壹、前言

一、研究動機

隨著人工智慧（Artificial Intelligence, AI）的快速發展，大型語言模型（Large Language Models, LLMs）逐漸成為各類智慧應用的基礎。這些模型展現了卓越的語言理解與生成能力，並推動了 AI Agent 在教育、客服、知識檢索與決策支援等領域的廣泛應用。然而，現有的 Agent 架構仍存在數個限制。

首先，目前缺乏一個統一的 Decision Layer（決策層），能有效協調多個模型與工具的運作，使其在處理複雜任務時輸出結果更具一致性與可靠性。其次，傳統的 API Agent 偏向於即時回應，往往僅能處理單一回合的任務互動，而缺乏能夠支援 長期互動的多層次記憶機制。這導致 Agent 在跨情境或跨任務時，難以延續先前的經驗與知識。最後，目前的記憶設計大多停留於簡單的上下文儲存，尚未形成結構化的記憶系統。例如，針對 事件、人、時間、地點與情緒 等資訊的系統化管理，仍未有成熟方案，這使得 Agent 難以隨著使用而逐步進化與優化。

基於上述不足，亟需提出一個能整合決策層與結構化記憶的 AI Agent 架構，以解決現有系統的限制，並推動 AI 在真實應用中的持續發展與落地。

二、研究目的

（一）建立一個標準化的工具呼叫格式，讓 AI 在操作多種工具時保持一致性與可控性。
（二）設計支援多平台協作的架構：確保 AI 能夠跨不同模型與外部系統協同運作。
（三）建立具自我拓展能力的系統：透過模組化設計，使系統能隨需求不斷擴充新功能。
（四）發展多維度記憶系統：支援長短期記憶讀寫，並結構化管理事件、人、時間、地點與情緒等資訊。

貳、文獻探討

一、什麼是大型語言模型 LLM?

大型語言模型（Large Language Models, LLMs）是一類基於深度學習（Deep Learning）技術，並以 Transformer 架構 為核心的自然語言處理模型。其主要特徵在於透過大規模文本語料進行預訓練（Pre-training），使模型能夠學習語言的統計特徵、語義關係與上下文依存。由於參數量可達數十億甚至數千億，LLMs 具備處理各種語言任務的泛化能力，包括機器翻譯、問答系統、文本生成、程式碼輔助等，LLMs 已成為近年 AI 技術進步的核心推動力。例如，OpenAI 的 GPT 系列、Google 的 Gemini、Anthropic 的 Claude，以及 Meta 的 LLaMA 等，皆在不同任務上展現了卓越的表現。

表一：Gemini 與 GPT 的性價比分析

Models	Input/Output(USD)	MMLU-Pro 完成率
Gemini 2.5 Flash	0.100/2.500	**unknown**
Gemini 2.5 Flash-Lite	0.100/0.400	**unknown**
Gemini 2.0 Flash	0.100/0.400	0.7624
Gemini 2.0 Flash-Lite	0.075/0.300	0.716
GPT-5-mini	0.250/2.000	0.871
GPT-4.1 mini	0.800/3.200	0.83
GPT-5-nano	0.050/0.400	**unknown**

資料來源：https://huggingface.co/spaces/galileo-ai/agent-leaderboard

二、什麼是 AI Agent?

AI Agent（人工智慧代理人）是一種基於大型語言模型（Large Language Models, LLMs）或其他 AI 技術的系統，其核心特徵在於能自主感知環境、規劃行動並執行任務。不同於傳統的聊天機器人僅能進行問答，AI Agent 具備任務導向性(task-oriented)與工具調用能力(tool use），能在複雜的應用場景中表現出更高的靈活性與智能。

三、AI Agent 的基本運作流程：

（一）決策 （Decision Making）：根據使用者輸入或外部環境，判斷應採取的行動。
（二）工具調用 （Tool Invocation）：透過 API、外部系統或其他模型，執行決策所需的操作。
（三）回饋 （Feedback & Adaptation）：根據執行結果進行修正，並調整後續策略。

2024 年的研究顯示，AI Agent 在「多步驟工具操作」上的表現仍有顯著挑戰。例如，在 GTA Benchmark（General Tool Agent Benchmark）中，最先進的 GPT-4 系列模型的任務完成率僅約 50%，而大多數模型甚至低於 25%。這表示，即使模型具備強大的語言推理能力，若缺乏完善的決策層與記憶機制，其在真實場景下的工具操作仍顯不足。

另一份 Galileo-AI Agent Leaderboard 的公開評測也顯示，即便是 GPT-4.1，其在工具操作情境下的平均完成率也僅 62%，而 Claude-Sonnet 的工具選擇準確率高達 92%，但最終任務完成度仍只有 55%。這突顯出「工具選擇」與「任務完成」之間存在鴻溝，需要額外的 Decision Layer 來協調。

表二：Gemini 與 GPT 在任務完成度上的比較

Models	Vendor	平均任務完成度
gpt-4.1	OpenAI	0.620
gpt-4.1-mini	OpenAI	0.560
gemini-2.5-flash-lite	Google	0.470
gemini-2.5-pro	Google	0.430
gemini-2.5-flash	Google	0.380

資料來源：OpenAI、AI Studio

什麼是 memory 系統?在 AI Agent 的架構中，記憶系統（Memory System） 扮演了關鍵角色。它決定了模型是否能在多輪對話或跨任務情境下保持一致性、延續性與可進化性。傳統的大型語言模型（LLMs）僅能依賴 上下文視窗（context window），在處理長期交互時往往受限於 Token 上限（例如 GPT-4o 的 128k tokens，Claude-3 Opus 的 200k tokens），一旦超出範圍，舊資訊就會被遺忘。因此，額外的記憶機制成為 AI Agent 研究中的重要方向。

（四）現有的記憶方法：

1、目前常見的 AI 記憶系統大致可分為以下幾類：

（1） Buffer Memory（緩衝記憶）

將對話內容完整保存並附加於 Prompt 中，能確保上下文不會遺失，實作方式相對簡單直接；然而，隨著對話規模逐漸增長，會導致 Token 消耗快速上升，進而增加成本。

（2） Summary Memory（摘要記憶）

將歷史對話壓縮為摘要，再餵回模型，以降低成本並保留主要內容。這種方法能有效減少 Token 消耗，但在壓縮過程中會遺失部分細節，可能導致模型在後續推理或回應時產生誤判。

（3） Vector Database Memory（向量資料庫記憶）

將對話或文件轉換為 Embeddings，存入向量資料庫（如 FAISS、Milvus），並透過語義相似度檢索所需資訊。此方法能處理大規模歷史資料，並具備高

速檢索的優點；然而，檢索結果未必符合當前語境需求，且缺乏上下文的邏輯關聯。

（4）Generative Memory（生成式記憶）

近幾年的研究開始嘗試「生成記憶」，例如 Generative Agents （Park et al., 2023），模擬虛擬小鎮居民的長期記憶與行為習慣。

這種方法讓 Agent 能根據過往事件生成未來行為，但仍存在可控性不足的問題。

（5）反思式記憶（Reflexion）

Reflexion （Shinn et al., 2023） 提出讓模型對自己的失敗做反思，進而調整行為。相當於一種「經驗學習」，補足 LLM 缺乏持續優化的缺陷。

2、現有系統的局限性：

（1） 缺乏結構化管理：大部分記憶僅基於文字或向量檢索，無法清楚區分「事件、人、時間、情緒」等不同維度。

（2） 短期導向：Buffer 與 Summary 偏向短期維護，缺乏能支援「長期演進」的機制。

（3） 檢索限制：Vector Memory 依賴語義相似度，可能在跨領域或多層邏輯任務中表現不佳。

（4） 進化不足：目前的記憶大多是「靜態儲存」，尚缺乏能持續自我組織與優化的框架。

參、研究方法

一、指令架構設計

為了提升 AI Agent 在決策過程中的一致性與效率，本研究提出一套 分層指令歷史（Command History Hierarchy），將歷史操作依據使用頻率與重要性劃分為三個層級，並輔以統一化指令格式，以確保系統在執行過程中具備可控性與擴展性。

（一）分層指令歷史（Command History Hierarchy）

本研究將歷史操作紀錄設計為三個層級：

1、L3（完整歷史層）

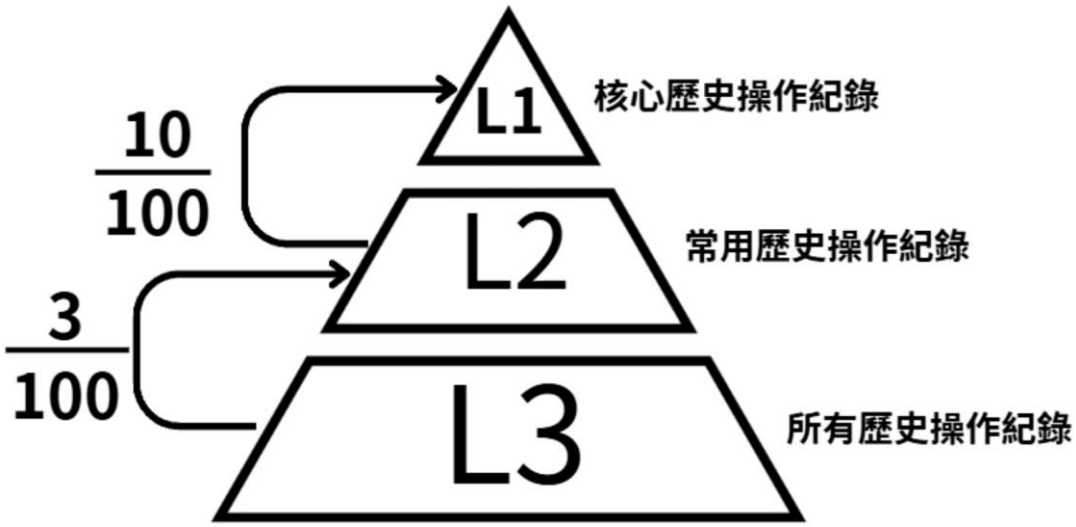
系統最多保留最近 100 筆操作紀錄，並僅將最新 25 筆輸入 Decision Layer 參考。此設計確保 AI 能掌握最新上下文，同時避免過長歷史造成 Token 成本與延遲過高。

2、L2（常用歷史層）

由 L3 中篩選出「高頻使用」的指令，形成使用者的操作習慣摘要。此層級能補足 L3 的即時性不足，確保 Decision Layer 在決策時能同時考慮長期偏好。

3、L1（核心歷史層）

從 L2 進一步濃縮為「最常使用」或「任務關鍵」的指令，作為 AI 的核心行為基礎。L1 的存在提升了系統在高頻任務中的穩定性與一致性，避免 AI 偏離主要任務流程。



圖一：歷史操作紀錄分層架構
圖片來源：自製圖表

透過此三層結構，AI 能在決策時兼顧「近期上下文」與「長期偏好」，形成既靈活又穩定的行為模式。

（二）統一化指令格式（Unified Command Schema）

為了提升指令的可讀性與可擴展性，本研究採用 JSON 格式統一表示指令{ "command": "工具名稱.function", "value": "根據 function 填入對應參數" }，此設計確保所有指令具備一致性，並能隨著系統需求擴展而持續增長，僅需新增新的 command 類別即可實現新功能，無須改變整體架構。

（三）指令執行流程（Execution Pipeline）

與傳統 AI Agent 直接將使用者需求轉換為單一步驟指令不同，本研究的 Decision Layer 採用「獨立執行區域（Isolated Execution Space）」進行操作。AI 的指令規劃與執行均在此區域完成，並透過工具的自帶說明（*.help）確保每一步驟均為合法且正確的行為。所有執行紀錄會自動寫入 完整歷史池（L3，100 筆），再由系統篩選生成 L2 與 L1，以降低模型的運算負擔。

1、 需求解析
使用者輸入需求（例如：「幫我查詢現在幾點」），Decision Layer 將其轉化為可執行的行為目標。
2、 工具探索
AI 透過 *.help 指令查詢相關工具的操作說明。例如：
earch.help → 取得搜尋工具的功能與可用指令。
3、 工具執行

根據工具說明，AI 執行對應的指令：
search.time → 查詢目前時間。

4、 中繼回覆
工具回傳執行結果（例如：「現在時間為 14:32」），此回覆僅存在於獨立執行區域，尚未回傳至使用者。

5、 系統確認
AI 發現需對使用者進行回覆，因而再次呼叫 system.help，確認如何輸出訊息。

6、 最終回覆
AI 下達統一化回覆指令：{ "command": "system.message", "value": "現在時間是 14:32" }
系統將此輸出傳回使用者，並結束當前任務。

7、 歷史更新
本次所有操作紀錄自動寫入 L3，並由系統篩選最常用指令至 L2 與 L1。

（四）設計理念

1、 降低 AI 負擔：所有紀錄自動進入歷史池，再由系統篩選，避免模型自行處理大規模紀錄。
2、 確保執行正確性：每個工具需先查詢 *.help，確保操作符合工具規格，避免非法調用。
3、 分離回覆與執行：AI 的中繼操作與使用者回覆分開處理，提升任務靈活性。
4、 維持可控性：所有輸出最終統一透過 system.message，確保回覆格式一致。

（五）記憶架構題區分 （Topic-based Segmentation）

1、採用「短期記憶 （Short-term） 與長期記憶 （Long-term）」的雙層設計。
2、短期記憶用於即時任務的上下文保持，避免 AI 在多輪互動中遺失關鍵訊息。
3、長期記憶則以「話題」為索引，將對話或事件劃分至不同主題，確保資訊檢索時更具針對性。

（六）多維查詢架構 （Multi-dimensional Querying）

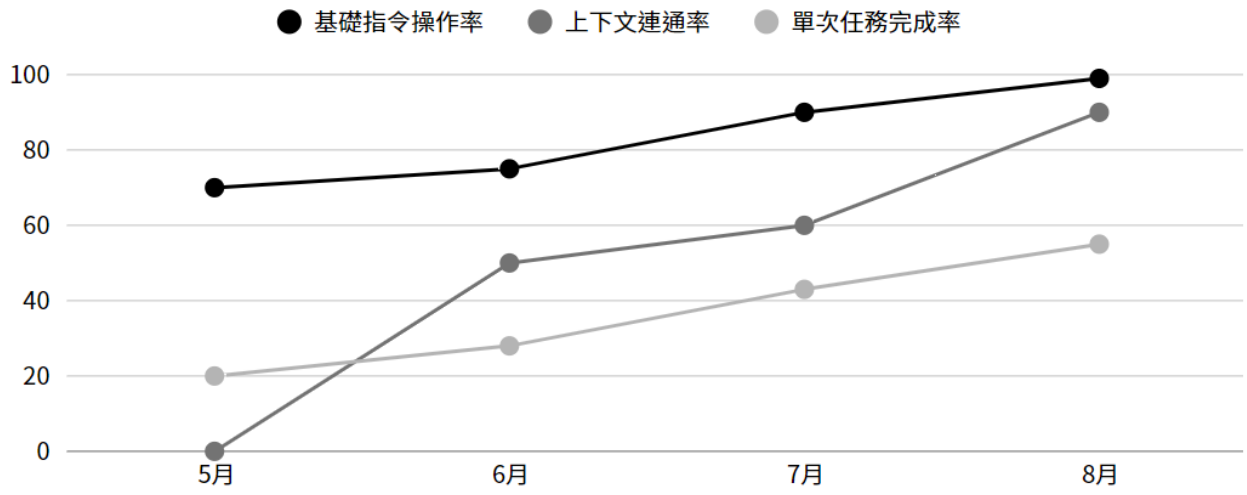
1、 事件 （Event）
2、 人物 （People）
3、 時間 （Time）
4、 地點 （Place）
5、 情緒 （Emotion）
6、 物件歷程 （Physical）

（七）混合式記憶策略

1. Buffer：保存最近的互動（如 25 筆）。
2. Summary：透過主題劃分與頻率篩選，提取長期「習慣」與「核心偏好」。
3. Generative：透過 AI 生成與更新事件檔案，讓記憶能隨著互動動態演進。

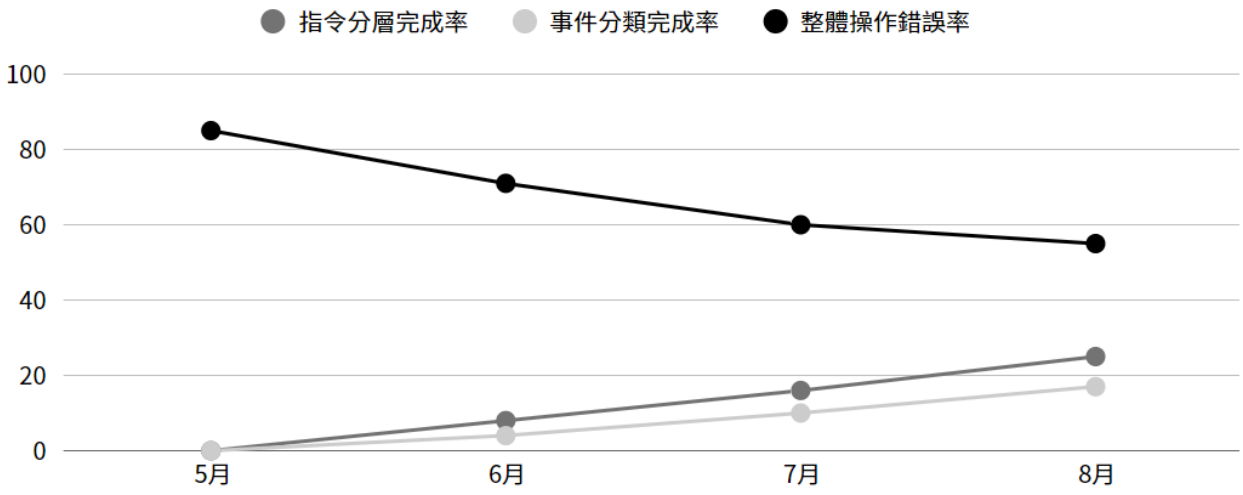
肆、研究分析

表三：操作能力成長趨勢圖



圖表來源：自製圖表

表四：操作分層與錯誤率趨勢圖



圖表來源：自製圖表

伍、研究結論

本研究驗證了架構在基礎指令處理與任務管理上的可行性，並展現出逐步提升的潛力。然而，系統仍存在幾個需持續優化的面向：首先，指令分層與事件分類需要進一步細化，以確保複雜任務的精準拆解；其次，上下文的連貫處理仍需加強，才能在多回合互動中保持穩定性；最後，工具操作的指引尚待優化，以提升使用者在實際操作時的效率與正確性。

整體而言，系統已展現出降低錯誤率與提升操作流暢度的成效，未來若能在細緻度與可用性上持續改進，將能更完整發揮其作為 AI 中樞架構的價值。

陸、參考文獻

1. Timo Schick（2023）Toolformer: Language Models Can Teach Themselves to Use Toolshttps://arxiv.org/pdf/2302.04761
2. Yaobo Liang（2023）TaskMatrix.AI: Completing Tasks by Connecting Foundation Models with Millions of APIshttps://arxiv.org/pdf/2303.16434
3. Shuofei Qiao（2023）Making Language Models Better Tool Learners with Execution Feedbackhttps://arxiv.org/pdf/2305.13068
4. Sirui Hong（2023）METAGPT: META PROGRAMMING FOR A MULTI-AGENT COLLABORATIVE FRAMEWORKhttps://arxiv.org/pdf/2308.00352
5. Qingyun Wu（2023）AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversationhttps://arxiv.org/pdf/2308.08155
6. DUNG DAO （ 2025 ） Multi-Agent Collaboration Mechanisms: A Survey of LLMshttps://arxiv.org/pdf/2501.06322
7. Joon Sung Park （ 2023 ） Generative Agents: Interactive Simulacra of Human Behaviorhttps://arxiv.org/pdf/2304.03442
8. Noah Shinn（2023）Reflexion: Language Agents with Verbal Reinforcement Learning https://arxiv.org/pdf/2303.11366
9. Jiazheng Kang（2025）Memory OS of AI Agenthttps://arxiv.org/pdf/2506.06326
10. Tapio Pitkäranta,Leena Pitkäranta（2025）HADA: HUMAN-AI AGENT DECISION ALIGNMENT ARCHITECTURE https://arxiv.org/pdf/2506.04253
11. OpenAi：https://openai.com/zh-Hant/
12. AI Studio：https://aistudio.google.com/prompts/new_chat