Technical Portfolio

Yuzhong Chen

https://russell12138.github.io

Multimodal LLM Research
System Engineering
Embodied Intelligence

Contents

1 Multimodal and LLM Research

1.1 Project 1: Modular Expert Routing System in Large Language Models

Brief Introduction

Paper to be submitted to **ACM Multimedia 2026**.

Large Language Models (LLMs) are typically adapted to new knowledge through full fine-tuning, a process that is computationally expensive and inflexible for domains where information changes frequently. To address this limitation, I designed and implemented the *Modular Expert Routing System (MERS)*, a modular architecture composed of multiple domain-specialized expert models coordinated by a lightweight routing mechanism.

Instead of forcing a single unified model to absorb all domains, MERS routes each input to the most relevant expert, allowing domain-specific updates while preserving system-level coherence. Inspired by Retrieval-Augmented Generation (RAG) and Mixture-of-Experts (MoE) principles, this design improves interpretability, reduces update cost, and enables controlled specialization–generalization trade-offs. The system was evaluated through a series of controlled experiments assessing feasibility, fine-tuning efficiency, and cross-domain generalization.

— Selected Excerpts —

System Architecture. MERS consists of multiple domain-specialized LLM experts and a classifier-based routing module. Each expert is fine-tuned exclusively on a designated subset of the MMLU dataset, while the router predicts the most appropriate expert given an input query. During inference, the router selects a single expert, enabling efficient and interpretable decision-making without requiring joint retraining of all model parameters.

This architecture decouples knowledge updates across domains, allowing individual experts to be updated or replaced independently. Compared to unified fine-tuning, MERS provides a more maintainable and evolvable system design, particularly suited for real-world deployment scenarios with limited compute budgets.

Experimental Design and Evaluation. To rigorously evaluate MERS, I designed three complementary experimental studies. First, feasibility was assessed by benchmarking the system on public question-answering datasets including ARC, BoolQ, and OpenBookQA, ensuring performance remained within a reasonable range of established baselines. Second, a unified model fine-tuned on the same domains was constructed as a baseline to compare accuracy, inference latency, and fine-tuning cost, enabling a quantitative analysis of modular versus unified adaptation strategies.

Finally, generalization was evaluated by testing both MERS and the unified model on unseen MMLU domains. Inputs were forcibly routed to existing experts, allowing analysis of robustness under domain mismatch and empirical characterization of the specialization–generalization trade-off.

Constraints and Design Considerations. The system was developed under realistic constraints, including limited API budgets and restricted computational resources. To mitigate these limitations, compact base models were selected and contingency plans were incorporated into the experimental workflow. These constraints informed architectural and experimental decisions, reinforcing the importance of system-level thinking beyond raw model performance.

Takeaway

This project marked my transition from model-level experimentation to system-level research, where architectural design, controlled evaluation, and real-world constraints jointly shape intelligent behavior. Through MERS, I explored how modularity can improve the maintainability and deployability of large language systems in dynamic knowledge environments.

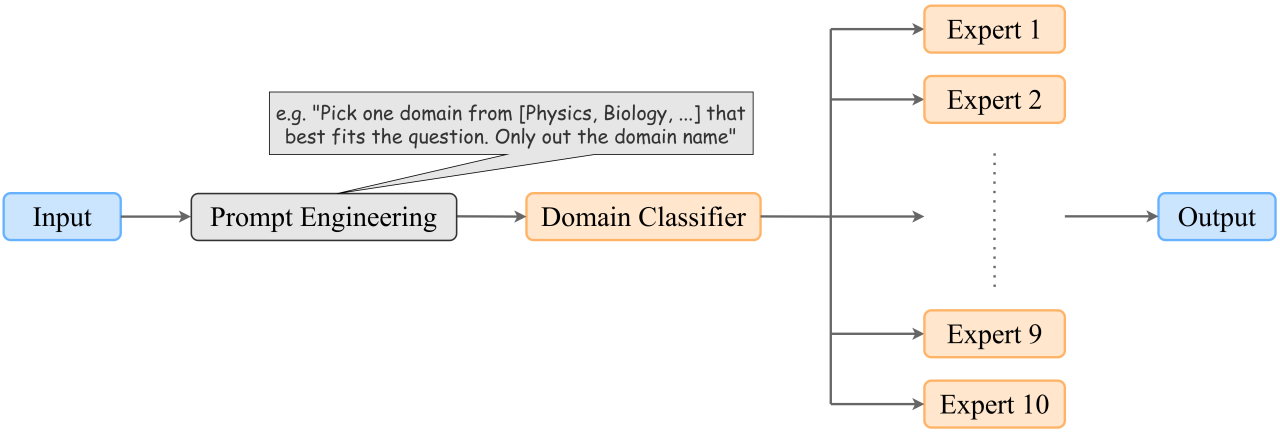

Figure 1: MERS Architecture Design

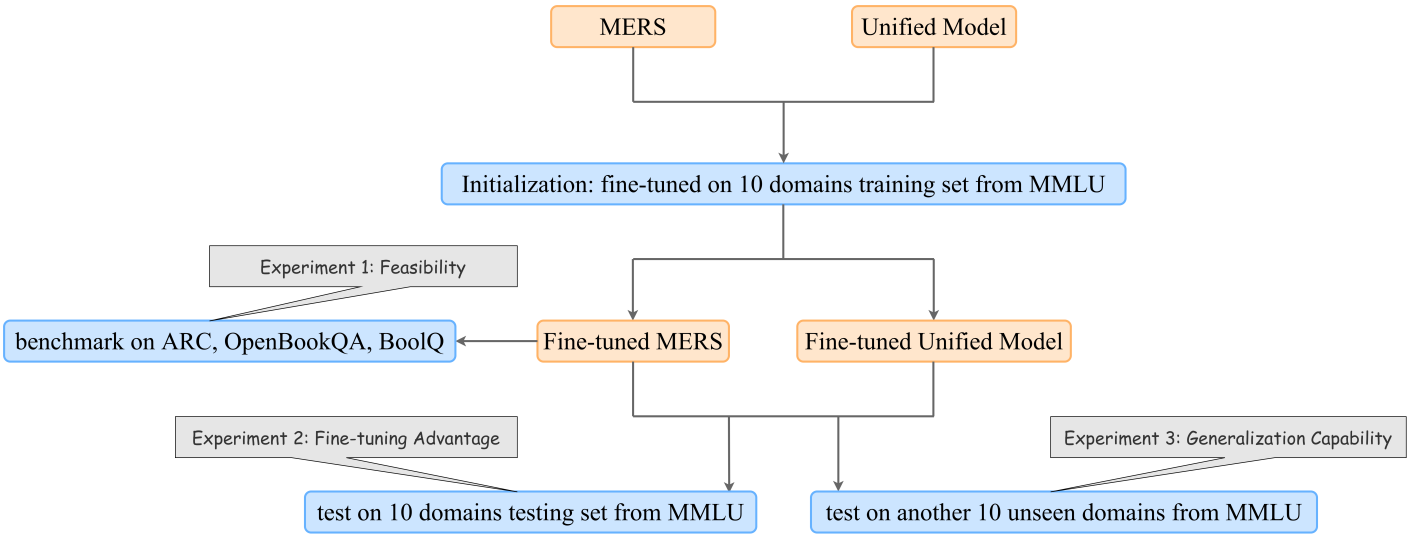

Figure 2: Experiment Design

1.2 Project 2: Modality Injection Order in Multimodal Large Language Models

Brief Introduction

Paper submitted to **ACL 2026**.

Multimodal Large Language Models (MLLMs) integrate heterogeneous inputs such as text, vision, and audio, and are increasingly fine-tuned as pretrained backbones across various downstream classification tasks. While recent studies suggest that the ordering of multimodal inputs may affect model behavior, existing work are largely limited to isolated settings, focusing on a single dataset, task, or model.

To facilitate a systematic investigation of modality injection order, we propose a Modality Order-aware Controlled Architecture for MLLMs namely *MOCA-MLLM*, that explicitly controls the sequence in which different modalities are incorporated into the decoder of a pretrained MLLM. Building upon this framework, we conduct a large-scale empirical study on the impact of modality injection order during fine-tuning, across four MLLMs, six downstream tasks, twelve multimodal datasets, three modalities, and six modality orders.

Across diverse tasks and datasets, our results reveal that modality injection order exhibits systematic, task-dependent patterns: affective recognition tasks such as sentiment and emotion benefit from early visual or acoustic cues, whereas semantic-grounded tasks including sarcasm, humor, hate speech, and deception detection consistently favor early access to textual information, with text-middle configurations emerging as the most consistently suboptimal. These findings highlight modality order as an important yet underexplored design factor in multimodal adaptation, and provide practical guidance for future MLLM fine-tuning and system design.

Takeaway

This project demonstrates my approach to multimodal research as a system-level investigation rather than a purely performance-driven task. By constructing a tightly controlled experimental framework, I was able to isolate and analyze the role of modality injection order within the decoder. The insights gained from this study contribute to my broader research goal of grounding language models in perception and action, and inform principled design choices for future multimodal and embodied AI systems.

1.3 Project 3: Survey of Efficient and Enhanced Attention Mechanisms in Transformers

Brief Introduction

Paper available at: `https://ace.ewapub.com/article/view/10938`.

The Transformer architecture has become a foundational model across natural language processing, computer vision, and multimodal learning, with the attention mechanism serving as its core computational primitive. However, the standard self-attention formulation suffers from limitations in efficiency, scalability, and redundancy, particularly as model size and sequence length increase.

In this project, I co-authored a comprehensive survey that systematically reviews recent advances in attention mechanism design, focusing on both computational efficiency and representational performance. The survey categorizes improvements into multiple dimensions, including linearized attention, sparse attention, multi-head optimization, and low-rank approximations. By synthesizing diverse strands of prior work, this study provides a structured understanding of how attention variants address real-world constraints while preserving or enhancing model capability.

— Selected Excerpts —

Multi-Head Attention Mechanism. In order to further optimize the feature representation, multi-head attention mechanism separately projects original Q, K and V onto a lower dimensional space through a linear layer for h times. Its structure is shown in Figure 2. H different outputs generated by performing Eq. (1) on each set are then combined and additionally projected to the ultimate value. The process can be represented as follows:

$$\mathrm{Att}(Q, K, V) = \mathrm{Concat}(H_1, \ldots, H_h)W^O, \quad H_i = \mathrm{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

This mechanism has similar advancement as convolutional neural networks (CNNs) as it provides multiple output channels to improve feature extraction. Therefore, multi-head self-attention enables attention with various sub-expressions and strengthens the model capability of focusing on different positions.

For specific tasks, improving the multi-head mechanism is often a major direction for improving model performance. Recent methods explore optimization at different stages of the multi-head process, including constraints on projection, attention span, and head combination strategies. Shatter simplifies multi-head attention into a single-head structure while improving efficiency and reducing parameter count. ConvFormer replaces standard projections with dynamic multi-head convolutional attention, achieving significant parameter reduction while improving performance on 3D pose estimation tasks. Other approaches prune redundant heads using relevance-based criteria, demonstrating that a small subset of attention heads often contributes disproportionately to final performance.

These results collectively suggest that naïvely increasing the number of heads may introduce redundancy, and that careful structural redesign of multi-head attention can improve both efficiency and generalization.

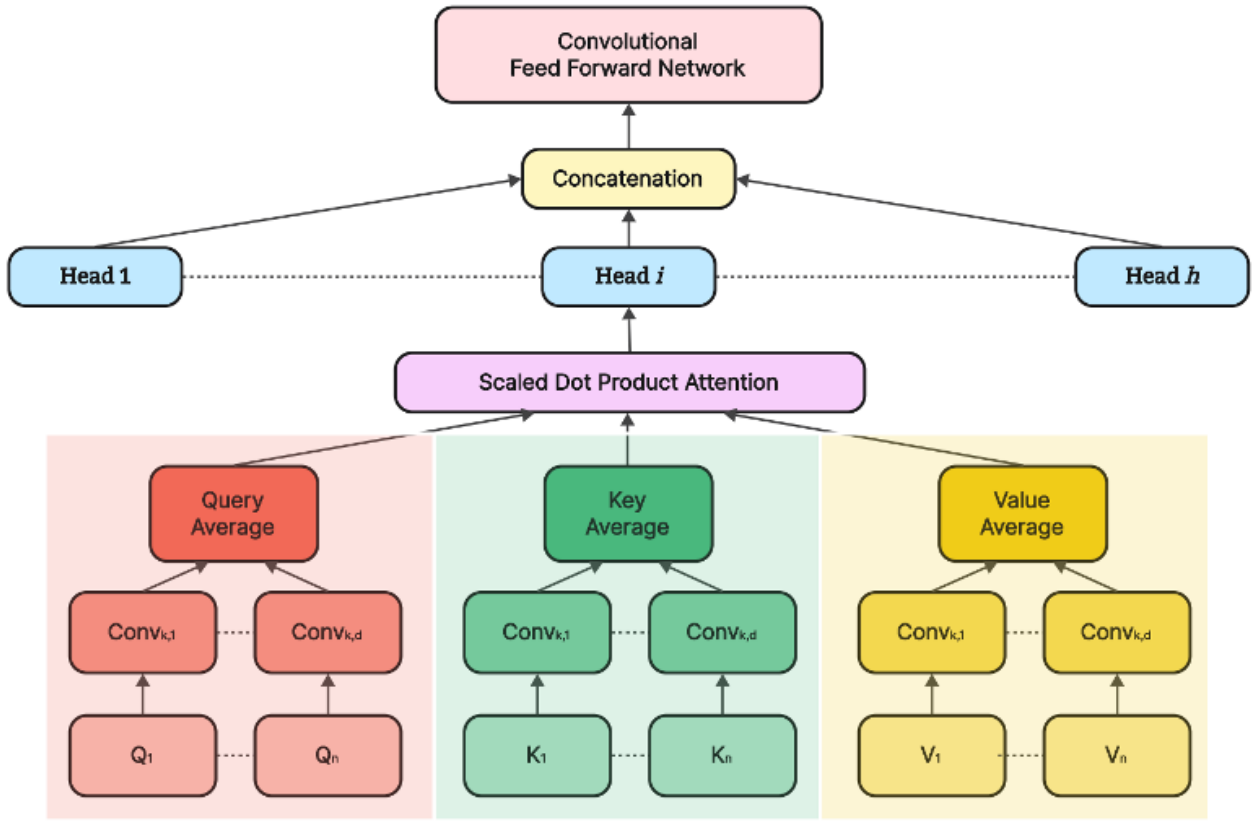

Figure 3: Graphical representation of the ConvFormer process

Takeaway

Through this survey project, I developed a system-level understanding of attention as a design space rather than a fixed module. Analyzing multi-head attention variants deepened my appreciation for the trade-offs between expressivity, efficiency, and redundancy in large models. This work strengthened my ability to read, compare, and synthesize complex research literature, and it directly informed my later research on modular and multimodal architectures, where architectural choices must be justified not only by performance but also by scalability and interpretability.

2 System and Application Development

2.1 Project 1: AIHire — Intelligent Recruitment Platform

Brief Introduction

Project repository available at `https://github.com/Russell12138/AIHire_platform`.

AIHire is a large-scale intelligent recruitment platform designed to support end-to-end hiring workflows in real-world organizational settings. The system addresses the limitations of traditional recruitment tools by integrating structured information management with AI-assisted interaction and decision support.

The platform serves three distinct user roles—job applicants, recruiters, and system administrators—each supported by a dedicated subsystem with role-specific functionality. Beyond conventional job posting and application management, AIHire incorporates intelligent components such as conversational assistance, automated resume analysis, and candidate–job matching, enabling more efficient and informed hiring processes. The project emphasizes system scalability, modularity, and real-world deployability rather than isolated algorithmic performance.

— Selected Excerpts —

Overall System Architecture. AIHire adopts a modular, service-oriented architecture that separates user-facing functionalities, business logic, and data management. The system is composed of three interconnected subsystems corresponding to applicants, recruiters, and administrators. Each subsystem exposes role-specific interfaces while sharing a unified backend infrastructure to ensure consistency, security, and scalability.

Core services include user authentication and authorization, job posting and application management, resume handling, and real-time communication. These services are designed to operate under concurrent multi-user access and to support extensibility for future intelligent modules.

Role-Based Subsystems and Workflows. The applicant subsystem supports profile creation, resume upload, job browsing, application submission, and AI-assisted interaction. Applicants can receive automated feedback and guidance through an intelligent chatbot, improving usability and engagement.

The recruiter subsystem enables job creation, applicant screening, interview coordination, and candidate evaluation. Recruiters can leverage AI-assisted resume parsing and candidate–job matching to reduce manual screening effort and to prioritize suitable candidates.

The administrator subsystem provides system-level oversight, including user management, data moderation, and platform monitoring. This separation of concerns ensures secure role-based access control while maintaining a coherent global workflow.

AI-Enhanced Functional Modules. In addition to the core platform logic, AIHire integrates multiple AI-driven modules implemented as independent services. These include an intelligent conversational agent for user assistance, automated resume parsing to extract structured candidate information, and a matching engine that aligns candidate profiles with job requirements.

These AI components communicate with the main system through well-defined service interfaces, allowing the platform to combine robust software engineering with intelligent decision support without tightly coupling AI logic to core infrastructure.

System Deployment and Engineering Practices. The system was deployed in a cloud environment and designed to support scalability and reliability under realistic usage conditions. Persistent data are managed through a relational database, while caching and concurrency control mechanisms are employed to ensure responsive interaction during peak access.

Throughout development, emphasis was placed on clear API specification, database schema design, and maintainable module boundaries, enabling the system to evolve beyond a coursework prototype into a deployable application architecture.

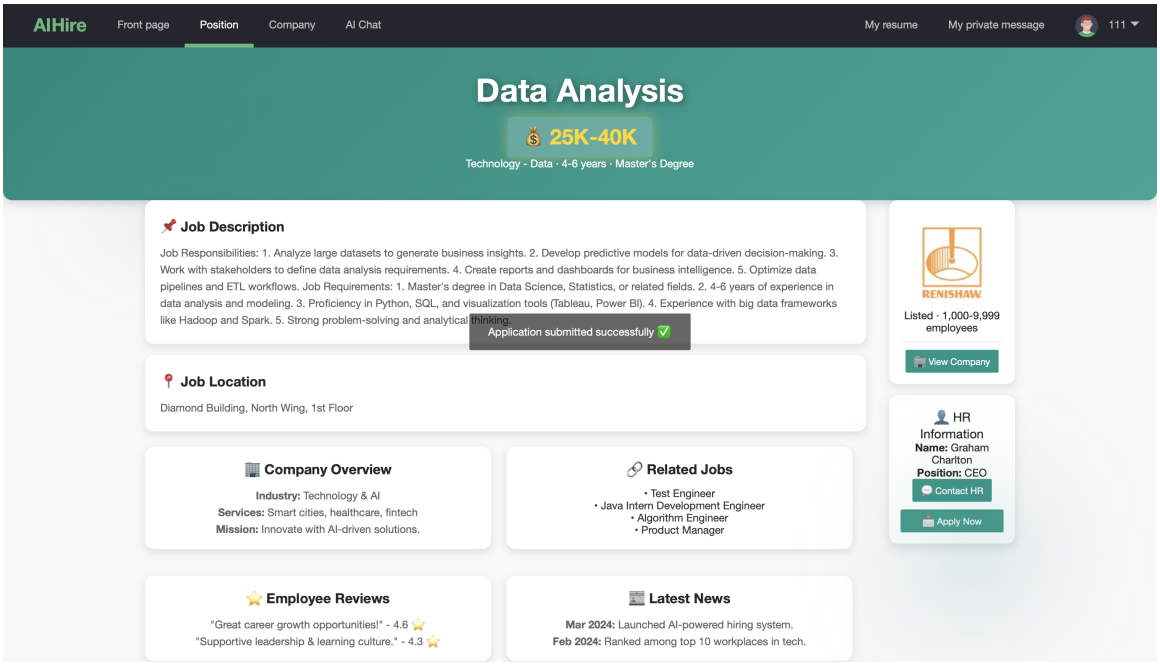

Figure 4: AIHire Position Detail Page

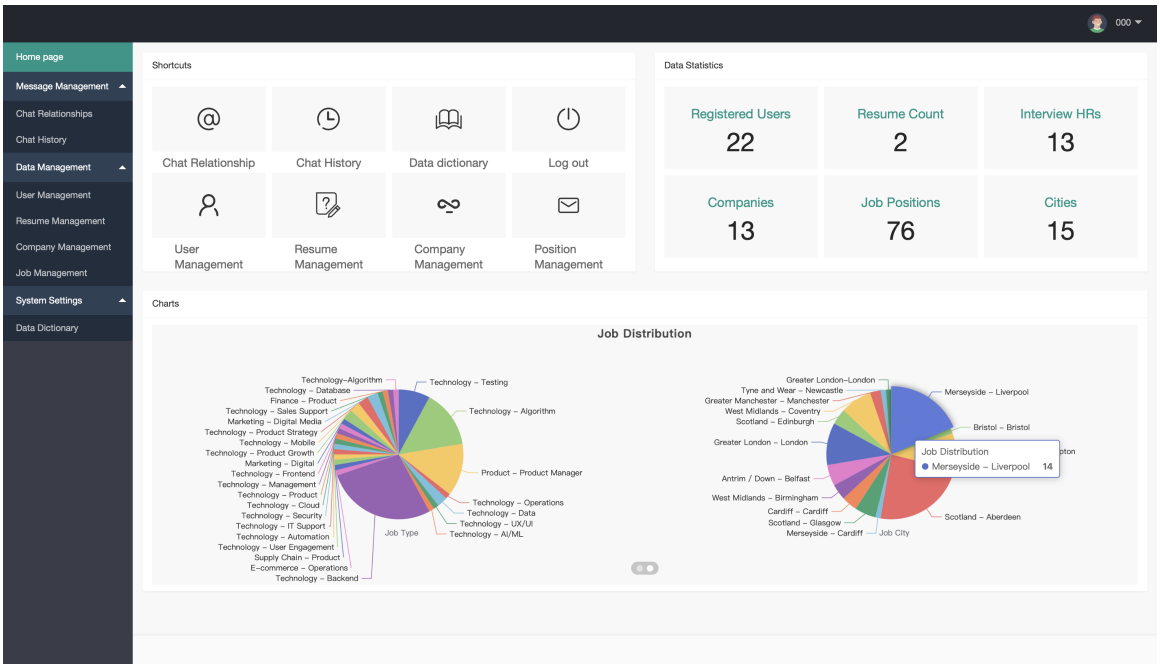

Figure 5: AIHire Administrator Management Page

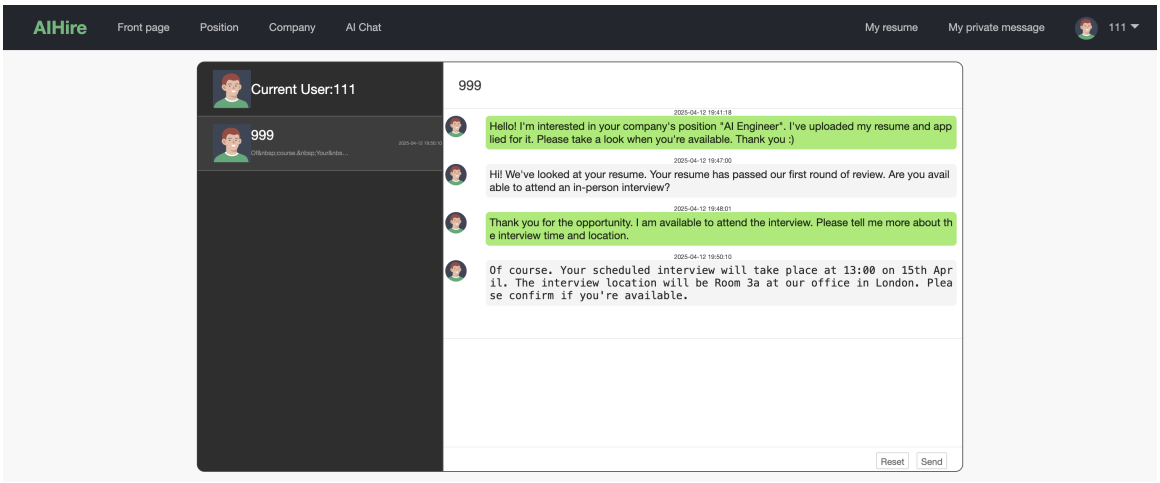

Figure 6: AIHire Real-time Chat Page

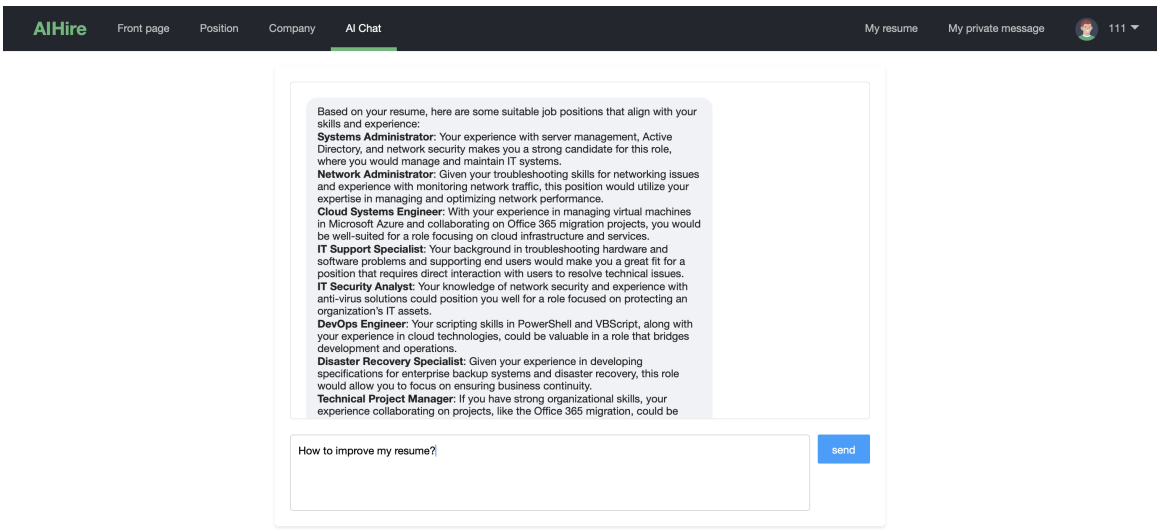

Figure 7: AIHire Intelligent Chatbot Page

Takeaway

AIHire represents my transition from isolated feature implementation to full-stack, system-level engineering. The project required coordinating multiple subsystems, user roles, and intelligent services within a unified architecture. Through this work, I gained practical experience in designing scalable software systems that integrate AI capabilities into real operational workflows, reinforcing my interest in building intelligent systems that function reliably in real-world environments.

2.2 Project 2: Gradio-based Interactive LLM Frontend

Brief Introduction

To support rapid prototyping and user-facing evaluation of large language models, I designed and implemented an interactive web-based frontend using *Gradio*. The goal of this project was to bridge backend LLM services and end users by providing a lightweight, extensible interface that supports real-time interaction, streaming responses, and prompt customization.

The frontend was designed to serve as a general-purpose interaction layer for multiple backend agents, enabling researchers and engineers to test conversational behaviors, prompt strategies, and system responses without modifying core backend logic.

— Selected Excerpts —

System Design and Interface Structure. The frontend was implemented using Gradio components to construct a modular UI supporting text input, system prompt configuration, and streaming model outputs. The interface allowed dynamic switching between different backend agents and prompt templates, enabling flexible experimentation with model behaviors.

Special attention was paid to usability and responsiveness. The system supported incremental token streaming to improve perceived latency and user experience, making long-form model responses easier to follow during interaction.

Backend Integration and Extensibility. The Gradio frontend communicated with backend LLM services through well-defined interfaces, allowing the frontend to remain decoupled from model-specific implementations. This design enabled rapid extension to new agents or APIs with minimal frontend modification.

The modular structure also allowed the frontend to be reused across multiple experimental settings during the internship, serving as a common interaction layer for different AI agent prototypes.

Takeaway

This project strengthened my ability to design user-facing AI systems that balance engineering robustness with experimental flexibility. By building a reusable interaction layer, I gained practical experience in connecting large language models to real users, emphasizing system modularity, usability, and rapid iteration in applied AI development.

2.3 Project 3: Real-Time Weather Chatbot with State-Aware Dialogue Management

Brief Introduction

In this project, I developed a real-time weather chatbot designed to provide accurate, context-aware weather information through multi-turn conversations. Unlike single-turn API-based chat systems, this chatbot needed to maintain dialogue coherence, track user intent, and dynamically integrate external weather data across multiple conversational turns.

The core challenge was to design an AI agent that could manage conversational state while remaining robust to incomplete or ambiguous user inputs in real-world usage scenarios.

— Selected Excerpts —

State Tracking and Dialogue Management. To support coherent multi-turn interaction, I designed a lightweight state-tracking mechanism based on structured JSON memory. The chatbot dynamically stored and updated key contextual variables such as location, time, and user query history, allowing the system to maintain continuity across dialogue turns.

This state representation was explicitly separated from the language model itself, enabling clear control over dialogue logic and reducing unintended context loss or hallucinated assumptions during conversation.

Real-Time API Integration. The chatbot integrated real-time weather data through external APIs, retrieving up-to-date forecasts and conditions based on the tracked dialogue state. The preserved context was injected into subsequent model prompts, ensuring that responses remained consistent with prior user intent without redundant clarification requests.

This design significantly reduced inappropriate context resets and improved user experience in extended interactions.

Evaluation and System Robustness. Through iterative testing, the system demonstrated substantial improvements in multi-turn coherence and reliability compared to stateless baselines. Error cases caused by missing or overwritten context were reduced by approximately 70%, highlighting the effectiveness of explicit state management in AI agents.

Takeaway

This project deepened my understanding of AI agent design beyond model invocation, emphasizing the importance of state, memory, and system-level control in real-world applications. It reinforced my interest in building intelligent systems that combine language models with structured reasoning and external knowledge sources to deliver reliable, context-aware behavior.

2.4 Project 4: Treasure Hunting Web Game Development

Brief Introduction

I developed a browser-based treasure hunting game using JavaScript, HTML, and CSS, implementing both the core game logic and the front-end presentation. The game operates on a 10×10 grid, where the player controls a hunter character that moves in four directions to collect treasures with varying values, while avoiding obstacles and an autonomous monster.

The project emphasizes interactive system design, real-time state management, and rule-based agent behavior in a client-side web environment. It integrates user-driven actions with computer-controlled logic, requiring careful coordination between game state transitions, rendering updates, and event handling.

Implementation and System Design

I designed and implemented the full game state management system, including player movement, collision detection, score accumulation, obstacle handling, and win/loss conditions. The hunter's movement is controlled through keyboard event listeners, while the game engine continuously updates the grid state and checks interaction constraints at each step.

A key component of the project is the monster AI logic. I implemented the monster as an autonomous agent with predefined movement rules, allowing it to navigate the grid independently and interact dynamically with the hunter. The game terminates either when all treasures are collected—triggering a victory state—or when the hunter is caught by the monster, resulting in a loss.

Beyond core logic, I was responsible for front-end design and visual styling. I integrated graphical assets for characters and objects, implemented layout and styling through CSS, and ensured smooth visual updates during gameplay. JavaScript was used to synchronize UI rendering with underlying game state changes in real time.

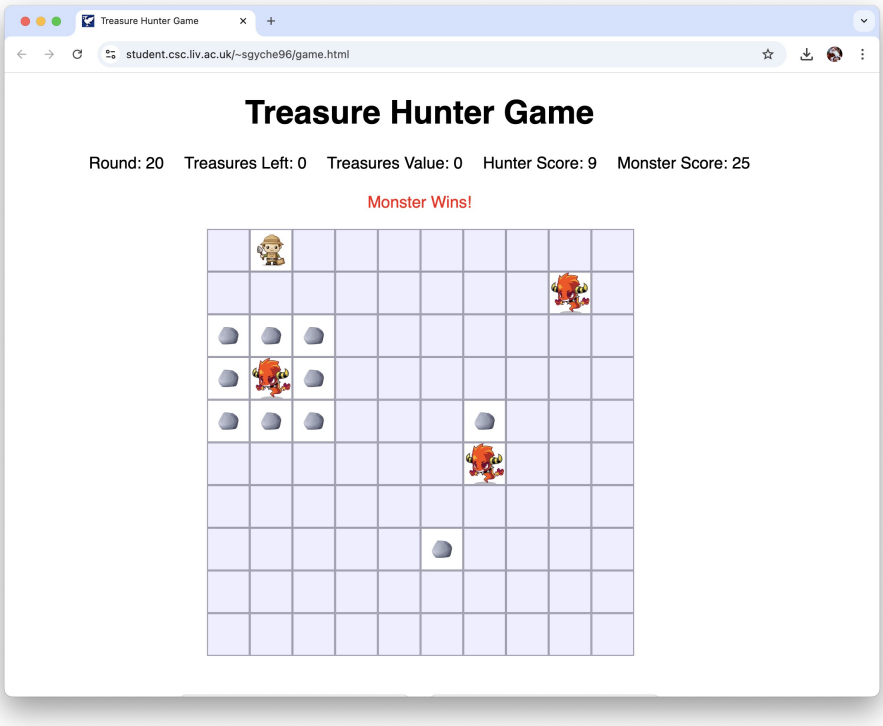

Figure 8: Treasure Hunting Web Game Interface and Gameplay

2.5 Project: Interactive Sankey Diagram Development in Java

Brief Introduction

This project involved the design and implementation of an interactive Sankey diagram visualization system using Java and JavaFX. The objective was to convert structured numerical data into a flow-based visual representation, where the width of each flow accurately reflects proportional relationships between a source entity and multiple target entities.

Instead of relying on external visualization libraries, I implemented the entire rendering pipeline from scratch, focusing on object-oriented design, graphical layout algorithms, and responsive user interaction. The resulting system supports dynamic input files, adaptive scaling, and interactive visualization, demonstrating both algorithmic reasoning and practical software engineering skills.

Implementation Overview

The system was implemented using JavaFX, with a modular object-oriented architecture separating data processing, layout computation, and rendering logic. Input data is read from structured text files and parsed into ordered key–value mappings, preserving semantic relationships and enabling deterministic layout generation.

To construct the Sankey diagram, I designed an algorithm that computes the height of each flow and target block proportionally based on its numeric value relative to the total. Smooth Bezier curves were used to render the flows between source and target nodes, ensuring visual continuity and readability. Each flow is color-coded consistently with its corresponding target block to enhance interpretability.

The visualization supports dynamic resizing by binding canvas dimensions to the application window and recalculating layout parameters in response to size changes. This ensures that the diagram remains legible and proportionally accurate across different screen sizes and data scales. User interaction and error handling were also carefully designed, including file validation, numeric format checking, and informative alert messages.

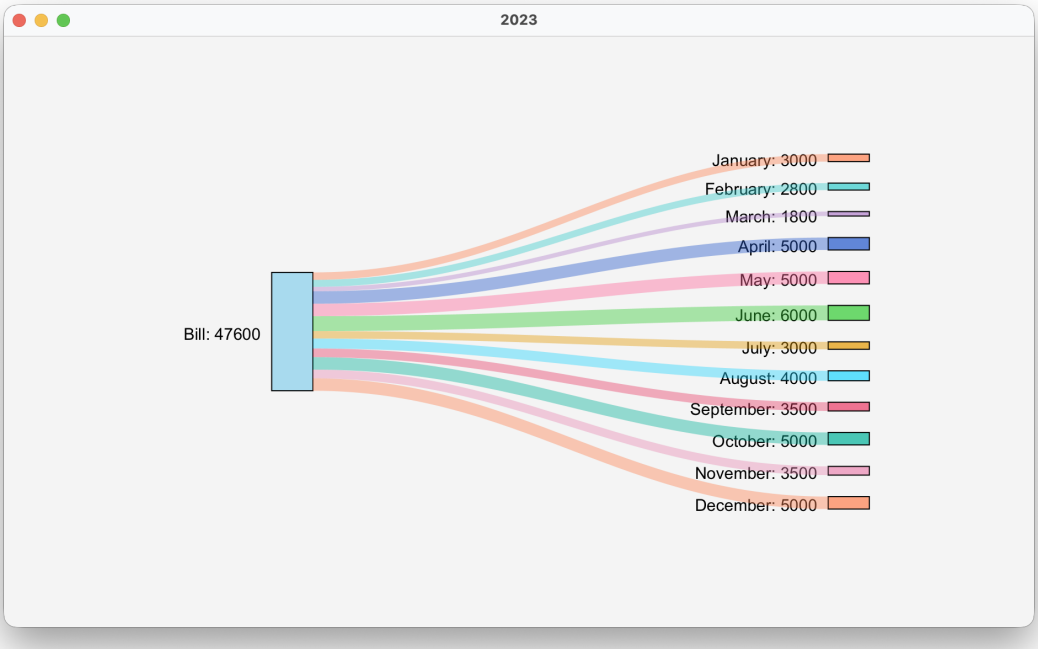

Figure 9: Sankey Diagram Display

Takeaway

3 Algorithms and Foundations

3.1 Project 1: Simulation and Analysis of Leader-Election Algorithms in Distributed Systems

Brief Introduction

Leader election is a fundamental coordination problem in distributed systems, requiring a set of autonomous nodes to agree on a unique leader without centralized control. In this project, I implemented and empirically evaluated two classical leader election algorithms—LeLann–Chang–Roberts (LCR) and Hirschberg–Sinclair (HS)—under a simulated ring network setting.

Beyond faithfully reproducing the original algorithms, I designed optimized variants of both LCR and HS to reduce communication overhead. The project emphasizes algorithmic correctness, communication complexity, and scalability under different network sizes and ID assignment strategies. Through systematic simulation and quantitative analysis, this work bridges theoretical distributed algorithms with practical performance evaluation.

— Selected Excerpts —

Leader Election Problem and Objectives. In distributed systems, Leader Election Problem is one of the crucial challenges, aiming to elect exactly one leader within multiple independent nodes. Various algorithms have been proposed to efficiently address this problem.

This report analyzes and compares two classical leader election algorithms, LCR (LeLann, Chang, and Roberts) and HS (Hirschberg-Sinclair), along with their potential optimized versions. Through experimental evaluation, we verify the correctness and measure the performance of each algorithm in terms of time complexity and communication complexity.

Algorithm Description: LCR and HS. LCR algorithm operates on directed (unidirectional) ring, where each node holds a unique ID. Initially each node sends its own ID to the neighbor, and upon receiving an ID, nodes will only forward it if it is larger than its own. Only if the node receives its own ID would it declare to be the leader, and broadcast the leader ID to every other node.

HS algorithm is designed for undirected (bidirectional) ring, and proceeds in synchronous phases. In each phase, each node tries to forward its ID up to the distance of 2^k to the left and right. Only larger ID will be forwarded otherwise discarded. The algorithm requires $O(\log n)$ rounds, and $O(n \log n)$ message transmissions to elect the leader.

Simulation Framework and Synchronization. To simulate the algorithms, two main classes, Node and Network, are created to implement a bidirectional ring structure. Each node can specify its right and left neighbor, thus the network could work as a unidirectional ring for the LCR algorithm as well.

Due to the nature of the Java programming language, the code cannot fully replicate the strict synchronization of a distributed system. To better simulate this process, message transmission and message processing are handled separately within each round.

Experimental Design and Evaluation. Verifying correctness involves comparing the results of the simulated execution with the theoretical expectations. The expectation is to retrieve correct leader ID, correct number of rounds, and correct number of message transmissions from the simulation.

Different network sizes and ID distribution strategies including random, clockwise, and counterclockwise were tested to ensure correctness and to identify performance patterns across algorithms and configurations.

Results and Analysis. Both algorithms were executed on networks of various sizes and different ID distribution strategies, with results recorded accordingly. At the end of the algorithm, all nodes successfully held the correct leader ID, and both the number of rounds and messages matched theoretical expectations.

The experimental results confirm that HS generally outperforms LCR in both round count and message count, particularly as network size increases, while optimized variants significantly reduce communication complexity under specific ID assignments.

Takeaway

This project strengthened my understanding of distributed coordination beyond theoretical proofs by grounding algorithm analysis in concrete system simulation. By implementing, optimizing, and benchmarking classical leader-election algorithms, I developed practical insight into how algorithmic design choices translate into communication cost, scalability, and robustness in distributed environments.

3.2 Project 2: Data Observation and Dimensionality Reduction for Programme Classification

Brief Introduction

This project investigates how numerical features extracted from real-world assessment data can be analyzed and visualized to reveal latent structure across different student programmes. Using exam data collected from the INT104 final examination, the goal was to explore feature distributions, identify informative dimensions, and evaluate the effectiveness of classical dimensionality reduction techniques for classification-oriented visualization.

Rather than directly training a predictive model, this work emphasizes data understanding and methodological reasoning. I systematically applied data observation, normalization, and multiple dimensionality reduction techniques—including PCA, t-SNE, and LDA—to study how programme-level separation emerges under different assumptions. The project highlights both the strengths and limitations of linear and nonlinear methods when applied to noisy, overlapping educational data.

— Selected Excerpts —

Data Observation and Preprocessing. To prepare for the latter dimensionality reduction, several necessary data visualizations are performed as follows. First of all, the feature 'Index' has been removed from the raw data, since it does not represent any meaningful information. The resulting data can be named as 'numerical data'. If generating Box Plot on the numerical data directly, the poor results are predictable, since the data from different features is not in the similar range of values. Several solutions are performed to address this issue, including performing Log Scale, Z-score Normalization and Min-Max Score on the numerical data.

For the outliers captured in the figure, not all of them represent mistakes in the data set. Taking 'Q1' for instance, when majority of the scores has reached 4 or even higher, 0 and 2 score will be considered as outliers. However, such data is actual collections from students from last year, and should not be removed entirely.

Principal Component Analysis (PCA). After removing 'Index' and 'Programme' from the raw data and performing Z-score Normalization, PCA is applied to the standardized data. By calculating the correlations between 'Programme' and each principal component, 'PC0' and 'PC1' are found to be mostly correlated to 'Programme'. The resulting scatter plot demonstrates that 'Programme3' is largely separated from other programmes, which remain mixed.

Nonlinear Dimensionality Reduction with t-SNE. Another way to visualize high-dimensional data by dimensionality reduction is t-SNE. Correlation analysis is used to select informative features, including 'Total', 'MCQ', 'Q2', 'Q4', and 'Grade'. While several configurations fail to separate programmes, adding 'Grade' as an input feature enables successful extraction of 'Programme3', confirming its discriminative role.

Analysis of Feature Contribution. Comparing 'Grade' with 'Programme', it is implied that students who belong to 'Programme3' have all got 'Grade3'. However, not all students with 'Grade3' belong to 'Programme3', explaining why partial overlap remains. Subsequent experiments removing 'Grade' further confirm its importance for programme-level separation.

3.3 Project 3: Supervised Classification and Ensemble Learning for Programme Identification

Brief Introduction

This project extends a prior data observation study by systematically applying supervised machine learning models to classify students from different academic programmes. Using the same real-world exam dataset, I explored how classical classifiers perform under varying preprocessing strategies, feature selections, and optimization techniques.

Rather than focusing on a single model, this work emphasizes comparative evaluation across diverse paradigms, including tree-based methods, support vector machines, probabilistic classifiers, and ensemble learning. By analyzing accuracy, confusion matrices, and failure cases, the project investigates not only which models perform best, but why certain programme distinctions are fundamentally difficult to resolve given the data distribution.

— Selected Excerpts —

Data Pre-processing. Before training any model, the raw data should be pre-processed as follows: Remove 'Index' and 'Programme' from raw data as DataFrame X and let DataFrame Y contain a single column 'Programme'. The reason we remove 'Index' is that the indexed numbers do not contain any practical meaning, and might lead to overfitting. Separate data into 80% training set and 20% test set. Within the training set, K-fold cross validation can be performed to obtain the optimal parameter combination. Perform Z-score normalization on the data for Support Vector Machine (SVM) and Naive Bayes (NB), and perform one-hot encoding on the 'Gender' column.

Decision Tree and Random Forest. To determine the optimal parameter combination for Decision Tree classifier, grid search and 5-fold cross-validation are performed. The trained model achieves an accuracy of 0.65 on the test set. Before performing Random Forest, Bayesian optimization is introduced to search for optimal parameter combinations efficiently. This RF model achieved an accuracy of 0.69 on the test set. Another attempt using post-pruning slightly decreases the accuracy, suggesting loss of valuable information.

Support Vector Machine. For linear SVM, Bayesian optimization selected a relatively weak regularization parameter, indicating enhanced generalization capability. The final model achieves an accuracy of 0.70. For kernel SVM, Bayesian optimization selects the sigmoid kernel with a large regularization parameter, but the accuracy slightly decreases compared to linear SVM.

Naive Bayes. Polynomial and Bernoulli Naive Bayes are excluded due to mismatch with data characteristics. Gaussian Naive Bayes is adopted and achieves an accuracy of 0.67. Feature histograms reveal approximate but imperfect Gaussian distributions, explaining the model's limitations.

Ensemble Learning. Predictions from multiple classifiers are integrated to form ensemble models. An ensemble composed of the three best-performing models achieves an accuracy of 0.70, matching the best individual model. Expanding the ensemble to include additional models does not yield further improvement.

3.4 Project 4: Unsupervised Clustering Analysis of Academic Programme Structure

Brief Introduction

This project investigates the relationship between unsupervised clustering structure and known academic programme labels using real-world exam data. Unlike supervised classification, the goal is not to optimize predictive accuracy directly, but to analyze whether intrinsic data geometry aligns with the programme categories, and under what conditions clustering metrics meaningfully reflect semantic separability.

I systematically evaluated three classical clustering paradigms—Gaussian Mixture Models (GMM), K-means, and Hierarchical Clustering—under different dimensionalities, feature subsets, and cluster counts. By jointly analyzing Silhouette Scores and post-hoc classification accuracy, this work reveals a non-trivial and sometimes contradictory relationship between clustering quality and semantic alignment, highlighting the limitations of unsupervised structure discovery when class labels do not correspond to natural clusters.

— Selected Excerpts —

Data Pre-processing and Evaluation. The data is split into a 70% training set and a 30% test set. Initially, the 'Index' and 'Programme' columns were removed, retaining all other columns. To enhance visualization and evaluation, PCA was used for dimensionality reduction. The top two principal components explained almost 95% of the variance.

To evaluate clustering performance, the Silhouette Coefficient was used to represent intra-cluster cohesion and inter-cluster separation. For assessing the 'Programme' representation capability, each cluster was assigned the label of the most frequent 'Programme' within it, and overall accuracy was calculated.

Cluster Number Determination. Determining the optimal number of clusters requires considering both clustering performance and accuracy. According to the elbow method, the curve levels off after the number of clusters reaches 4 and 7, suggesting these as optimal candidates.

GMM Clustering. GMM clustering was performed on both 2D and 3D datasets. The 2D GMM achieved a Silhouette score of 0.14 and accuracy of 0.49, while the 3D GMM achieved a Silhouette score of 0.26 and accuracy of 0.49. The improvement is likely due to increased inter-point distances in higher-dimensional space.

K-means Clustering. K-means clustering shows higher Silhouette scores than GMM, but the resulting cluster-to-programme mapping sacrifices certain programmes entirely due to majority-label assignment. The 2D K-means achieved a Silhouette score of 0.40 and accuracy of 0.51, while the 3D version achieved an accuracy of 0.61.

Hierarchical Clustering. Agglomerative hierarchical clustering using Ward's method produces compact clusters but exhibits similar trade-offs. The 3D clustering preserves more information for 'Programme 3', suggesting that additional PCA dimensions encode semantically meaningful variance not visible in 2D projections.

Key Observations. GMM clustering results show a positive correlation between Silhouette Score and accuracy, while K-means and hierarchical clustering show a negative correlation. This indicates that optimizing clustering cohesion does not necessarily improve alignment with semantic labels when the label itself does not reflect intrinsic cluster structure.