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 2024

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:

1. Assoc. Prof. Phan Duy Hung
2. Dr. Vu Thu Diep

© Copyright by Dinh Cong Bang 2024

DeepSeek-based ChatBot system supports work management

Dinh Cong Bang

Degree Master of Software Engineering

FPT University

2024

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]. Phương pháp này bảo toàn kiến thức nền nhằm giảm thiểu hiện tượng “quên lệch” (catastrophic forgetting) [2].Ngoài ra, tinh chỉnh đa giai đoạn giúp mô hình duy trì khả năng tổng quát trên nhiều nhiệm vụ [3].Nhờ đó, hệ thống cân bằng tính chuyên môn hóa và khả năng tổng quát, mang lại giải pháp quản lý công việc hiệu quả.

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

.

Table of Contents

[**Acknowledgments 2**](#_heading=h.2et92p0)

[**Table of Contents 3**](#_heading=h.tyjcwt)

[**List of Figures 4**](#_heading=h.3dy6vkm)

[**Chapter 1 Introduction 5**](#_heading=h.1t3h5sf)

[**1**](#_heading=h.4d34og8) **Problem & Motivation 5**

[**2**](#_heading=h.17dp8vu) **Existing work 6**

[**3**](#_heading=h.3rdcrjn) **Problem & Motivation 7**

[**Chapter 2 Background Study 9**](#_heading=h.26in1rg)

[**1**](#_heading=h.lnxbz9) **Business of Fashion Brands 9**

[**2**](#_heading=h.1ksv4uv) **Modular Software Architecture 10**

[**3**](#_heading=h.44sinio) **ML Models in Production 11**

[**Chapter 3 System Design 13**](#_heading=h.2jxsxqh)

[**1**](#_heading=h.z337ya) **Business Flow 13**

[**2**](#_heading=h.1y810tw) **Design Diagrams 14**

[2.1](#_heading=h.4i7ojhp) Context Diagram 15

[2.2](#_heading=h.1ci93xb) Container Diagram 16

[2.3](#_heading=h.2bn6wsx) Component Diagram 18

[**Chapter 4 Implementation & Discussion 21**](#_heading=h.3as4poj)

[**1**](#_heading=h.1pxezwc) **Try-On & Catalog Module 21**

[**2**](#_heading=h.ihv636) **Try-on Integration into an E-commerce Application 26**

[**3**](#_heading=h.41mghml) **MLOps Implementation 29**

[**4**](#_heading=h.vx1227) **Microservice Implementation 31**

[**5**](#_heading=h.4f1mdlm) **Discussion, Future work & Conclusion 34**

[**References 38**](#_heading=h.2u6wntf)

List of Figures

[Figure 1. The design process in the Textile & Clothing industry. 10](#_heading=h.35nkun2)

[Figure 2. GARMENTO system main flow 14](#_heading=h.3j2qqm3)

[Figure 3. Context Diagram of the GARMENTO Virtual Try-on system. 15](#_heading=h.2xcytpi)

[Figure 4. The Container Diagram of the GARMENTO Virtual Try-on System. 17](#_heading=h.3whwml4)

[Figure 5. The Component Diagram of GARMENTO Virtual Try-on System. 19](#_heading=h.qsh70q)

[Figure 6. User Interface of Try-On function. 21](#_heading=h.49x2ik5)

[Figure 7. Google sign-on screen. 22](#_heading=h.2p2csry)

[Figure 8. Obtaining service-level credentials for the Try-On & Catalog module. 23](#_heading=h.147n2zr)

[Figure 9. Performing virtual try-on with preset. 24](#_heading=h.3o7alnk)

[Figure 10. Catalogue status transitions. 25](#_heading=h.23ckvvd)

[Figure 11. Virtual try-on plugin integrated into an example e-commerce web application. 27](#_heading=h.32hioqz)

[Figure 12. Sequence diagram for E-commerce Application integration. 28](#_heading=h.1hmsyys)

[Figure 13. Model Registry’s Sequence Diagram. 29](#_heading=h.2grqrue)

[Figure 14. Service decomposition in the GARMENTO system 31](#_heading=h.3fwokq0)

[Figure 15. High-level deployment diagram (non-UML) of the GARMENTO prototype. 33](#_heading=h.1v1yuxt)

1. Introduction

# Problem & 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.

# Existing work

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.

Một trong những công trình có ảnh hưởng lớn trong lĩnh vực tinh chỉnh LLM là nghiên cứu của [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.

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.

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.

Cuối cùng, để triển khai các mô hình tinh chỉnh này vào thực tế, một vấn đề khác là hạn chế tài nguyên phần cứng. [9] đã chứng minh khả năng tinh chỉnh mô hình LLM quy mô trung bình (1.3B–7B tham số) bằng kỹ thuật LoRA ngay trên CPU thông qua kỹ thuật tối ưu tính toán ma trận và lượng tử hóa nhẹ. Điều này đặc biệt quan trọng với các doanh nghiệp nhỏ hoặc tổ chức không có GPU chuyên dụng, giúp hạ thấp rào cản triển khai hệ thống AI trong nội bộ.

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, 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 mô hình ngôn ngữ được phân tách theo các tác vụ cụ thể như: sinh báo cáo tổng hợp, gợi ý hướng xử lý công việc, và tạo hướng dẫn nhân sự mới. Việc chia nhỏ mô hình theo chức năng, thay vì sử dụng một mô hình duy nhất, cho phép tối ưu hóa hiệu năng và khả năng bảo trì trong dài hạn.

Bài viết áp dụng kỹ thuật LoRA để tinh chỉnh mô hình DeepSeek-R1 với dữ liệu nội bộ từ Jira và dữ liệu chat nội bộ. Những nghiên cứu [1] [2] cho thấy LoRA không chỉ giảm đáng kể chi phí tính toán và lưu trữ mà còn giúp duy trì tri thức nền của mô hình, từ đó giảm thiểu hiện tượng "quên lãng thảm họa" thường gặp khi fine-tuning LLM. Việc áp dụng LoRA trong bài toán quản lý công việc là một điểm mới có giá trị thực tiễn.

Bài viết xây dựng một quy trình xử lý dữ liệu từ hệ thống doanh nghiệp sang định dạng huấn luyện cho LLM, bao gồm tiền xử lý, cấu trúc hóa log, ánh xạ tác vụ và gán nhãn cho từng tình huống. Đây là bước quan trọng nhằm chuyển đổi dữ liệu phi cấu trúc thành tri thức đầu vào cho mô hình ngôn ngữ, đồng thời mở ra khả năng tự động hóa các quy trình ra quyết định nội bộ.

Bài viết minh chứng cho việc thu hẹp khoảng cách giữa nghiên cứu học thuật và ứng dụng doanh nghiệp. Các kết quả và quy trình được xây dựng có thể tái sử dụng cho các hệ thống hỗ trợ khác, hoặc mở rộng sang các lĩnh vực như chăm sóc khách hang.

1. 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.

Phổ biến nhất là Jira, Trello và Asana Jira, với khả năng tùy chỉnh cao và tích hợp mạnh mẽ, phù hợp với các nhóm phát triển phần mềm và dự án phức tạp. Trello, với giao diện trực quan và dễ sử dụng, thích hợp cho các nhóm nhỏ và dự án đơn giản . Asana cung cấp sự cân bằng giữa tính năng và độ phức tạp, phù hợp với nhiều loại hình dự án khác nhau [10] .

Các công cụ quản lý công việc không chỉ hỗ trợ theo dõi tiến độ dự án mà còn tạo ra một hệ sinh thái dữ liệu phong phú và có cấu trúc rõ ràng. Mỗi tác vụ trong các hệ thống này thường đi kèm với thông tin chi tiết như mô tả bằng ngôn ngữ tự nhiên, trạng thái, thời gian xử lý, người phụ trách và mối quan hệ với các tác vụ khác. Dữ liệu này cung cấp đầu vào lý tưởng cho các mô hình AI đa phương thức hoặc nhận thức ngữ cảnh, cho phép hiểu sâu hơn về quy trình làm việc và hỗ trợ ra quyết định thông minh hơn.

Hơn nữa, các công cụ như Jira và Trello cung cấp API mạnh mẽ, cho phép truy xuất và xử lý dữ liệu một cách hiệu quả.Điều này tạo điều kiện thuận lợi cho việc xây dựng pipeline huấn luyện cho các mô hình LLMs, đặc biệt là trong việc tinh chỉnh mô hình để phù hợp với ngữ cảnh doanh nghiệp cụ thể. Khả năng truy cập vào dữ liệu lịch sử và tương tác người dùng giúp mô hình học hỏi từ các mẫu hành vi thực tế, nâng cao độ chính xác và hiệu quả trong các tác vụ như gợi ý công việc tiếp theo, tạo checklist tự động và tổng hợp báo cáo

Bên cạnh các hệ thống quản lý công việc chuyên dụng, nhiều doanh nghiệp hiện nay vẫn duy trì mô hình quản lý truyền thống, dựa trên các công cụ phổ thông như Excel, tài liệu/văn bản nội bộ hoặc thậm chí là trao đổi trực tiếp qua các ứng dụng trò chuyện như Telegram, Zalo, Viber. Trong các mô hình này, dữ liệu công việc tồn tại ở dạng phi cấu trúc hoặc bán cấu trúc, các bên cung cấp dịch vụ nhắn tin cũng thường không cung cấp công cụ để khai thác lượng dữ liệu này. Gây khó khăn cho việc tổng hợp và phân tích tự động.

Tuy nhiên, [11] cho thấy rằng các mô hình ngôn ngữ lớn (LLMs) có thể được huấn luyện để hiểu và trích xuất thông tin từ dữ liệu phi cấu trúc, như nội dung email, lịch sử trò chuyện hoặc nội dung các file văn bản. Bằng cách sử dụng kỹ thuật như Information Extraction, Semantic Segmentation hoặc Chain-of-Thought prompting, mô hình có thể tái tạo lại luồng công việc từ dữ liệu rời rạc.

Ngoài ra, thông qua việc phân tích metadata như thời gian gửi mail, người nhận, tần suất trả lời hoặc chủ đề trò chuyện, AI có thể xây dựng một graph tương tác công việc nhằm mô hình hóa cấu trúc làm việc ẩn của tổ chức. Cách tiếp cận này được thử nghiệm trong nghiên cứu [11], cho thấy tiềm năng trong việc phát hiện luồng công việc (workflow patterns) trong các doanh nghiệp chưa sử dụng hệ thống quản lý hiện đại.

Khác với dữ liệu từ Jira hay Trello, dữ liệu truyền thống thường yêu cầu bước công phu hơn. Tuy nhiên, với sự phát triển của các mô hình AI tiên tiến cho tác vụ information extraction, việc tái cấu trúc quy trình làm việc từ dữ liệu rời rạc là khả thi.

# Deepseek model

DeepSeek-R1-Distill-Qwen-7B là một mô hình ngôn ngữ lớn (LLM) với 7 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 [12]:

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-7B 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-7B 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) for Efficient Fine-Tuning

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 WW 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]. Thay vì cập nhật toàn bộ trọng số của mô hình, 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*, Thay vào đó, nó giữ nguyên *W* và chèn thêm một phần hiệu chỉnh có hạng thấp, ký hiệu là ΔW. Mô hình có một **ma** trận trọng số W∈R*d×k* – nghĩa là có d hàng và k cột. Như nếu fine-tune ta phải cập nhật hết *dxk* tham số, thay vào đó LORA sẽ thêm vào một ma trận mới được tính từ tích của hai ma trận nhỏ hơn theo công thức ΔW=B⋅A, trong đó A∈Rr×k, B∈Rd×r và r≪min(d,k). Sau đó ta tính W′=W+α⋅ΔW trong đó α là hệ số điều chỉnh, trong đó α được 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; theo đó 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 [13], 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 [14]. Sebastian Raschka cũng đưa ra nguyên tắcα / r = 2 [15] 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 [16] 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 [16].

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.

Nghiên cứu [17] cũng cho thấy rằng bằng cách tập trung vào orthogonal low-rank subspaces hay có thể gọi là O‑LoRA, LoRA có khả năng hạn chế hiện tượng catastrophic forgetting. Theo đó, thay vì dùng chung một adapter cho tất cả nhiệm vụ, O-LoRA dùng mỗi adapter riêng cho từng nhiệm vụ, Để các adapter này không ảnh hưởng lẫn nhau, O-LoRA ràng buộc các adapter hoạt động ở orthogonal subspaces. Phương pháp này giúp kiểm soát việc học của mô hình, không mất đi kiến thức cũ, mà vẫn linh hoạt cập nhật cái mới. **(phần này khá khó hiểu và em đọc các công thức trong tài liệu thì em cũng ko hiểu cặn kẽ được cách nó hoạt động, cái này chắc em hỏi thêm thầy rồi sẽ cập nhật sau ạ, đoạn này em nhờ GPT tóm tắt tài liệu và cách triển khai của nó trước ạ)**

# Multi-Stage Fine-Tuning

Trong quá trình phát triển Large language model(LLMs) cho ứng dụng thực tế, việc huấn luyện mô hình trên toàn bộ dữ liệu cùng lúc thường bị ràng buộc về tài nguyên, tính không đồng nhất của dữ liệu, và yêu cầu kiểm soát hiệu quả học của mô hình. [2] chỉ ra rằng, ngay cả khi chỉ sử dụng một số lượng tham số hạn chế như trong LoRA, mô hình vẫn có nguy cơ quên kiến thức đã học trước đó, vấn đề trở nên nghiêm trọng hơn khi tăng số tham số được cập nhật hoặc số bước huấn luyện. Multi-Stage Fine-Tuning là một hướng tiếp cận nhằm khắc phục những hạn chế này bằng cách chia nhỏ quá trình tinh chỉnh thành các bước huấn luyện tuần tự, mỗi bước tập trung vào một tập con dữ liệu hoặc một miền nhiệm vụ cụ thể [2] [18] .

Một hướng tiếp cận nổi bật trong huấn luyện nhiều giai đoạn là ProMoT(Prompt Tuning with MOdel Tuning), được đề xuất trong [18] nhằm tăng khả năng tổng quát hoá của mô hình. ProMoT chia quá trình fine-tuning thành hai giai đoạn tách biệt. Trong giai đoạn đầu, mô hình được huấn luyện bằng kỹ thuật prompt tuning, với phần backbone được giữ nguyên. Thay vì cập nhật toàn bộ trọng số, hệ thống chỉ học các prompt embeddings/prompt vectors giúp mô hình làm quen với dạng thức đầu vào và đầu ra của nhiệm vụ, đầu ra của bước này là prompt embeddings/prompt vectors. Sau khi đã học được các prompt embeddings mô hình đã làm quen với dạng thức đầu vào và đầu ra của nhiệm vụ, mô hình bước vào giai đoạn thứ hai – model fine-tuning. Tại đây, prompt embeddings đóng vai trò như một contextual anchor cho mô hình. Mục tiêu của bước này là tinh chỉnh trọng số thực của mô hình. Cách tiếp cận này giúp mô hình học sâu hơn về ngữ cảnh của từng câu hỏi mà không quá phụ thuộc vào format cố định, từ đó giảm hiện tượng format specialization, một vấn đề thường gây suy giảm khả năng thích ứng của mô hình trong các ngữ cảnh khác biệt. Khi áp dụng vào hệ thống quản lý công việc, ProMoT có thể giúp mô hình DeepSeek lọc ra các câu hỏi không liên quan hoặc các câu hỏi nhạy cảm, tránh được các vấn đề về pháp lý, truyền thông.

Một hướng tiếp cận khác đó là phương pháp Phased Instruction Fine-Tuning (Phased IFT) được đề xuất bởi [19], bằng cách tiếp cận nhiều giai đoạn (multi-stage), dựa trên giả thuyết “progressive alignment” – cho rằng mô hình lớn cần học tuần tự từ các nhiệm vụ đơn giản đến phức tạp để cải thiện khả năng tuân thủ hướng dẫn. Bản chất của phương pháp gồm 3 bước:

Xác định độ khó: Sử dụng mô hình như GPT-4 để đánh giá mỗi bộ ba *(*instruction, input, output) từ tập dữ liệu.

Phân lớp dữ liệu: Dữ liệu được chia thành nhiều giai đoạn (thường là 3) dựa trên ngưỡng điểm từ dễ đến khó. Các ngưỡng này được chọn dựa vào phân phối xác suất và ý kiến chuyên gia.

Fine-tuning theo từng giai đoạn: Mô hình được huấn luyện từng giai đoạn từ dễ đến khó, sử dụng cùng bộ siêu tham số (learning rate, batch size...). Kết quả thí nghiệm cho thấy Phased IFT cải thiện đáng kể tỉ lệ win-rate trong các benchmark so với cách One-off IFT[19]**.**

Trong bài toán quản lý công việc, Phased IFT giúp mô hình DeepSeek thích nghi tuần tự từ nhiệm vụ dễ đến khó.

Việc kết hợp giữa Multi‑Stage Fine‑Tuning với LoRA đã được chứng minh tính khả thi qua phương pháp Chain of LoRA (COLA), được đề xuất bởi [20], COLA triển khai mô hình theo nhiều giai đoạn:

Tune LoRA: Fine-tune module LoRA đầu tiên trên dữ liệu nhiệm vụ đầu tiên.

Tie a knot: Merge adapter đã học vào mô hình gốc để cố định kiến thức đó.

Extend the chain: Khởi tạo adapter LoRA mới và tiếp tục fine-tune trên nhiệm vụ tiếp theo.

Quá trình này lặp lại cho đến khi hoàn thành chuỗi nhiệm vụ. COLA 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. Trên các benchmark thử nghiệm với LLaMA‑2 và OPT, COLA liên tục vượt qua LoRA đơn lẻ về khả năng tổng quát hóa, minh chứng khả năng kết hợp chuỗi kỹ thuật multi-stage và LoRA một cách rất hiệu quả [20].

Việc kết hợp ProMoT với LoRA cũng được chứng minh qua phương pháp PILLOW [21], bằng việc Xây dựng một prompt pool do người dùng định nghĩa trước, mô hình LoRA được fine-tune để học các embedding task-specific, Trước mỗi instruction, họ dùng matching network để chọn prompt thích hợp nhất từ pool dựa trên embedding của câu hỏi. Prompt đã chọn được ghép vào đầu input, sau đó đưa vào mô hình LoRA đã fine-tune, Kỹ thuật huấn luyện sử dụng Reinforcement Learning (RL) để tối ưu độ phù hợp của lựa chọn prompt và đầu ra của mô hình. Kết quả cho thấy: PILLOW có thể đạt được hiệu năng tương đương với full fine-tuning SFT, nhưng chỉ sử dụng phần cứng GPU bình dân, nhờ tận dụng tốt In‑Context Learning vốn có của LLM và khả năng hiệu chỉnh tham số tinh tế bằng LoRA [21].

Từ những nghiên cứu trên, việc kết hợp 2 kỹ thuật Multi-Stage Fine-Tuning và LoRA là khả thi, thực tiễn cũng chứng minh việc kết hợp này còn tối ưu được tài nguyên phần cứng [21] nhưng vẫn đạt hiệu suất tương đương với các biện pháp truyền thống [20] [21].

1. System Design

Befo

# Business Flow

X1.5

# Design Diagrams

X1

## Context Diagram

X2

## Container Diagram

X2

## Component Diagram

x2.5

1. Implementation & Discussion

This .

# Work management system

X2

# Training with basic Vietnamese data

X4

# Training with specialized terminology

X4

# MLOps Implementation

X2

# Microservice Implementation

X3

# Discussion, Future work & Conclusion

X3

Appendix

**Appendix A**. System test cases

List of test cases used in the development of the GARMENTO system is available publicly at: <https://bit.ly/3W4mXUM>. Currently, all critical paths have been tested and passed all the test cases.

**Appendix B.** Latest source code directory

In addition to the monorepo provided in the References list, we provide an organization that clearly shows the decomposed and developed microservices. The Github organization is available at: <https://github.com/garmento-microservices>.

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] | "Asana and Trello: A Comparative Assessment of Project Management Capabilities," March 2024. [Online]. Available: https://www.researchgate.net/publication/379486813\_Asana\_and\_Trello\_A\_Comparative\_Assessment\_of\_Project\_Management\_Capabilities. [Accessed 03 06 2025]. |
| [11] | Cheng’an Wei1,2, Yue Zhao1,2, Yujia Gong1,2, Kai Chen 1, Lu Xiang1,2, and Shenchen Zhu1,2 1Institute of Information Engineering, Chinese Academy of Sciences, China 2School of Cyber Security, University of Chinese Academy of Sciences, China, "arxiv," 06 Sep 2024. [Online]. Available: https://arxiv.org/html/2405.20234v3. [Accessed 02 06 2025]. |
| [12] | DeepSeek-AI, "DeepSeek-R1-Distill-Qwen-7B," [Online]. Available: https://huggingface.co/deepseek-ai/DeepSeek-R1-Distill-Qwen-7B. |
| [13] | 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]. |
| [14] | H. Face, "Seq2Seq Parameters," [Online]. Available: https://huggingface.co/docs/autotrain/en/seq2seq\_params. [Accessed 07 Jun 2025]. |
| [15] | 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]. |
| [16] | 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]. |
| [17] | 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]. |
| [18] | Yihan Wang, Si Si, Daliang Li, Michal Lukasik, Felix Yu, Cho-Jui Hsieh, Inderjit S Dhillon, Sanjiv Kumar, "Two-stage LLM Fine-tuning with Less Specialization and More Generalization," 01 Nov 2022. [Online]. Available: https://arxiv.org/abs/2211.00635. [Accessed 08 Jun 2025]. |
| [19] | Wei Pang, Chuan Zhou, Xiao-Hua Zhou, Xiaojie Wang, "Phased Instruction Fine-Tuning for Large Language Models," 01 Jun 2024. [Online]. Available: https://arxiv.org/abs/2406.04371. [Accessed 08 Jun 2025]. |
| [20] | 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]. |
| [21] | Zhenting Qi, Xiaoyu Tan, Shaojie Shi, Chao Qu, Yinghui Xu, Yuan Qi, "PILLOW: Enhancing Efficient Instruction Fine-tuning via Prompt Matching," 09 Dec 2023. [Online]. Available: https://arxiv.org/abs/2312.05621. [Accessed 08 Jun 2025]. |