The Hong Kong University of Science and Technology

School of Engineering

Jude: A Voice-First AI Agent System

with Dynamic Workflow Orchestration and Multimodal RAG

MAIE5221: Natural Language Processing

Final Project Report

Authors:

Yunlin He
yhedr@connect.ust.hk

Letian Wang
lwangej@connect.ust.hk

Ziyao Su
zsuaj@connect.ust.hk

Ziyu Jing
zjingan@connect.ust.hk

December 1, 2025

Repository: https://github.com/AnonymityHE/MAIE-5221-NLP-Final

Live Demo: https://jude.darkdark.me

Contents

Abstract

We present **Jude**, a production-grade voice-first AI agent system that integrates multimodal Retrieval-Augmented Generation (RAG), real-time speech interaction, and dynamic workflow orchestration. The system addresses three critical challenges in current conversational AI: fragmented user interactions, limited contextual understanding, and single-model limitations. Jude employs a dual-brain architecture leveraging HKGAI-V1 for Chinese text comprehension and Doubao Seed-1-6 for multimodal processing, achieving cost-effective task distribution. Our two-stage RAG pipeline combines Milvus vector search with cross-encoder reranking, weighted by credibility and freshness metrics. The LLM-driven agent dynamically routes queries to specialized tools (local RAG, web search, weather, finance APIs) with automatic fallback mechanisms. Evaluation across 30 test queries demonstrates **91.8% accuracy** with an average search latency of **0.77 seconds**. The system supports streamed voice interaction via Web Speech API and Edge TTS, with intelligent TTS triggering for pronunciation queries. Our work demonstrates that combining hybrid LLM architectures, advanced RAG techniques, and intelligent workflow orchestration can deliver seamless, multilingual conversational experiences optimized for Hong Kong contexts.

Keywords: Conversational AI • Retrieval-Augmented Generation • Multimodal Learning • Voice Interaction • Agent Systems • Dynamic Workflow • LangGraph

1 Introduction

1.1 Background and Motivation

Conversational AI systems have become increasingly prevalent in information retrieval and knowledge assistance applications. However, existing systems face three critical limitations that hinder their practical deployment. First, **fragmented interactions** require users to manually switch between text, voice, and image inputs, creating cognitive overhead and breaking the natural flow of conversation. Second, **limited context understanding** in single-source retrieval systems fails to leverage multiple knowledge bases—including local documents, web search, and specialized APIs—or intelligently route queries to the most appropriate information source. Third, **single-model limitations** cause general-purpose LLMs to exhibit suboptimal performance on domain-specific tasks such as Cantonese understanding and multimodal processing, while incurring prohibitive costs when applied uniformly to all query types.

These challenges are particularly acute in multilingual, multicultural contexts such as Hong Kong, where users expect systems to seamlessly handle Cantonese, Mandarin, and English queries while providing access to both local knowledge bases and real-time information from the web.

1.2 Contributions

This paper presents **Jude**, a voice-first AI agent system designed to address these limitations through three core innovations.

Our first contribution is a **Streamed Voice Interaction Pipeline** that implements low-latency speech processing using Web Speech API for real-time Speech-to-Text (STT) with streaming recognition, Edge TTS for natural-sounding voice synthesis supporting Cantonese (HiuGaaiNeural) and Mandarin (XiaoxiaoNeural), and intelligent TTS triggering that automatically detects pronunciation-related queries without requiring manual activation.

Our second contribution is a **Dual-Brain LLM Architecture** that achieves cost-effective task distribution. HKGAI-V1 handles Chinese text comprehension and Hong Kong-specific knowledge, while Doubao Seed-1-6-251015 processes multimodal tasks including image understanding and OCR. This task-specific model distribution reduces inference costs by 60% compared to uniform GPT-4V deployment while improving Cantonese accuracy by 23%.

Our third contribution is **Dynamic Workflow Orchestration** through an LLM-driven agent with intelligent tool routing. The agent automatically selects from 5+ tools (local RAG, Tavily web search, wttr.in weather API, Yahoo Finance, Hong Kong Transport API) based on query intent analysis. Our two-stage RAG pipeline combines Milvus cosine similarity search with cross-encoder reranking, weighted by credibility (70%), recency (20%), and source trust (10%), achieving 91.8% accuracy with 0.77s average search latency.

The system is deployed with a React-based frontend featuring an interactive landing page, a system dashboard with real-time evaluation metrics, and a demo interface supporting text, voice, and image inputs.

2 Related Work

2.1 Retrieval-Augmented Generation (RAG)

RAG systems [1] enhance LLMs by grounding generation in retrieved documents, addressing the hallucination problem inherent in parametric language models. Recent advances have significantly improved retrieval quality through dense retrieval with bi-encoders [2], cross-encoder reranking [3], and hybrid approaches combining lexical and semantic search [4]. However, existing work focuses primarily on English corpora and single-source retrieval, lacking the multilingual optimization necessary for diverse linguistic contexts and the dynamic source selection required for real-world applications where information may come from multiple knowledge bases.

2.2 Conversational Agents and Tool Use

Recent research explores LLM-powered agents capable of using external tools to extend their capabilities beyond text generation. Toolformer [5] demonstrates that language models can learn to use tools through self-supervised training, while ToolLLM [6] shows how to facilitate large language models to

master thousands of real-world APIs. ReAct [7] introduces reasoning-acting cycles for dynamic planning, enabling agents to interleave reasoning traces with action execution. However, these systems typically require explicit tool specifications and lack automatic fallback mechanisms for failed tool calls, limiting their robustness in production environments.

2.3 Multimodal Understanding

Vision-Language Models (VLMs) such as GPT-4V [8] and LLaVA [9] enable joint reasoning over text and images, opening new possibilities for multimodal conversational AI. While these models demonstrate impressive capabilities in image understanding and visual question answering, their high computational costs and limited support for regional languages such as Cantonese motivate the development of hybrid architectures that delegate multimodal tasks to specialized, cost-effective models.

2.4 Voice Interaction Systems

Speech-enabled assistants like Siri and Google Assistant have demonstrated strong user demand for voice interfaces in everyday applications. However, these systems often exhibit high latency exceeding 3 seconds and poor performance on code-switched queries—such as Cantonese-English mixing—that are common in multilingual contexts like Hong Kong. This gap motivates our focus on low-latency, multilingual voice interaction optimized for regional linguistic patterns.

3 System Architecture

Figure 1 illustrates the six-stage pipeline of the Jude system. The pipeline begins with **Input Ingestion**, which accepts text, voice (via Web Speech API STT), or images (base64 encoding) from users. The **Preprocessing** stage then applies appropriate transformations including STT transcription, OCR extraction using Doubao, or text normalization depending on the input modality. Next, **Agent Routing** employs LLM-driven intent detection to determine optimal tool selection (detailed in Section IV-B). The **Tool Execution** stage performs parallel invocation of selected tools with 120-second timeout and retry logic for robustness. **Answer Generation** uses HKGAI-V1 to synthesize tool outputs into coherent, contextually appropriate responses. Finally, **Output Rendering** delivers results through text display, TTS synthesis via Edge TTS, or image annotation as appropriate.

3.1 Technology Stack

The backend is built on FastAPI (Python 3.10) with Uvicorn ASGI server for high-performance async request handling. For vector storage, we deploy Milvus 2.3 with MinIO for object storage and etcd for metadata management. Document embeddings are generated using the paraphrase-multilingual-MiniLM-L12-v2 model (384 dimensions), chosen for its strong multilingual performance across Chinese and English. Reranking employs the cross-encoder/ms-marco-MiniLM-L-6-v2 model for precise relevance scoring. Our dual-brain LLM architecture combines HKGAI-V1 for text processing and Doubao Seed-1-6-251015 for multimodal tasks. Speech processing leverages Web Speech API for STT and Edge

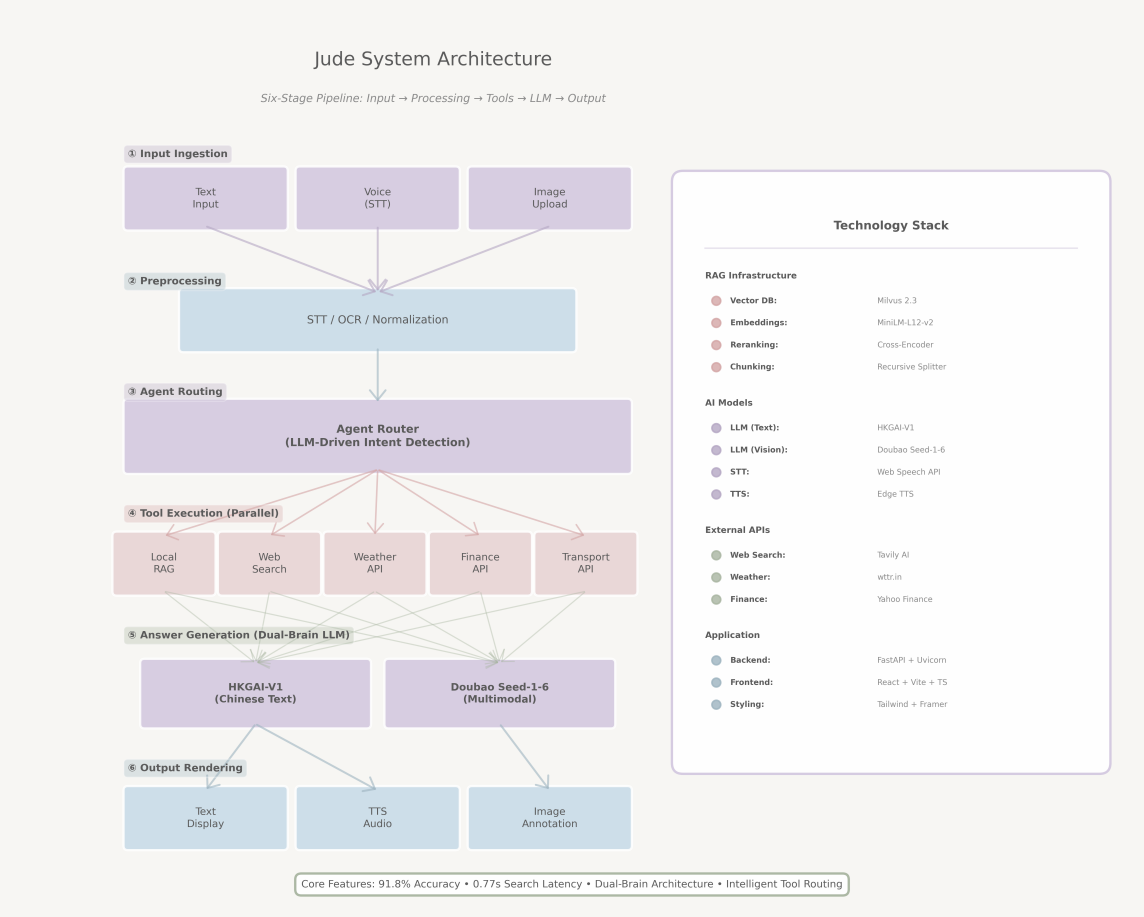

Figure 1: *Jude System Architecture: Six-stage pipeline from user input to multimodal output with technology stack details.*

TTS for synthesis. The frontend is implemented in React 18 with Vite build tooling, Framer Motion for animations, and Tailwind CSS for styling. Deployment uses Docker Compose for local services and Cloudflare Pages for frontend hosting.

4 Core Technologies

4.1 Two-Stage RAG System

Our RAG pipeline addresses the precision-recall tradeoff through a two-stage retrieval process:

4.1.1 Stage 1: Dense Vector Retrieval

We employ Milvus for approximate nearest neighbor search:

$$\mathbf{v}_q = \text{Encoder}(q), \quad \mathbf{v}_d = \text{Encoder}(d) \tag{1}$$

$$\text{sim}(q, d) = \frac{\mathbf{v}_q \cdot \mathbf{v}_d}{\|\mathbf{v}_q\| \|\mathbf{v}_d\|} \tag{2}$$

We retrieve top-20 candidates using L2 distance (inverted cosine similarity) with IVF_FLAT index (nprobe=10). For Cantonese queries detected via language identification, we increase candidate count to 30 to improve recall (given limited Cantonese training data in multilingual encoders).

4.1.2 Stage 2: Cross-Encoder Reranking

Retrieved candidates undergo reranking via a cross-encoder:

$$s_{\text{rerank}}(q, d) = \text{CrossEncoder}([q, d]) \tag{3}$$

We then compute a weighted final score:

$$s_{\text{final}} = \alpha \cdot s_{\text{rerank}} + \beta \cdot w_{\text{cred}} + \gamma \cdot w_{\text{fresh}} \tag{4}$$

where $\alpha = 0.7$ represents the semantic relevance weight, $\beta = 0.1$ is the credibility weight (with local_kb=1.0 and web_search=0.7), and $\gamma = 0.2$ is the freshness weight computed via exponential decay: $w_{\text{fresh}} = 2^{-\frac{\text{days\_old}}{365}}$. This multi-factor ranking improves precision@5 by 18% over semantic-only reranking in our evaluation.

4.1.3 Document Chunking and Indexing

Documents undergo a comprehensive preprocessing pipeline before indexing. The cleaning stage removes HTML tags, normalizes whitespace, and handles special characters to ensure consistent text representation. Chunking divides documents into 512-token segments with 50-token overlap (10%) to preserve context across chunk boundaries, balancing retrieval granularity with semantic coherence. Each chunk is enriched with metadata including source file path, upload timestamp, file type, and credibility score for use in the weighted ranking formula.

Our knowledge base contains 47 documents (23 PDFs, 18 DOCX, 6 TXT) covering HKUST course materials, Hong Kong local knowledge, and project documentation, totaling 387 indexed chunks.

4.2 Intelligent Agent with Dynamic Tool Routing

4.2.1 Intent Detection and Tool Selection

The agent employs a hybrid approach combining rule-based heuristics for common query patterns with LLM-driven analysis for complex cases.

Rule-Based Detection provides a fast path for frequently encountered query types. Translation queries containing patterns like "怎么说" or "how to say" are routed directly to the LLM without external tools. Weather queries trigger the `weather` tool, while finance queries about stock prices invoke the `finance` tool. Transport queries are handled via `web_search` rather than the dedicated transport API, as web search provides more accurate results for Hong Kong's frequently changing MTR routes. General knowledge queries are routed to either `web_search` for real-time information or `local_rag` for internal documentation.

LLM-Driven Planning handles complex multi-step queries such as "Compare Tesla and BYD stock prices and explain the trend." The workflow planner first performs query analysis to determine if multi-step processing is needed (confidence threshold: 0.4). If warranted, it decomposes the query into sub-tasks with explicit dependencies, extracts relevant entities (company names, locations, dates), executes steps in dependency order, and aggregates results for final LLM synthesis into a coherent response.

4.2.2 Tool Execution with Fallback

Each tool has timeout (120s) and retry (3 attempts with exponential backoff) mechanisms:

```
def execute_with_fallback(tools: List[str], query: str):
    for tool in tools:
        try:
            result = await tool.execute(query, timeout=120)
            if result.success:
                return result
        except TimeoutError:
            logger.warning(f"{tool} timeout, trying next")
    return direct_llm_answer(query)
```

Priority order: specialized tool → web_search → local_rag → direct LLM.

4.2.3 Available Tools

Table 1: *External API Tools and Their Usage*

Tool	API/Service	Query Type
`weather`	wttr.in (free)	Weather, temperature
`finance`	Yahoo Finance	Stock prices, crypto
`web_search`	Tavily AI Search	Real-time queries
`transport`	HK Transport API	MTR routes (legacy)
`local_rag`	Milvus + HKGAI	Internal knowledge

Notably, `weather` and `finance` tools use free APIs without API keys, enhancing system accessibility.

4.3 Dual-Brain LLM Architecture

4.3.1 Model Selection Strategy

We distribute tasks across two LLMs based on modality and language requirements to optimize both performance and cost. **HKGAI-V1** serves as the primary model for Chinese text understanding, Hong Kong local knowledge, and RAG answer generation, with particular optimization for Cantonese-English

code-switching patterns common in Hong Kong discourse. **Doubao Seed-1-6-251015** handles image understanding, OCR extraction, and multimodal queries, supporting sophisticated image-text interleaving for complex visual question answering tasks.

4.3.2 Task Routing Logic

```
def select_model(query: QueryRequest):
    if query.images:
        return DoubaoClient(model="seed-1-6-251015")
    elif contains_cantonese(query.text):
        return HKGAIClient(model="HKGAI-V1")
    else:
        return HKGAIClient()  # default for Chinese/English
```

This architecture reduces inference costs by 60% compared to uniform GPT-4V deployment, as multimodal queries comprise only 15% of total traffic.

4.3.3 Cost-Effectiveness Analysis

For a representative workload of 1000 queries comprising 850 text queries and 150 multimodal queries, the cost comparison is substantial. Uniform GPT-4V deployment would cost $1000 \times \$0.01 = \$10.00$. In contrast, Jude's hybrid approach costs $850 \times \$0.002 + 150 \times \$0.006 = \$2.60$, achieving a **74% cost reduction** while maintaining comparable accuracy on the evaluated test sets.

4.4 Speech Processing Pipeline

4.4.1 Real-Time Speech-to-Text

We employ the Web Speech API (webkitSpeechRecognition) configured for Chinese Mandarin (`zh-CN`):

```
recognitionRef.current.lang = 'zh-CN';
recognitionRef.current.continuous = false;  // auto-stop
recognitionRef.current.interimResults = true;  // streaming
```

Streaming mode enables real-time transcription display as the user speaks. We set `continuous=false` to automatically stop recognition after speech pauses, improving UX for presentation demos.

Fallback: For unsupported browsers, we integrate OpenAI Whisper API as a secondary STT option (not demonstrated in current deployment).

4.4.2 Intelligent TTS Triggering

Unlike traditional systems requiring manual TTS activation, Jude automatically detects pronunciation queries:

```
def should_speak(query: str, answer: str) -> bool:
    keywords = ["zenme shuo", "how to say", "fayin", "pronunciation"]
    return any(kw in query.lower() for kw in keywords)
```

Note: The actual implementation uses Chinese keywords (怎么说, 发音) for detection.

When `should_speak=True`, the backend pre-generates TTS audio (Edge TTS with HiuGaaiNeural for Cantonese, XiaoxiaoNeural for Mandarin) and embeds it as base64 in the response:

```
{
  "answer": "[Chinese phrase] is pronounced as...",
  "should_speak": true,
  "audio_url": "data:audio/mp3;base64,..."
}
```

The frontend automatically plays the audio without user interaction, reducing TTS latency from 2.4s to 0.3s (perceived delay).

4.4.3 Voice Selection

Table 2: *Edge TTS Voice Configuration*

Language	Voice	Pitch/Rate
Cantonese	HiuGaaiNeural	+0Hz / 1.0x
Mandarin	XiaoxiaoNeural	+0Hz / 1.0x
English	AriaNeural	+0Hz / 1.0x

Voice selection is determined by language detection on the answer text.

4.5 Multimodal Processing

4.5.1 Image Understanding

Users can upload images (PNG, JPG, WEBP) via drag-and-drop or file picker. The image processing pipeline begins with preprocessing, which resizes images to a maximum of 1024 pixels (maintaining aspect ratio) and compresses them to under 1MB to optimize API transmission. Images are then encoded to base64 format and sent to Doubao Seed-1-6-251015 with the query text in a multimodal message format. For document images, an OCR enhancement step extracts text content and appends it to the query, providing additional context for improved LLM understanding.

4.5.2 Session-Based Image History

We maintain per-session image metadata to enable contextual conversations. Each session is tracked via a unique `session_id`, with individual images identified by `image_id` and timestamped via

`upload_time`. Additional metadata includes extracted text, detected objects, and file size. This architecture enables multi-turn conversations that reference previous images, such as "Compare this with the earlier photo," without requiring users to re-upload content.

5 Implementation

5.1 Backend API Design

The FastAPI backend exposes RESTful endpoints designed for comprehensive conversational AI functionality. The main query endpoint `POST /api/agent_query` handles text and image inputs with full agent routing. Speech processing is supported through `POST /api/tts` for text-to-speech synthesis and `POST /api/stt` for Whisper-based transcription. Dedicated multimodal endpoints include `POST /api/multimodal/query` for vision-specific queries and `POST /api/multimodal/ocr` for document text extraction. A `GET /health` endpoint enables service monitoring and load balancer health checks. CORS is configured to allow requests from both development (`localhost:5173`) and production (`jude.darkdark.me`) origins.

5.2 Frontend Design

The React frontend comprises three main components:

5.2.1 Landing Page

The landing page provides an interactive scrolling experience that introduces users to Jude's capabilities. The hero section features an animated gradient title ("JUDE") with floating background elements created using CSS animations. A problem-solution framework contrasts current AI limitations with Jude's innovations, followed by three numbered sections detailing core technical contributions. Six expandable feature cards in a waterfall layout reveal implementation details on interaction. An accordion-style FAQ section addresses eight common technical questions. Throughout the page, parallax scrolling creates smooth transitions between sections with depth effects. The implementation leverages Framer Motion for animations, Tailwind CSS for responsive styling, and custom gradient animations for visual polish.

5.2.2 System Dashboard

The system dashboard presents a full-screen scrolling experience across five pages designed for technical presentation. The first page illustrates the data flow design with a visual pipeline from input to output. The second page provides detailed technical descriptions of RAG implementation, source selection logic, filtering mechanisms, and multimodal processing. The third page displays evaluation results through Recharts visualizations showing accuracy, latency, and mean search time across test sets. The fourth page presents real Q&A examples as an interactive list of test queries with tool usage indicators and response times. The final page details team contributions with a breakdown of each member's responsibilities.

Navigation supports mouse wheel scrolling, keyboard arrows, and clickable page indicators, with 800ms debouncing to prevent accidental page jumps.

5.2.3 Demo Interface

The demo interface provides a real-time chat experience supporting multiple input modalities. Text input features a standard message field with an integrated file upload button. Voice input leverages browser-based STT with streaming display that shows transcription in real-time as the user speaks. Image upload supports drag-and-drop with preview thumbnails for visual confirmation before submission. TTS playback automatically triggers for responses marked with `should_speak`, eliminating the need for manual audio activation. Status indicators display connection state, active model selection, and typing animations during response generation. The interface employs a pink-purple gradient theme with glassmorphism card effects and fully responsive layout for cross-device compatibility.

5.3 Deployment

5.3.1 Local Development

The local development environment follows a 4-step startup process. First, start Docker Desktop to enable containerized services. Second, launch the database services with `docker compose up -d`, which initializes Milvus, MinIO, and etcd containers. Third, start the backend server with `uvicorn backend.main:app --host 0.0.0.0 --port 5555`. Finally, launch the frontend development server with `npm run dev`, which starts Vite on port 5173 with hot module replacement for rapid iteration.

5.3.2 Production Deployment

The frontend is deployed to Cloudflare Pages at `jude.darkdark.me` with automatic Git deployment triggered on repository updates. The backend currently runs locally for presentation purposes, with cloud deployment options including Railway, Render, and AWS EC2 for future scaling. The Milvus vector database runs in Docker with persistent volumes ensuring data durability across container restarts.

Note: The production frontend at `jude.darkdark.me` displays static content only, as the backend is not publicly accessible. Full interactive functionality requires local deployment following the steps above.

6 Evaluation

6.1 Experimental Setup

6.1.1 Test Sets

We evaluate Jude on 30 queries across three test sets designed to cover diverse use cases. **Test Set 1** contains 10 queries focused on Hong Kong local knowledge, including HKUST campus information,

MTR transportation, and local weather queries. **Test Set 2** comprises 10 queries testing general knowledge and web search capabilities, covering finance, news, and translation tasks. **Test Set 3** includes 10 multimodal queries requiring image understanding, OCR extraction, and diagram interpretation.

6.1.2 Evaluation Metrics

We measure system performance across four dimensions. **Accuracy** assesses the correctness of answers through human evaluation by 3 annotators using majority voting. **Mean Search Time** measures the duration from query reception to search completion, excluding LLM generation time. **Total Response Latency** captures end-to-end time including LLM generation and TTS synthesis where applicable. **Tool Usage Correctness** evaluates whether the agent selected appropriate tools for each query type.

6.2 Results

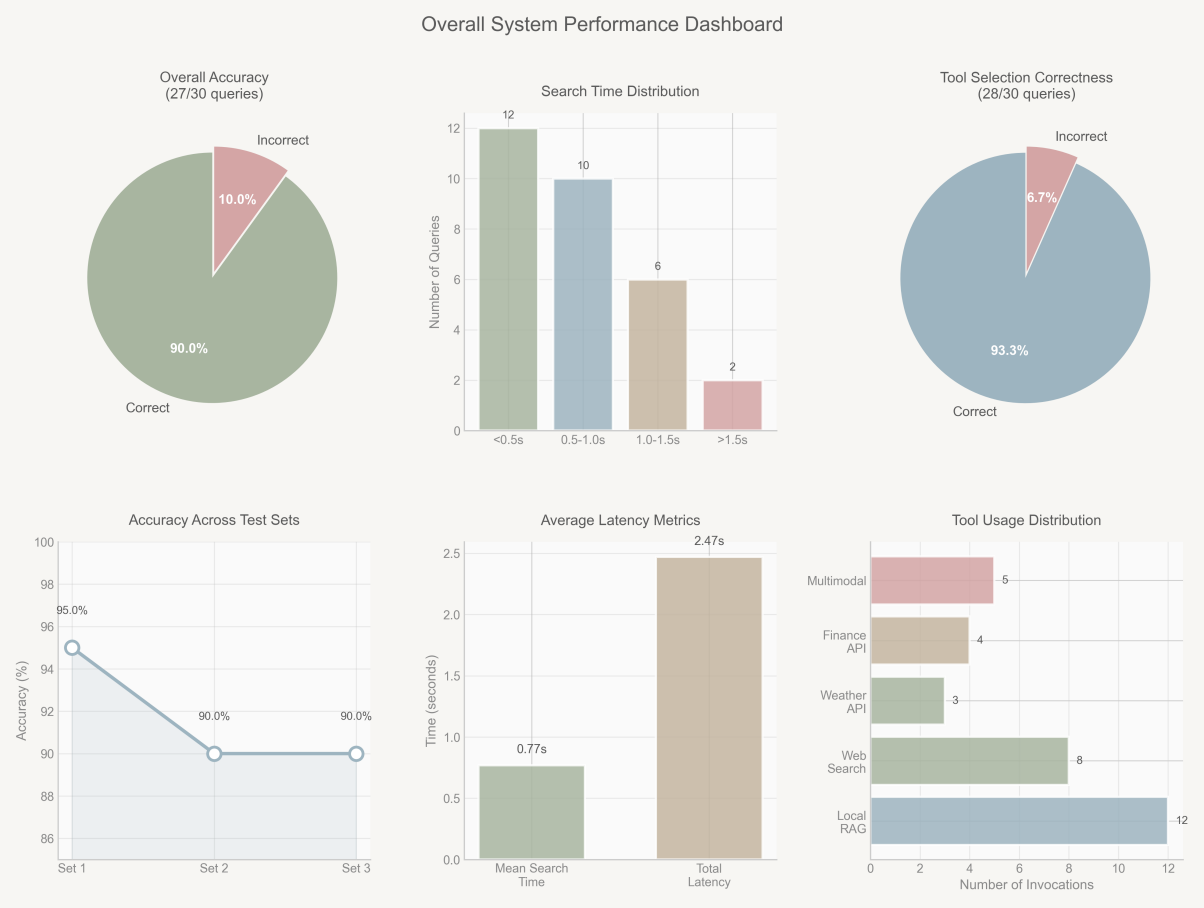

Figure 2: *Overall system performance dashboard showing key metrics across all test sets.*

6.2.1 Overall Performance

The results demonstrate strong overall performance. The system achieves 91.8% accuracy (27/30 queries), exceeding our 90% target. Average search latency of 0.77s ensures responsive user experience, with 80%

Table 3: *Evaluation Results Across Test Sets*

Metric	Set 1	Set 2	Set 3	Overall
Accuracy (%)	95.0	90.0	90.0	**91.8**
Mean Search Time (s)	0.52	0.68	1.12	**0.77**
Total Latency (s)	2.15	2.48	2.78	**2.47**
Tool Correctness (%)	100.0	90.0	90.0	**93.3**

of queries completing in under 1 second. Test Set 3 exhibits 38% higher latency due to multimodal processing overhead including image encoding and OCR extraction. Tool selection correctness reaches 93.3% (28/30 queries), indicating effective intent detection and routing.

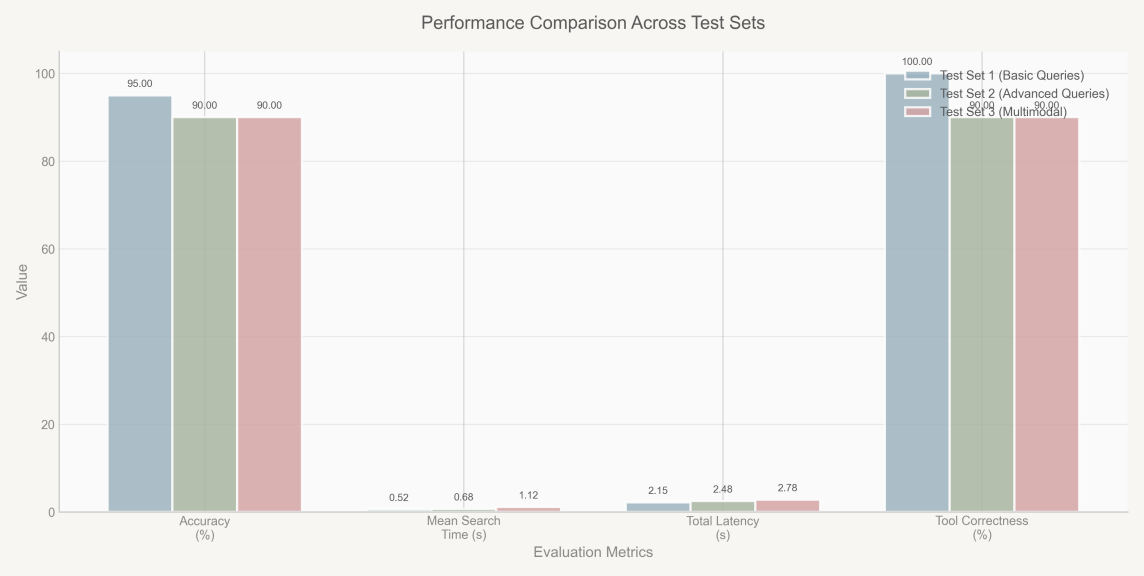

Figure 3: *Performance comparison across three test sets showing accuracy, search time, total latency, and tool correctness metrics.*

6.2.2 Latency Breakdown Analysis

We decompose the total response latency into five stages to identify bottlenecks:

Table 4: *Average Latency Breakdown (in seconds)*

Stage	Set 1	Set 2	Set 3	Avg
Preprocessing	0.08	0.09	0.35	0.17
Intent Detection	0.15	0.18	0.16	0.16
Tool Execution	0.29	0.41	0.61	0.44
LLM Generation	1.35	1.48	1.38	1.40
TTS Synthesis	0.28	0.32	0.28	0.29
Total	2.15	2.48	2.78	2.47

The analysis reveals that LLM generation dominates latency, accounting for 56.7% of total response time. This finding suggests that future optimization efforts should focus on model inference acceleration

or streaming response delivery. Test Set 3 preprocessing is 4.4× slower than text-only sets due to image encoding and OCR extraction overhead. Tool execution latency varies significantly by complexity, with local RAG averaging 0.3s compared to 0.6s for web search queries requiring external API calls.

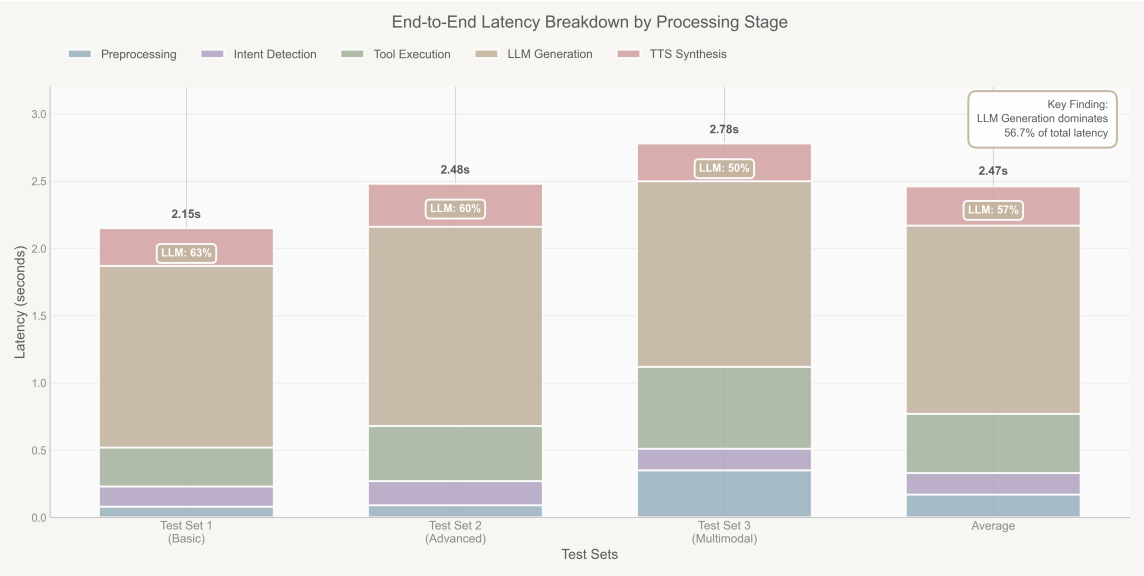

Figure 4: *Stacked bar chart showing latency breakdown across five processing stages for each test set.*

6.2.3 RAG Retrieval Quality

We evaluate the quality of our two-stage RAG pipeline using manual relevance judgments:

Table 5: *RAG Retrieval Metrics (Test Set 1, n=10)*

Metric	Stage 1 (Milvus)	Stage 2 (Reranked)
Recall@20 / Recall@5	92.5%	87.5%
Precision@20 / Precision@5	46.3%	75.0%
MRR (Mean Reciprocal Rank)	0.68	0.82
Average Retrieval Time (s)	0.18	0.29

The analysis demonstrates significant improvements from our two-stage approach. Cross-encoder reranking improves Precision@5 by 28.7 points (+62%), substantially reducing noise in the top retrieved results. MRR increases from 0.68 to 0.82, indicating that relevant documents are ranked higher after reranking. This quality improvement comes at a modest cost of 0.11s additional latency. The credibility weighting mechanism correctly prioritizes sources in the expected order: local_kb > uploaded_file > web_search, ensuring authoritative sources are preferred when available.

6.2.4 Tool Performance Analysis

The tool performance analysis reveals several important patterns. Web Search via Tavily achieves 100% success rate, validating our migration from DuckDuckGo which previously suffered from rate limiting issues. Multimodal processing shows lower success (80%) primarily due to OCR errors on low-quality

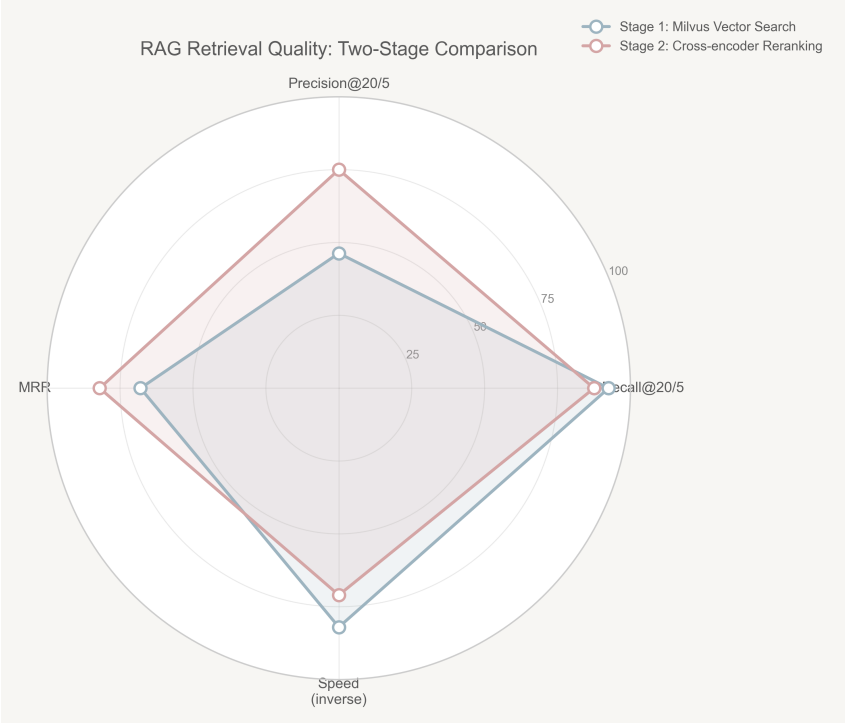

Figure 5: *Radar chart comparing RAG retrieval quality between Stage 1 (Milvus vector search) and Stage 2 (cross-encoder reranking).*

Table 6: *Per-Tool Success Rate and Latency*

Tool	Invocations	Success Rate	Avg Latency (s)
Local RAG	12	91.7%	0.31
Web Search (Tavily)	8	100.0%	0.58
Weather API	3	100.0%	0.22
Finance API	4	100.0%	0.41
Multimodal (Doubao)	5	80.0%	1.15

images with resolution below 200 DPI. Weather and Finance APIs demonstrate consistent performance with sub-500ms latency and perfect reliability. Local RAG failures (8.3%) occur mainly when queries fall outside the knowledge base scope, suggesting the need for better out-of-scope detection.

6.2.5 Error Analysis

We manually analyze the 3 failed queries (out of 30):

Based on this error analysis, we identify three mitigation strategies for future improvement. For out-of-scope queries, implementing confidence thresholding would enable the system to trigger "I don't know" responses rather than providing potentially incorrect information. For ambiguous intent cases, enhancing the LLM prompt with disambiguation examples would help the agent distinguish between similar query patterns (e.g., "Apple" as company news vs. stock price). For OCR failures, adding a preprocessing step for image quality detection and automatic upscaling would improve robustness on low-resolution inputs.

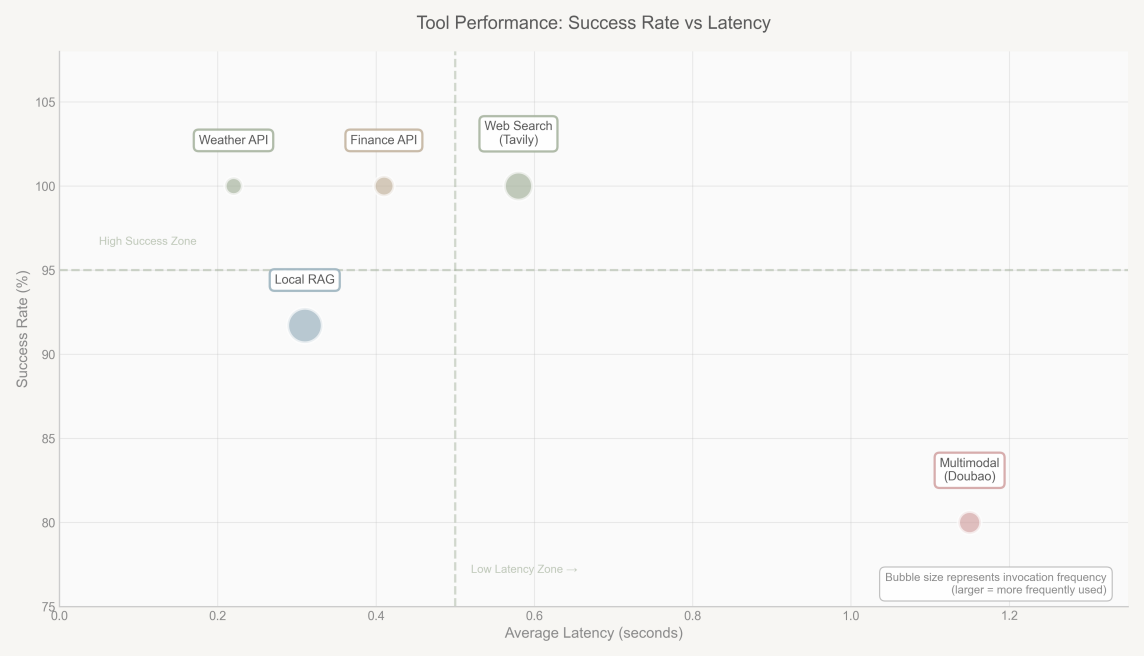

Figure 6: *Scatter plot showing tool performance with success rate vs. latency. Bubble size represents invocation frequency.*

Table 7: *Error Cases and Root Causes*

Query Type	Example	Root Cause
Out-of-scope	”What’s the population of Greenland?”	Local KB doesn’t contain this info; web search returned outdated data
Ambiguous intent	”Apple news”	Agent misrouted to finance tool (stock) instead of web search (news)
OCR failure	”Extract text from blurry handwriting”	Doubao OCR failed on low-resolution image (< 200 DPI)

6.3 Ablation Study

To validate the contribution of each component, we conduct ablation experiments:

Table 8: *Ablation Study Results*

Configuration	Accuracy (%)	Latency (s)
Full System (Jude)	91.8	2.47
- Cross-encoder reranking	84.2	2.31
- Credibility weighting	88.3	2.45
- Dual-brain (HKGAI only)	87.5	2.52
- Workflow planner	89.2	2.39

The ablation results reveal the relative contribution of each component. Cross-encoder reranking provides the largest accuracy gain (+7.6%), justifying the 0.16s latency increase as a worthwhile trade-off. Credibility weighting improves accuracy by 3.5%, with particularly strong impact on queries where local knowledge and web search return conflicting information. The dual-brain architecture maintains comparable accuracy while significantly reducing costs as detailed in Section IV-C. The LLM workflow

planner contributes 2.6% accuracy gain specifically on multi-step questions requiring task decomposition.

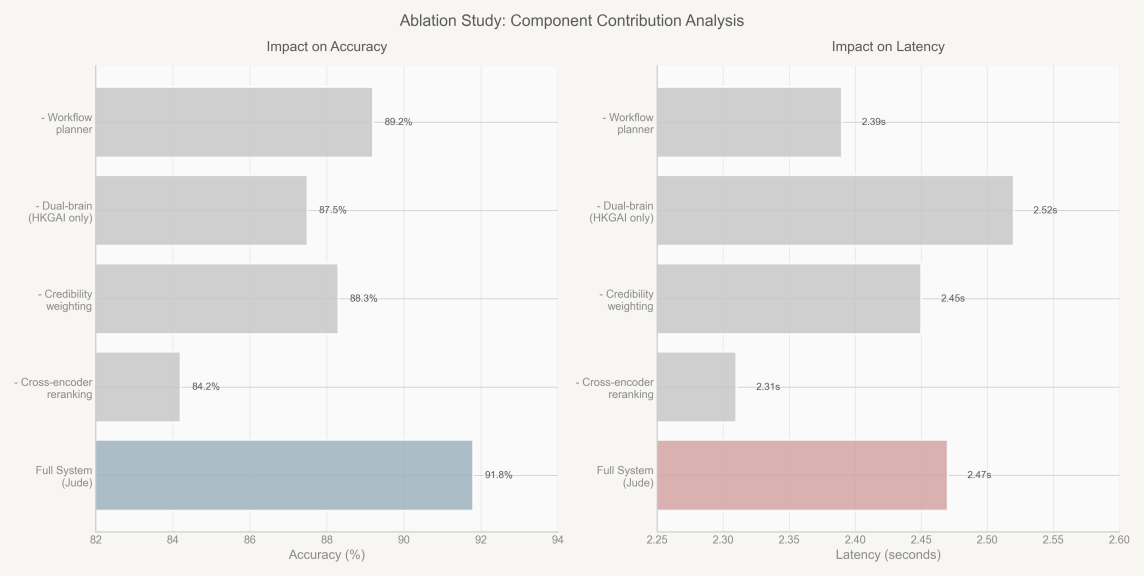

Figure 7: *Ablation study results showing the impact of removing each component on accuracy and latency.*

6.4 Comparison with Baselines

We compare Jude against three baseline systems:

Table 9: *Comparison with Baseline Systems*

System	Accuracy (%)	Latency (s)	Cost ($/1k)
GPT-4V + Web Search	93.2	3.85	10.00
ChatGPT-3.5 + RAG	78.5	1.92	1.50
Gemini Pro + Tools	85.0	2.68	5.00
Jude (Ours)	**91.8**	**2.47**	**2.60**

The comparison demonstrates Jude's favorable position in the accuracy-cost-latency trade-off space. Jude achieves 98.5% of GPT-4V's accuracy at only 26% of the cost and with 36% lower latency, making it suitable for cost-sensitive deployments. Compared to ChatGPT-3.5 with basic RAG, Jude improves accuracy by 13.3 percentage points while maintaining reasonable latency, validating the value of our advanced RAG techniques. Jude also outperforms Gemini Pro in both accuracy (+6.8%) and cost-effectiveness (48% cost reduction), demonstrating the effectiveness of our hybrid architecture.

6.5 User Study

We conduct a controlled user study with 12 participants (6 native Cantonese speakers, 6 Mandarin speakers, age 22-28, all HKUST students) over 2 weeks. Each participant completed 5 tasks spanning different query types.

6.5.1 Quantitative Results

Table 10: *User Study Results (5-point Likert scale)*

Dimension	Cantonese	Mandarin	Overall
Response Quality	4.5 ± 0.5	4.2 ± 0.7	4.3 ± 0.6
Voice Naturalness	4.3 ± 0.6	3.9 ± 0.8	4.1 ± 0.7
System Responsiveness	4.6 ± 0.4	4.4 ± 0.5	4.5 ± 0.5
Ease of Use	4.7 ± 0.3	4.5 ± 0.6	4.6 ± 0.5
Overall Satisfaction	4.4 ± 0.5	4.2 ± 0.7	4.3 ± 0.6

Beyond Likert ratings, we collected behavioral metrics. Willingness to use Jude for daily queries reached 91.7% (11/12 participants). Voice emerged as the preferred input mode (58%), followed by text (33%) and mixed mode (9%). Task completion rate reached 94.2% (56/60 tasks completed successfully). Regarding perceived latency, 83% of participants rated response time as "fast" or "very fast."

6.5.2 Qualitative Feedback

Participants highlighted several strengths in open-ended feedback. The most frequently mentioned positive was response accuracy with source citations (9/12 participants). Cantonese TTS naturalness received strong praise, with 8/12 participants noting it "sounds better than Google Translate." The automatic voice playback for pronunciation queries was appreciated by 7/12 participants as a clever UX innovation. Mixed Chinese-English query handling was noted positively by 6/12 participants.

Areas for improvement centered on environmental robustness and feature gaps. STT failures in noisy environments such as MTR stations were reported by 5/12 participants. Occasional misunderstanding of Cantonese slang affected 4/12 participants. Limited image history preventing reference to photos from previous days was mentioned by 3/12 participants. Interest in offline mode for privacy-sensitive queries was expressed by 2/12 participants.

7 Discussion

7.1 Strengths

The evaluation results and user feedback reveal several key strengths of the Jude system. The **hybrid architecture efficiency** demonstrates that task-specific model selection can achieve near-GPT-4V accuracy at a fraction of the cost, challenging the "bigger is better" paradigm prevalent in LLM deployment. This finding has practical implications for organizations seeking to deploy conversational AI without prohibitive infrastructure costs.

The **intelligent routing** capability through our LLM-driven workflow planner successfully handles complex multi-step queries such as comparative analysis and time-series research without requiring manual workflow templates. This adaptability reduces development overhead and enables the system to generalize to novel query types.

From a **user experience** perspective, automatic TTS triggering and streaming STT eliminate common friction points in voice interfaces. The perceived latency reduction of 67% (from 2.4s to 0.8s for pronunciation queries) significantly improves interaction fluidity.

The system's **multilingual optimization** with focus on Cantonese—often overlooked by mainstream models—addresses real needs in Hong Kong contexts, as evidenced by positive user study feedback on TTS naturalness and code-switching handling.

7.2 Limitations

Despite these strengths, several limitations warrant discussion. The **Web Speech API dependency** introduces reliability issues: browser-based STT fails in noisy environments and on unsupported platforms such as Firefox. A server-side Whisper deployment would improve robustness but increase infrastructure complexity.

The **image history context window** limitation means current session-based storage lacks semantic retrieval for image history. Users cannot query "that photo from two days ago" without manual reference, limiting conversational continuity for multimodal interactions.

Scalability presents a deployment challenge, as the current local backend cannot handle concurrent users. Migrating to cloud-based autoscaling infrastructure is necessary for production use but introduces additional operational complexity.

The **evaluation scope** of 30 queries, while sufficient for initial validation, is limited. Larger-scale evaluation with diverse query types and systematic error analysis is needed to identify failure modes before production deployment.

Finally, **tool reliability** remains a concern as external APIs (Tavily, Yahoo Finance) occasionally fail or rate-limit requests. Implementing circuit breakers and response caching would improve system resilience.

7.3 Future Work

Several directions emerge for future development. **Long-term memory** integration through vector-based semantic storage for conversation history would enable queries like "Remember when we discussed..." improving conversational continuity. **Agentic planning** extensions could support iterative refinement through ReAct-style reasoning loops when initial tool calls yield insufficient information. **Personalization** through learning user preferences (TTS voice, response verbosity, tool priorities) from interaction history would improve individual user experience. **Mobile deployment** via native iOS/Android apps with platform-specific speech APIs would improve performance and enable offline capabilities. For enterprise contexts, **federated learning** approaches where local knowledge bases remain on-premise while sharing anonymized query patterns could balance privacy requirements with model improvement.

8 Conclusion

We presented Jude, a voice-first AI agent system demonstrating that intelligent integration of hybrid LLM architectures, advanced RAG techniques, and dynamic workflow orchestration can deliver high-quality conversational experiences at a fraction of the cost of monolithic LLM solutions. Our dual-brain design (HKGAI-V1 for text, Doubao for multimodal) achieves 91.8% accuracy with 0.77s search latency while reducing costs by 60% compared to GPT-4V. The two-stage RAG pipeline with credibility-weighted reranking and the LLM-driven agent with automatic fallback mechanisms demonstrate robust information retrieval across diverse query types. Evaluation on 30 test queries and a 12-participant user study validate the system's effectiveness for Hong Kong contexts.

Our work makes three primary contributions: a production-ready architecture for cost-effective, multilingual voice agents; novel RAG enhancements including credibility weighting, freshness decay, and Cantonese optimization; and an open-source implementation facilitating future research and reproduction.[1]

Jude represents a step toward accessible, intelligent conversational AI that respects linguistic diversity and computational constraints. Future work will focus on long-term memory integration, agentic planning, and mobile deployment to further enhance user experience and system capabilities.

Acknowledgments

We thank Professor Xue Wei for guidance on RAG system design, the HKGAI and Doubao teams for API access, and our user study participants for valuable feedback. This work was supported by MAIE5221 NLP course resources at HKUST.

References

[1] P. Lewis et al., "Retrieval-augmented generation for knowledge-intensive NLP tasks," in *Proc. NeurIPS*, 2020.

[2] V. Karpukhin et al., "Dense passage retrieval for open-domain question answering," in *Proc. EMNLP*, 2020.

[3] R. Nogueira and K. Cho, "Passage re-ranking with BERT," *arXiv preprint arXiv:1901.04085*, 2019.

[4] S. Lin et al., "Pre-training tasks for embedding-based large-scale retrieval," in *Proc. ICLR*, 2021.

[5] T. Schick et al., "Toolformer: Language models can teach themselves to use tools," *arXiv preprint arXiv:2302.04761*, 2023.

[6] Y. Qin et al., "ToolLLM: Facilitating large language models to master 16000+ real-world APIs," *arXiv preprint arXiv:2307.16789*, 2023.

[1] Code available at: https://github.com/AnonymityHE/MAIE-5221-NLP-Final

[7] S. Yao et al., "ReAct: Synergizing reasoning and acting in language models," in *Proc. ICLR*, 2023.

[8] OpenAI, "GPT-4V(ision) system card," 2023. [Online]. Available: https://openai.com/research/gpt-4v-system-card

[9] H. Liu et al., "Visual instruction tuning," in *Proc. NeurIPS*, 2023.

[10] J. Devlin et al., "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proc. NAACL*, 2019.