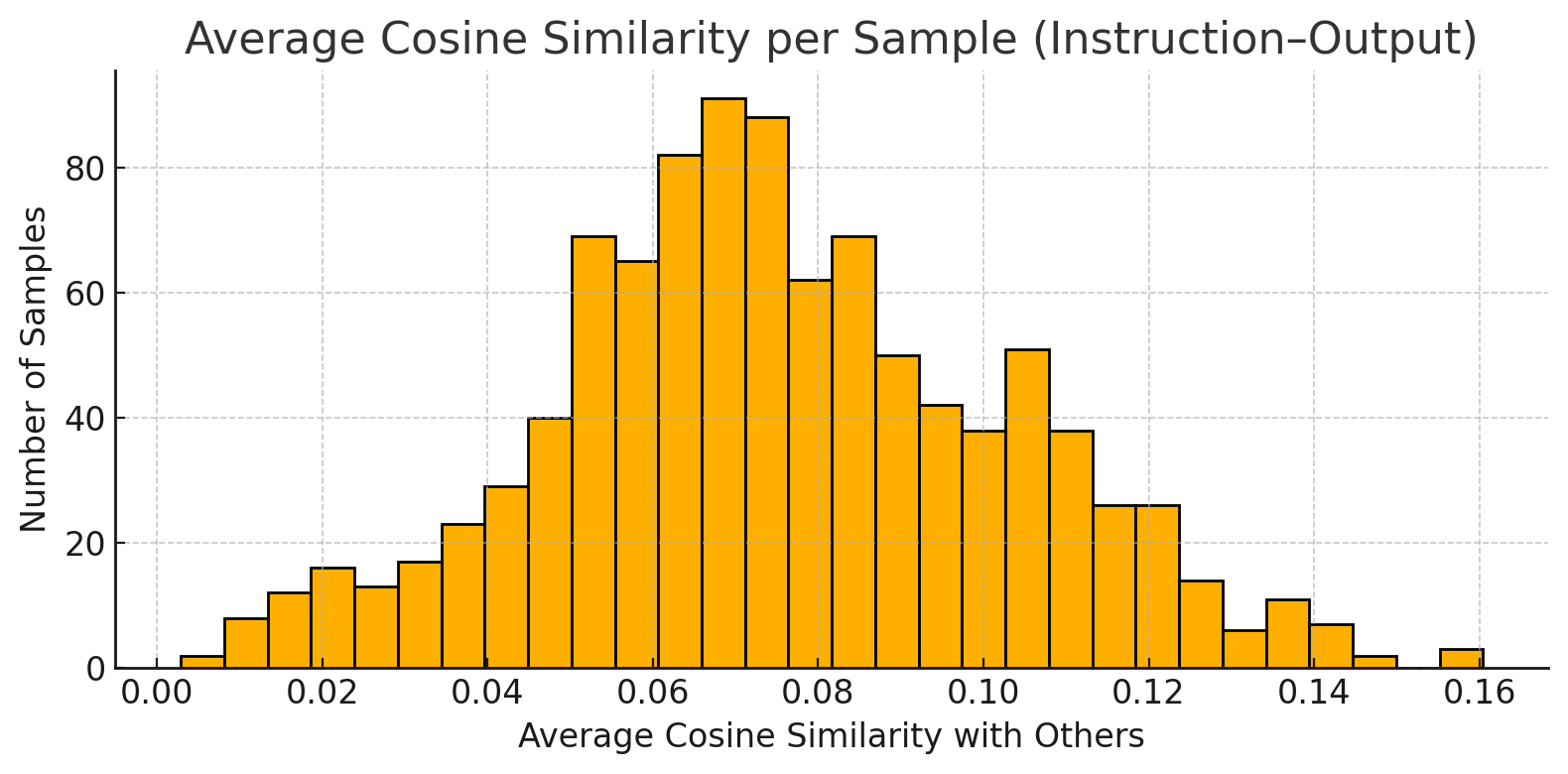
Với dữ liệu lấy từ các hệ thống quản lý công việc, người tạo task để role là "user", vì thường đây là người đưa ra thắc mắc chính và là bên cung cấp thông tin, những người khác trong task sẽ là đặt role là "assistant", những dữ liệu này thường chia ra nhiều lượt nói/trao đổi thông tin. Vì vậy với những phần comment liền nhau của cùng 1 người, bài viết sẽ gộp lại thành 1 comment, để khả năng suy luận của model được liền mạch theo dạng user hỏi 1 câu rồi đến assistant trả lời, tránh cho mô hình bị rối.

Sau khi lấy được 1000 dữ liệu, với 300 dữ liệu lấy từ Jira và 700 dữ liệu được tổng hợp từ các file Excel FAQ khi triển khai phần mềm. Bằng cách áp dụng phương pháp đo Cosine Similarity trung bình giữa mỗi mẫu với toàn bộ các mẫu còn lại. Có thể thấy, dữ liệu phân phối độ tương đồng có dạng lệch trái, tập trung chủ yếu trong khoảng từ 0.04 đến 0.10, cho thấy đa số mẫu hội thoại có nội dung khác biệt nhau đáng kể. Điều này phản ánh mức độ đa dạng cao, giảm thiểu rủi ro mô hình bị học lặp hoặc overfit. Chỉ có một tỷ lệ nhỏ (~5–10%) các mẫu có similarity trung bình vượt ngưỡng 0.12, là ngưỡng mà có khả năng trùng ý tưởng hoặc paraphrase. Có thể thấy tập dữ liệu lấy từ các hệ thống quản lý công việc đạt chất lượng tốt cho mục tiêu fine-tuning mô hình ngôn ngữ hướng hội thoại.



**Figure 3.** Data taken from Jira

Với những dữ liệu Generate từ Chat GPT, do đã quy định mẫu từ prompt, nên không cần format. Tuy nhiên chất lượng của bộ dữ liệu không được bằng dữ liệu lấy từ Jira, Excel khi có >120 cặp gần giống nhau (cosine similarity > 0.90), nội dung có thể bị viết lại lặp lại (paraphrase) nhiều lần. Điều này do ChatGPT không hiểu sâu về bối cảnh nội bộ, quy trình nghiệp vụ, hay ngôn ngữ chuyên ngành như người vận hành thực tế, dẫn đến trả lời chỉ dừng lại ở mức mô phỏng mặt ngôn ngữ. Dù có một tỷ lệ lặp nội dung nhất định, dữ liệu được generate từ ChatGPT vẫn hoàn toàn đủ điều kiện để sử dụng trong huấn luyện mô hình, đặc biệt trong các thiết lập fine-tuning dạng PEFT (LoRA, CoLA).



**Figure 4.** Data generate from Chat GPT

Để đảm bảo được tính đa dạng của mô hình, bài viết dựa theo nghiên cứu của [18] chứng minh rằng việc pha trộn dữ liệu thực và dữ liệu sinh (GPT-generate) theo tỷ lệ vàng ~61.8 : 38.2 sẽ giúp mô hình giữ được độ chính xác từ dữ liệu thật, vừa mở rộng được khả năng biểu đạt, tổng quát hóa nhờ vào sự đa dạng ngôn ngữ của dữ liệu sinh. Trong ngữ cảnh của bài viết, với 1.000 mẫu dữ liệu thật đang có, tỷ lệ này đồng nghĩa với việc cần bổ sung thêm khoảng 618 mẫu dữ liệu generate để đạt tổng ~1.618 mẫu nhằm đảm bảo tính đa dạng mà không giảm chất lượng của mô hình.

## Modeling Module

Model gốc là DeepSeek-R1-Distill-Qwen-1.5B, được tinh chỉnh bằng kỹ thuật Chain of LoRA kết hợp ràng buộc trực giao theo pipeline như sau:

* Tokenization & Data Formatting: Dữ liệu (gồm dữ liệu thật và dữ liệu sinh) được tiền xử lý và chuẩn hóa theo định dạng chat format phù hợp với mô hình Qwen, sau đó được mã hóa thành token thông qua tokenizer của DeepSeek
* Trước mỗi vòng, các ma trận A từ các adapter đã huấn luyện được load lại để tính toán ràng buộc trực giao với adapter hiện tại. Trong mỗi vòng một LoRA module mới được khởi tạo và huấn luyện trên mô hình với các adapter trước đó đã được load lại ở trạng frozen. Sau khi hoàn thành vòng huấn luyện, các tham số của LoRA mới được lưu lại dưới dạng adapter riêng, nhưng chưa được merge vào mô hình chính. Mỗi vòng sau được huấn luyện với rank thấp hơn ở vòng trước, hiện mô hình tuân theo nguyên tắc rank vòng sau bằng ½ rank vòng trước.
* Ở mỗi vòng, các hàng của ma trận A trong cùng một module LoRA được yêu cầu phải trực giao với nhauvà ma trận A hiện tại phải trực giao với các adapter cũ. Để đảm bảo nguyên tắc đó, với mỗi vòng huấn luyện Tức là, ở mỗi vòng huấn luyện, mô hình sẽ tính độ non-orthogonality của các ma trận A(cả trong nội bộ module hiện tại lẫn so với các vòng trước) và cộng giá trị này vào hàm loss để dần ép các vector của ma trận A trực giao.
* Ở vòng cuối cùng, các module LoRA được merge lại để tạo thành một mô hình cuối cùng duy nhất, không còn phụ thuộc vào adapter ngoài

Trong quá trình huấn luyện, việc điều chỉnh lại tham số ở mỗi vòng là yếu tố quyết định khả năng học của mô hình. Ngoài các siêu tham số quen thuộc như rank, learning rate, và số epochs, mô hình còn bổ sung hai tham số đặc biệt nhằm đảm bảo ràng buộc trực giao giữa các vector trong các module LoRA, bao gồm:

* Lambda internal: đại diện cho mức độ kiểm soát tính trực giao giữa các vector hàng trong cùng một ma trận A (nội bộ từng module LoRA). Giá trị này càng cao, ràng buộc trực giao giữa các vector trong cùng module càng chặt, giúp tăng tính đa dạng trong biểu diễn.
* Lambda external: đại diện cho mức độ kiểm soát trực giao giữa ma trận A hiện tại và các ma trận A từ các adapter đã huấn luyện ở các vòng trước. Tham số này giúp đảm bảo rằng biểu diễn mới không trùng lặp với biểu diễn cũ, từ đó tránh lặp lại thông tin và tăng khả năng tổng quát hóa của mô hình.

Đồng thời nhằm tối ưu hiệu suất và chất lượng, model chỉ nên ép trực giao với 1 số module nhất định, bao gồm các module:

* q\_proj: do vector quyết định trực tiếp việc token muốn hỏi gì, nên ép trực giao là điều cần thiết
* k\_proj: tạo ra các vector key để attention tính trọng số, ép vector để tránh các khóa trùng hướng.
* v\_proj: tạo các vector value để được tổng hợp trong attention, vì vậy ép trực giao là cần thiết để giúp mô hình học các giá trị bổ sung khác nhau đảm bảo sự đa dạng thông tin.
* gate\_proj: Điều khiển dòng chảy thông tin thông qua nonlinearity, việc ép trực giao giúp học được các cách mở cổng khác nhau qua các vòng LoRA.

Với các module còn lại, việc trực giao không hề mang đến hiệu quả nhất định, thậm chí với 1 số module như o\_proj là tầng tổng hợp cuối, không tạo ra biểu diễn mới nên ép trực giao có thể sai lệch thông tin đã học từ q/k/v.

Với các vòng đầu tiên, mô hình cần học được biểu diễn nền tảng cho nhiệm vụ, do đó các tham số cần được cấu hình để đảm bảo khả năng học mạnh mẽ với phạm vi biểu diễn rộng. Cụ thể, chiến lược cấu hình tham số cho vòng đầu tiên tuân theo các nguyên tắc sau:

* Rank được đặt ở mức cao để mô hình có đủ năng lực biểu diễn các đặc trưng quan trọng trong dữ liệu.
* Learning rate, do chưa có cập nhật nào trước đó, nên được thiết lập ở mức tương đối cao để giúp mô hình nhanh chóng điều chỉnh các tham số trong module LoRA mới khởi tạo.
* Số epochs cần được lựa chọn vừa đủ, nhằm tránh overfitting nhưng vẫn đảm bảo mô hình có đủ thời gian học được các cấu trúc cơ bản trong dữ liệu.
* Do ở vòng đầu tiên, không gian biểu diễn của mô hình còn rộng mở, hệ số lambda internal nên được đặt ở mức vừa hoặc cao để khuyến khích các vector trong ma trận LoRA A phân biệt nhau nhiều nhất có thể. Điều này giúp mô hình tận dụng tối đa khả năng học biểu diễn trong giai đoạn khởi đầu.

Khi mô hình đã học được phần lớn các biểu diễn nền tảng từ vòng đầu tiên, nhiệm vụ các vòng sau chuyển sang việc tinh chỉnh và hoàn thiện biểu diễn thông qua học phần sai số còn lại chưa được biểu diễn đầy đủ. Do đó, các siêu tham số ở các vòng sau cần được điều chỉnh theo hướng giảm độ phức tạp, tăng tính ổn định, nhưng vẫn đảm bảo khả năng học bổ sung thông tin mới mà không trùng lặp với các vòng trước. Cụ thể, chiến lược cấu hình tham số cho các vòng sau như sau:

* Rank: Được giảm dần qua từng vòng (ví dụ: 16 → 8 → 4), theo hướng mô hình hóa ngày càng tinh gọn. Việc giảm rank dựa trên giả định rằng các phần residual còn lại có độ phức tạp thấp hơn so với biểu diễn chính ban đầu.
* Learning rate: Cần giảm nhẹ nhằm đảm bảo quá trình tối ưu ổn định hơn và tránh phá vỡ các biểu diễn đã học ở các vòng trước.
* Số epochs: tăng nhẹ ở các vòng sau vì phần residual thường nhỏ hơn, khó học hơn và cần nhiều bước tối ưu hơn để điều chỉnh chính xác.
* Lambda internal: giảm dần ở các vòng sau, vì mô hình đã đạt được mức đa dạng biểu diễn mong muốn. Tuy nhiên, vẫn cần duy trì ở mức vừa giữa các vòng giữa phải để tránh việc các vector trong ma trận A mới học bị trùng lặp nhau.
* Lambda external: Đây là yếu tố quan trọng ở các vòng sau và cần được tăng. Ở các vòng giữa Mục tiêu là đảm bảo rằng biểu diễn được học trong các LoRA module sau là mới mẻ và không lặp lại thông tin đã học từ các adapter trước đó, thông qua ràng buộc trực giao giữa các ma trận A, có thể bắt đầu tăng dần lambda external. Tuy nhiên, ở vòng cuối, ràng buộc trực giao giữa các biểu diễn mới và adapter trước nên được hạn chế tối đa, thậm chí có thể đưa về 0, nhằm tránh loại bỏ những hướng biểu diễn tuy có thể trùng lặp nhưng lại cần thiết để học nốt phần thông tin còn sót lại

Chiến lược này giúp mô hình tiến hành tinh chỉnh liên tiếp trên nền biểu diễn đã có, theo nguyên tắc residual learning, trong khi vẫn duy trì được tính độc lập và bổ sung giữa các vòng. Nhờ đó, toàn bộ quá trình huấn luyện giữ được hiệu quả biểu diễn cao, giảm trùng lặp thông tin, và tối ưu hóa tài nguyên huấn luyện thông qua kiểm soát rank và learning rate hợp lý.

Chiến lược huấn luyện mô hình Chain of LoRA cần được điều chỉnh linh hoạt theo quy mô của tập dữ liệu, nhằm tối ưu hiệu quả học và tránh các hiện tượng như overfitting hoặc underfitting. Do mô hình được chủ yếu train trên tập dưới 10.000 mẫu nên có một số chiến lược để tối ưu khả năng học như sau:

* Sử dụng ít vòng lặp hơn, thường chỉ 2-3 vòng. Do các vòng sau rank giảm ở mức quá nhỏ, dẫn đến các vòng sau gần như không có tác động đến mô hình.
* Lambda internal và lambda external cao hơn, để tăng cường ràng buộc trực giao, đảm bảo mô hình không học trùng lặp hoặc quá khớp với mẫu ít.
* Vòng đầu, số epochs ở mức vừa phải, do không gian học còn rộng. Tuy nhiên càng về các vòng cuối, không gian học càng thu hẹp. Điều này dẫn đến việc mô hình khó tối ưu hóa các tham số LoRA mới trong điều kiện hạn chế, đặc biệt là khi các thông tin còn sót lại có xu hướng phức tạp, nhiễu hoặc nằm ở vùng biên của không gian biểu diễn. Do đó, tại các vòng sau, số epochs phải tăng lên đáng kể để đảm bảo mô hình có đủ thời gian học ổn định và tránh việc dừng sớm khi mô hình chưa kịp hội tụ. Việc tăng epochs trong trường hợp này không nhằm mục tiêu học thêm quá nhiều, mà để mô hình có cơ hội khám phá hết không gian học rất hẹp và khó hội tụ này, đặc biệt trong bối cảnh dữ liệu hạn chế.
* Alpha trong LoRA nên được giảm nhẹ để hạn chế khuếch đại quá mức. Bài viết tính toán alpha bằng tỉ lệ thuận căn bậc hai của rank theo công thức của [14].

Như vậy, việc tinh chỉnh mô hình DeepSeek-R1 bằng chiến lược Chain of LoRA kết hợp ràng buộc trực giao giúp mô hình tận dụng hiệu quả biểu diễn đã học ở các vòng trước, mở rộng khả năng học bổ sung thông tin mới một cách có kiểm soát. Trong bối cảnh dữ liệu huấn luyện hạn chế, cách tiếp cận này giúp cân bằng giữa khả năng generalization và độ đa dạng biểu diễn, đồng thời tối ưu tài nguyên tính toán bằng việc kiểm soát rank và learning rate hợp lý.

# Experiments And Results

## Data collection

Dữ liệu của chúng tôi được thu thập từ hệ thống quản lý Jira nội bộ và các tài liệu liên quan đến quy trình xử lý hóa đơn điện tử. Trong đó, các tệp Excel dạng FAQ là Trong quá trình triển hệ thống hoá đơn điện tử theo thông tư 70. Bên cạnh đó, một phần dữ liệu được sinh tự động từ nội dung các tài liệu hướng dẫn chính thức được ban hành từ 25/05/2025, thông qua mô hình ChatGPT để đảm bảo độ phủ và tính đa dạng của tập dữ liệu. Dữ liệu tập trung vào 2 bộ phận:

Bộ phận developer: tập trung vào hướng dẫn dử dụng base của hệ thống để code các chức năng mở rộng và các tài liệu tích hợp hệ thống hoá đơn điện tử được publish cho bên thứ 3.

Bộ phận it help desk: tập trung vào hỗ trợ khách hàng và hướng dẫn các quy trình, các quy định về hoá đơn theo thông tư 70.

Tập dữ liệu được phân chia theo tỷ lệ "vàng" giữa dữ liệu thật và dữ liệu sinh, với 66,6% là dữ liệu thật (tương đương 1.000 mẫu), và 33,3% là dữ liệu được generate (tương đương 618 mẫu), trên tổng số 1.618 mẫu.

Dữ liệu sau khi tổng hợp được tiền xử lý, lọc trùng lặp, và định dạng lại theo chuẩn chat-format đầu vào cho mô hình ngôn ngữ. Bảng tổng hợp phân bố dữ liệu và đặc điểm thống kê được trình bày trong Bảng 2:

**Table 2.** Average token length and quantity statistics by data source

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Data Source | Number of Samples | Median | Longest | Average Length |
| Jira(Real data) | 550 | 96 | 367 | 111.12 |
| Excel | 450 | 101 | 185 | 102.84 |
| GPT Generate | 618 | 88 | 161 | 88.35 |
| Total | 1618 | 101 | 237.67 | 100.1 |

## Experiments

Quá Trong thực nghiệm, mô hình được sử dụng là DeepSeek-R1-Distill-Qwen-1.5B, một phiên bản nhẹ nhưng hiệu suất cao của dòng DeepSeek, với khả năng hỗ trợ tiếng Việt và suy luận logic tốt. Đây là mô hình có 1.5 tỷ tham số, được phát hành mã nguồn mở, giúp dễ dàng triển khai và tinh chỉnh trong các môi trường doanh nghiệp vừa và nhỏ mà không yêu cầu hạ tầng tính toán quá mạnh.

Quá trình huấn luyện được thực hiện theo chiến lược Chain of LoRA, trong đó mỗi vòng sẽ huấn luyện một module LoRA mới trên nền mô hình đã được cập nhật từ vòng trước. Các module LoRA sẽ lần lượt học residual còn lại của trọng số chưa được điều chỉnh, đồng thời áp dụng ràng buộc trực giao để đảm bảo tính đa dạng biểu diễn và hạn chế trùng lặp thông tin giữa các vòng.  Bảng 3 là thông tin cụ thể về tham số của mỗi vòng.

**Table 3.** Parameters of each round

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Round | Rank | Epochs | Learning Rate | λ internal | λ external | Reason |
| 1 | 16 | 5 | 2e-5 | 0.1 | 0.0 | Vòng đầu tiên cần học biểu diễn nền mạnh nhất |
| 2 | 8 | 7 | 1.5e-5 | 0.05 | 0.05 | Vòng 2 bắt đầu học phần residual |
| 3 | 4 | 9 | 1e-5 | 0.0 | 0.1 | Vòng cuối dùng học để tinh chỉnh phần sai số nhỏ còn lại |

Do yêu cầu thực tế về chi phí và tính khả thi, quá trình huấn luyện được triển khai trên nền tảng Apple MacBook sử dụng chip M-series với GPU tích hợp hỗ trợ Metal Performance Shader (MPS). Mặc dù không mạnh bằng các GPU NVIDIA cao cấp, nhưng MPS vẫn đủ khả năng chạy các mô hình LLM cỡ vừa như DeepSeek-R1-Distill-Qwen-1.5B với batch size nhỏ và chiều dài token hợp lý.

Cấu hình môi trường cụ thể:

* Thiết bị: MacBook Pro M2 với 16GB RAM (hoặc tương đương)
* Phần mềm: Python 3.10, PyTorch 2.x, Transformers 4.41+, PEFT 0.9, Datasets
* Thiết lập MPS: Bật fallback và điều chỉnh watermark để tránh lỗi bộ nhớ khi training.

## Results

Model được training trong vòng 3 vòng, mỗi vòng loss đều giảm về khoảng 3.0-4.0 và dao động trong khoảng đó. Vòng 1 loss giảm mạnh từ 86.98 về 2.96, chứng tỏ adapter học nhanh và đảm bảo xây dựng được kiến thức nền tảng. Vòng 2 bắt đầu từ 27.97 giảm dần và hội tụ quanh 3.0, adapter 2 củng cố hướng học từ vòng 1, giảm dao động và làm mịn biểu diễn. Adapter cuối chỉ giảm nhẹ từ 8.03 về quanh 4. Càng vòng sau loss cuối thường có xu hướng tăng, điều này có thể giải thích do rank thấp khiến mô hình bị giới hạn khả năng biểu diễn, đồng thời vẫn bị ép trực giao khiến việc học càng khó hơn. Tuy nhiên do giới hạn của tập dữ liệu và token, đây là mức loss có thể chấp nhận được, đồng thời loss không giảm quá sâu đảm bảo mô hình không bị overfit.

A graph of a graph

AI-generated content may be incorrect.

**Figure 5.** Training result

Kết quả test trên tập test với các câu hỏi được biến tấu hoặc copy từ tập dữ liệu cho thấy mô hình có sự khác biệt rõ rệt trong việc tiếp thu các dạng kiến thức cụ thể:

* Với những kiến thức dạng định nghĩa như hoá đơn là gì, hoá đơn máy tính tiền là gì, mô hình diễn giải chính xác hoặc gần chính xác với định nghĩa.
* Với những câu hỏi mô tả tình huống như sai tên người mua hàng, sai thành tiền hoá đơn, mô hình trả lời khá chuẩn xác hướng xử lý. Tuy nhiên với các hình huống phức tạp hơn, mô hình vẫn chỉ trả lời chung chung chứ không đưa ra câu trả lời đúng kỳ vọng ví dụ đã lập hoá đơn thay thế mà vẫn sai.
* Với những câu hỏi có chứa thông tin lỗi cực dài ví dụ: “-1:Bộ MST, ký hiệu mẫu số, ký hiệu và số hóa đơn không duy nhất\_Kiểm tra lại định dạng dữ liệu”, mô hình hoàn toàn không hiểu được và đưa ra câu trả lời linh tinh, có thể do mô hình không hiểu được cấu trúc câu và hiểu được ngữ cảnh của câu hỏi.
* Với những câu hỏi về code về dotnet, react, mô hình gần như không học được gì, tuy nhiên bắt đầu có dấu hiệu biết được một số thuật ngữ như BaseController. Với những câu hỏi dạng mã lỗi như ERR, với các tham số bình thường mô hình vẫn không hiểu được. Tuy nhiên nếu tăng số lượng epoch lên mức thật cao như từ 10-15, mô hình bắt đầu hiểu được những mã lỗi như ERR:5; còn các mã lỗi có sự tương đồng nhất định như ERR:1, ERR:10-19, mô hình hoặc trả lời trùng lặp hoặc trả lời chung chung .Điều này mở ra kỳ vọng cho việc mô hình nhiều tham số hơn có thể hiểu và trả lời được câu hỏi.

## Conclusion And Future Work

Mô hình đã thành công trong mục tiêu ban đầu là học được kiến thức nền và phản ứng hợp lý với tình huống dạng support/dev, đặc biệt với dữ liệu ngắn và có cấu trúc. Mặc dù không đạt đến khả năng tổng quát hóa mạnh (generalization) cho toàn bộ các loại lỗi hoặc câu hỏi kỹ thuật, mô hình vẫn cho thấy tiềm năng, khi các mã lỗi không có dạng trùng lặp (ERR:5, ERR:7) bắt đầu được học đúng nếu tăng số epoch. Sự tiến bộ rõ rệt giữa các vòng huấn luyện chứng minh hiệu quả của kỹ thuật chaining và orthogonal LoRA để tăng độ đa dạng và tránh overfitting.

Một vài nguyên nhân khiến khả năng học tập hạn chế của mô hình có thể do:

* Khả năng hiểu ngữ cảnh còn hạn chế, đặc biệt trong các câu hỏi mang nhiều tầng nghĩa hoặc logic. Điều này một phần do token length ngắn, một phần do kiến trúc model nhỏ không đủ năng lực biểu diễn ngữ nghĩa sâu.
* Câu hỏi về mã lỗi kỹ thuật hoặc code (như .NET, React) gần như không được mô hình học, có thể vì không đủ dữ liệu huấn luyện chuyên biệt hoặc vì các câu hỏi này yêu cầu sự hiểu biết nền tảng lập trình – vốn mô hình này không được chuyên biệt cho lập trình như các bản deepseek coder.
* Các lỗi có pattern token trùng nhau (ERR:1 và ERR:10–19) dễ gây nhiễu trong vector biểu diễn, làm mô hình hiểu sai hoặc trả lời trùng lặp

Trong điều kiện có nhiều dữ liệu và tài nguyên hơn, mô hình có thể Sử dụng các model lớn hơn (như DeepSeek-R1 7B hoặc Qwen 7B) sẽ giúp cải thiện khả năng diễn giải và học ngữ cảnh dài, từ đó giúp mô hình phân biệt tốt hơn các lỗi có biểu thức tương tự.

Mô hình có thể tách riêng các task như dev/support thay vì huấn luyện trên một tập dữ liệu chung, đồng thời có thể huấn luyện song song nhưng vẫn giữ điều kiện trực giao để các task có khả năng chuyên biệt hoá tốt hơn nhưng vẫn giữ được khả năng chia sẻ tri thức giữa các task.

# References

|  |  |
| --- | --- |
| [1] | Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, "LoRA: Low-Rank Adaptation of Large Language Models," 17 Jun 2021. [Online]. Available: https://arxiv.org/abs/2106.09685. [Accessed 12 05 2025]. |
| [2] | D. Kalajdzievski, "Scaling Laws for Forgetting When Fine-Tuning Large Language Models," 11 Jan arxiv. [Online]. Available: https://arxiv.org/abs/2401.05605. [Accessed 13 05 2025]. |
| [3] | Hong Liu, Saisai Gong, Yixin Ji, Kaixin Wu, Jia Xu, Jinjie Gu, "Boosting LLM-based Relevance Modeling with Distribution-Aware Robust Learning," 17 Dec 2024. [Online]. Available: https://arxiv.org/abs/2412.12504. [Accessed 13 05 1015]. |
| [4] | DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang,..100 additional authors not shown), "DeepSeek-V3 Technical Report," 2025 . |
| [5] | Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, Luke Zettlemoyer, "QLoRA: Efficient Finetuning of Quantized LLMs," 23 May 2023. [Online]. Available: https://arxiv.org/abs/2305.14314. [Accessed 25 May 2025]. |
| [6] | Haizhou Shi, Zihao Xu, Hengyi Wang, Weiyi Qin, Wenyuan Wang, Yibin Wang, Zifeng Wang, Sayna Ebrahimi, Hao Wang, "Continual Learning of Large Language Models: A Comprehensive Survey," 25 Apr 2024 . [Online]. Available: https://arxiv.org/abs/2404.16789. [Accessed 25 May 2025]. |
| [7] | DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi. (100 additional authors not shown), "DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning," 22 Jan 2025. [Online]. Available: https://arxiv.org/abs/2501.12948. [Accessed 25 May 2025]. |
| [8] | "Fine-Tuning DeepSeek R1 (Reasoning Model)," 27 Jan 2025. [Online]. Available: https://www.datacamp.com/tutorial/fine-tuning-deepseek-r1-reasoning-model. [Accessed 25 May 2025]. |
| [9] | E. Page, "Fine-Tuning LLMs using Intel Xeon CPUs," 18 Mar 2025. [Online]. Available: https://lenovopress.lenovo.com/lp2179-fine-tuning-llms-using-intel-xeon-cpus. [Accessed 25 May 2025]. |
| [10] | DeepSeek-AI, "DeepSeek-R1-Distill-Qwen-7B," [Online]. Available: https://huggingface.co/deepseek-ai/DeepSeek-R1-Distill-Qwen-7B. |
| [11] | Sze Wai Yuen, Liam Li, Kevin Musgrave, "Finding the best LoRA parameters," 11 Sep 2024. [Online]. Available: https://www.determined.ai/blog/lora-parameters. [Accessed 7 Jun 2025]. |
| [12] | H. Face, "Seq2Seq Parameters," [Online]. Available: https://huggingface.co/docs/autotrain/en/seq2seq\_params. [Accessed 07 Jun 2025]. |
| [13] | S. Raschka, "Practical Tips for Finetuning LLMs Using LoRA (Low-Rank Adaptation)," 19 Nov 2023. [Online]. Available: https://magazine.sebastianraschka.com/p/practical-tips-for-finetuning-llms. [Accessed 08 Jun 2025]. |
| [14] | D. Kalajdzievski, "A Rank Stabilization Scaling Factor for Fine-Tuning with LoRA," 28 Nov 2023. [Online]. Available: https://arxiv.org/abs/2312.03732. [Accessed 08 Jun 2025]. |
| [15] | Xiao Wang, Tianze Chen, Qiming Ge, Han Xia, Rong Bao, Rui Zheng, Qi Zhang, Tao Gui, Xuanjing Huang, "Orthogonal Subspace Learning for Language Model Continual Learning," 22 Oct 2023. [Online]. Available: https://arxiv.org/abs/2310.14152. [Accessed 08 Jun 2025]. |
| [16] | Quan Cheng, Yuanyu Wan, Lingyu Wu, Chenping Hou, Lijun Zhang, "Continuous Subspace Optimization for Continual Learning," 17 May 2025. [Online]. Available: https://arxiv.org/abs/2505.11816. [Accessed 17 Jun 2025]. |
| [17] | Ruiyu Wang, Sen Wang, Xinxin Zuo, Qiang Sun, "Lifelong Learning with Task-Specific Adaptation: Addressing the Stability-Plasticity Dilemma," 08 Mar 2025. [Online]. Available: https://arxiv.org/abs/2503.06213. [Accessed 18 Jun 2025]. |
| [18] | Wenhan Xia, Chengwei Qin, Elad Hazan, "Chain of LoRA: Efficient Fine-tuning of Language Models via Residual Learning," 08 Jan 2024. [Online]. Available: https://arxiv.org/abs/2401.04151. [Accessed 08 Jun 2025]. |
| [19] | K. Miltos, H. Robin and M. Arnab, "Metro railway safety: An analysis of accident precursors," *Safety Science,* vol. 50, no. 7, pp. 1535-1548, 2012. |
| [20] | Y. Cao, Y. An, S. Shuai, G. Xie and Y. Sun, "A statistical study of railway safety in China and Japan 1990–2020," *Accident Analysis & Prevention,* vol. 175, 2022. |