

AI 發展與應用深度報告

從搜尋爬蟲到智能代理：人工智能的演進與未來

作者：筑夢

日期：2026 年 4 月 18 日

摘要

本報告探討人工智能（AI）從早期搜尋技術到現代智能代理（AI Agent）的演進歷程，深入分析技術核心概念包括 Token、API 以及 AI 系統的局限性與風險。同時探討開源與自託管（Self-hosted）方案如 OpenClaw 的興起，並展望人機協作的未來趨勢。

第一章：網絡資訊的開荒時代 — 搜尋引擎與爬蟲

1.1 互聯網初期的資訊孤島

1990 年代互聯網誕生之初，網頁數量稀少，人們依靠人手整理的書籤或目錄（Directory）來尋找資訊。隨著萬維網（World Wide Web）快速擴張，人手整理變得不可能，自動化的「爬蟲」（Crawler/Spider）應運而生。

1.2 爬蟲的工作原理

爬蟲本質上是一個自動化程序，24 小時不間斷地：

- 發現**：從已知網址出發，透過超連結發現新頁面
- 下載**：抓取網頁 HTML 內容

- 3. **解析**：提取文字、標題、連結等結構化數據
- 4. **索引**：建立倒排索引 (Inverted Index) 以便快速搜尋
- 5. **排名**：根據相關性、權威性等因素決定搜尋結果排序

Google 的 PageRank 演算法革命性地將「連結數量」轉化為「權威性指標」，使搜尋質素大幅提升。

1.3 局限性：理解與處理的鴻溝

早期搜尋引擎有個根本缺陷：它們只懂關鍵字匹配，並不理解內容意義。例如搜尋「蘋果」，引擎無法區分是水果還是科技公司，只能依靠使用者的其他關鍵字 (如「食」、「iPhone」) 來推斷。這種「語意鴻溝」成為下一代 AI 技術需要突破的核心問題。

第二章：對話式 AI 的黎明 — Chatbot 的興起

2.1 從 ELIZA 到現代 Chatbot

1966 年，MIT 的 Joseph Weizenbaum 開發了 ELIZA，透過簡單的規則匹配模擬心理治療師對話。雖然 ELIZA 並不理解對話內容，但它展示了人類對「機器對話」的強烈興趣。

現代 Chatbot 經歷了三個階段：

階段	技術	能力	例子
第一代	規則引擎	固定問答配對	早期客服機械人
第二代	檢索式	從數據庫找最相似答案	FAQ 機械人
第三代	生成式	理解上下文，生成新回應	ChatGPT、Claude

2.2 Chatbot 的核心挑戰

上下文理解：人類對話充滿指代、省略和暗示。例如：

「部電話幾時到？」

—— 這句話依賴前文提到的「買咗部新 iPhone」。

早期 Chatbot 無法維持長篇對話的上下文，容易「失憶」。現代大語言模型 (LLM) 透過 Attention 機制和巨大的上下文窗口 (Context Window) 大幅改善此問題。

意圖識別：同一個問題可能有不同意圖。例如「我要訂位」可能是餐廳訂座、機票預訂或酒店預訂，需要結合場景推斷。

第三章：AI Agent — 從對話到行動

3.1 什麼是 AI Agent ？

AI Agent (智能代理) 是 AI 發展的重要里程碑。與傳統 Chatbot 「問一答一」的被動模式不同，Agent 具備：

- **目標導向：**理解使用者的高層次意圖
- **自主規劃：**拆解任務，制定執行步驟
- **工具使用：**調用 API、查詢數據庫、執行代碼
- **記憶持久：**長期記住使用者的偏好和過往互動
- **反思修正：**根據執行結果自我調整

3.2 Agent 的實際應用場景

場景一：行程規劃

使用者說：「下個月我想去東京玩幾日，預算一萬蚊。」

Agent 會：

1. 查詢航班價格和時間
2. 搜尋符合預算的酒店
3. 根據使用者興趣推薦景點
4. 規劃每日行程路線
5. 預訂餐廳
6. 生成完整行程表和預算明細

場景二：程式開發助手

使用者描述功能需求，Agent 可以：

1. 撰寫代碼
2. 執行測試
3. 發現錯誤後自我修正
4. 部署到伺服器

3.3 ReAct 框架：推理與行動的結合

現代 Agent 多採用 ReAct (Reasoning + Acting) 框架：

思考 (Thought) → 行動 (Action) → 觀察 (Observation) → 循環

這種設計讓 AI 能夠「邊想邊做」，例如先思考「需要查天氣」，然後調用天氣 API，再根據結果決定下一步。

第四章：Token — AI 理解世界的「貨幣」

4.1 什麼是 Token ？

Token 是 LLM 處理文字的基本單位。與人類以「字」為單位不同，AI 使用 Token：

語言	大約比例
英文	1 字 $\approx$ 0.75 Token
中文	1 字 $\approx$ 1.5-2 Token
廣東話	類似中文，視乎用字

例如句子「我愛香港」可能被拆解為：「我 | 愛 | 香 | 港」或「我 | 愛 | 香港」，視乎使用的 Tokenizer。

4.2 Token 的技術原理

現代 LLM 使用 **Byte Pair Encoding (BPE)** 或類似的子詞 (Subword) 算法：

- 1. **頻繁組合優先**：常見的字組合 (如「香港」、「人工智能」) 會被視為單一 Token
- 2. **生僻字拆解**：罕見字會被拆成更小的單位甚至字符
- 3. **多語言支持**：同一套 Tokenizer 可處理多種語言

4.3 Context Window：AI 的「記憶容量」

Context Window 指模型能同時處理的 Token 數量上限：

模型	Context Window
GPT-3.5	4K-16K tokens
GPT-4	8K-128K tokens
Claude 3	200K tokens
Gemini	1M+ tokens

限制 Context Window 意味著：

- 長文檔需要分段處理

- 超長對話會「遺忘」早期內容
- 需要 RAG (檢索增強生成) 等技術補充外部記憶

4.4 Token 的經濟意義

Token 也是 AI 服務的計費單位：

- **Input Tokens**：送給 AI 的提示 (Prompt)
- **Output Tokens**：AI 生成的回應

理解 Token 有助於優化成本：簡潔的提示、明確的指示可以減少不必要的 Token 消耗。

第五章：API — AI 與世界的橋樑

5.1 API 的本質

API (Application Programming Interface，應用程式介面) 是軟件之間的「通用語言」。就像餐廳菜單讓顧客無需了解廚房運作也能點菜，API 讓應用程式無需了解內部實現也能使用服務。

5.2 AI 時代 API 的重要性

AI Agent 的能力很大程度上取決於它能調用什麼 API：

API 類型	功能	例子
搜尋 API	獲取即時資訊	Google Search API、Bing API
地圖 API	位置、導航、地點資訊	Google Maps、OpenStreetMap
天氣 API	氣象數據	OpenWeatherMap

API 類型	功能	例子
通訊 API	發送訊息	WhatsApp Business API、Twilio
金融 API	支付、交易	Stripe、PayPal
數據庫 API	存取結構化數據	PostgreSQL、MongoDB

5.3 Function Calling : AI 使用工具的能力

現代 LLM 支援 **Function Calling (函數調用)** :

1. 開發者定義可用函數 (名稱、參數、描述)
2. AI 判斷是否需要調用函數
3. AI 生成函數調用的參數
4. 系統執行函數，將結果返回給 AI
5. AI 綜合結果生成回應

這機制讓 AI 從「純聊天」進化為「能執行任務」的 Agent。

5.4 API 的安全與隱私考量

- **權限控制** : AI 應該只能調用必要的 API，遵循最小權限原則
 - **數據洩露** : 避免將敏感數據送給第三方 API
 - **成本控制** : API 調用通常有費用，需要限流機制
 - **錯誤處理** : API 可能失敗，AI 需要學會處理異常
-

第六章：OpenClaw — 自主可控的 AI 基礎設施

6.1 為何需要自託管？

主流 AI 服務 (ChatGPT、Claude 等) 雖然方便，但存在隱憂：

問題	說明
數據隱私	對話記錄儲存在廠商伺服器
服務依賴	網絡中斷、API 限制會影響使用
費用波動	訂閱費用隨時調整
功能限制	無法自定義行為或接入內部系統
審查機制	某些話題可能被過濾

6.2 OpenClaw 的設計理念

OpenClaw 代表了「Bring Your Own AI」的趨勢：

- 多模型支持：整合 Gemini、Claude、GPT、本地模型等
- 多平台接入：WhatsApp、Telegram、Discord、iMessage
- 自託管：運行在自己的伺服器或電腦
- 插件系統：可擴展功能
- 記憶持久：長期記住用戶偏好

6.3 本地模型 vs 雲端模型

自託管可以選擇：

雲端模型 (透過 API)：

- 優點：能力強大、持續更新
- 缺點：需要網絡、數據離開本地

本地模型 (如 Llama、Mistral) :

- 優點：完全離線、數據絕對私有
- 缺點：硬件要求高、能力較弱

混合方案：

- 敏感任務用本地模型
- 複雜任務用雲端模型
- 根據場景自動切換

6.4 部署考量

自建 AI 基礎設施需要考慮：

- **硬件**：GPU 顯存決定可運行的模型大小
- **網絡**：對外連接的穩定性
- **安全**：認證、加密、存取控制
- **備份**：對話記錄、配置的備份
- **更新**：模型和軟件的持續維護

第七章：Prompt Engineering — 與 AI 溝通的藝術

7.1 什麼是 Prompt Engineering ?

Prompt Engineering (提示工程) 是指設計和優化給 AI 的指令 (Prompt) , 以獲得最佳回應的技術。這不僅僅是「問問題」, 而是一門關於如何精確表達意圖、引導 AI 思維、控制輸出格式的學問。

好的 Prompt 能讓 AI 從「泛泛而談」變為「精準有用」，從「答非所問」變為「切中要點」。

7.2 核心原則：CRISPE 框架

CRISPE 是一個實用的 Prompt 設計框架：

字母	代表	含義	例子
C	Context	提供背景情境	「你是一個資深的 Python 開發者...」
R	Role	指定角色定位	「請以專案經理的身 份...」
I	Intent	明確表達意圖	「我需要一份詳細的 技術方案...」
S	Specificity	具體化要求	「包含三個主要部 分，每部分 300 字」
P	Examples	提供參考範例	「類似這樣的格式： [例子]」
E	Evaluation	設定評估標準	「請確保建議具有可 執行性」

7.3 實用技巧大全

技巧一：角色扮演 (Role Prompting)

原理：AI 會根據角色調整語氣、知識領域和思維方式。

範例：

✗ 差：

「解釋區塊鏈」

✓ 好：

「你是一位資深的區塊鏈技術專家，正在向一位完全不懂技術的 CEO 解釋區塊鏈。

請用商業比喻而非技術術語，確保對方能在 5 分鐘內理解核心價值。」

技巧二：思維鏈 (Chain of Thought)

原理：要求 AI 逐步推理，而非直接給答案，提高複雜問題的準確率。

範例：

「請解決這道數學題：一間商店賣蘋果，第一天賣了 120 個，第二天比第一天多賣 25%，

兩天共賣了多少個？

請按以下步驟思考：

1. 先找出第一天的銷售量
2. 計算第二天的銷售量
3. 計算兩天的總和
4. 給出最終答案」

技巧三：少樣本學習 (Few-Shot Learning)

原理：透過提供 2-3 個例子，讓 AI 學習你想要的格式和風格。

範例：

「請將以下產品描述轉換為廣告標語：

範例 1：

產品：智能手錶，功能：健康監測、訊息通知

標語：「腕間健康管家，重要時刻不錯過」

範例 2：

產品：降噪耳機，功能：主動降噪、40 小時續航

標語：「靜享純淨，聆聽無限」

現在請為以下產品創作標語：

產品：便攜咖啡機，功能：3 分鐘快速萃取、無需電源」

技巧四：輸出格式控制

原理：明確指定想要的格式，減少後續處理工作。

常用格式指令：

「請以以下格式回應：

核心觀點

[一兩句話總結]

詳細分析

- 要點 1：[說明]

- 要點 2：[說明]

- 要點 3：[說明]

行動建議

1. [具體步驟]

2. [具體步驟]

3. [具體步驟]

表格格式：

「請將比較結果整理成表格，包含以下欄位：

| 方案 | 優點 | 缺點 | 適用場景 | 成本估算 |」

代碼格式：

「請提供 Python 代碼，要求：

- 包含詳細的註釋

- 使用 Type Hints

- 包含錯誤處理

- 添加使用範例」

技巧五：分解複雜任務

原理：將大任務拆成小步驟，降低 AI 的認知負荷。

範例：

✗ 差：

「幫我寫一個完整的電商網站」

✓ 好：

「我們要開發一個電商網站，請按以下階段協助：

第一階段：需求分析

- 列出核心功能模塊

- 建議技術棧

- 估算開發時間

第二階段：資料庫設計

- 設計主要的數據表結構

- 說明表之間的關係

第三階段：API 設計

- 設計 RESTful API 端點

- 包含請求/響應格式

我們先從第一階段開始。」

技巧六：設定約束條件

原理：透過明確的邊界，引導 AI 生成符合需求的內容。

常用約束：

「請遵守以下限制：

- 字數：500-600 字
- 語氣：專業但友善
- 避免：過度技術化的術語
- 必須包含：成本效益分析
- 目標讀者：中小企業主」

技巧七：負面提示 (Negative Prompting)

原理：明確告知 AI 「不要做什麼」，避免不想要的輸出。

範例：

「請分析這份市場報告。

請注意：

- 不要重述報告中已有的數據
- 不要給出過於籠統的建議
- 不要假設我們有無限預算

請專注於：

- 找出三個被忽視的市場機會
- 提出可執行的下一步行動」

技巧八：迭代優化

原理：對話是一個迭代的過程，根據 AI 的回應不斷修正方向。

範例對話：

用戶：「請介紹太陽能發電」

AI：[給出一般性介紹]

用戶：「很好，但我更關心香港住宅安裝太陽能板的實際情況，

包括法規限制、成本回收期、適合的屋苑類型」

AI：[給出針對性回應]

用戶：「請加入具體的政府資助計劃，以及兩三個實際案例」

7.4 不同場景的 Prompt 模板

場景一：寫作輔助

「你是一位資深的 [類型] 編輯。我正在撰寫一篇關於 [主題] 的 [文章類型]，

目標讀者是 [描述]。

請協助我：

1. 提供 5 個吸引人的標題選項
2. 建議文章的結構大綱
3. 針對以下草稿給出修改建議：

[你的草稿]」

場景二：程式開發

「你是一位 [語言] 專家。我需要實現以下功能：

功能描述：

[詳細描述]

輸入範例：

[例子]

預期輸出：

[例子]

要求：

- 時間複雜度優化
- 包含單元測試
- 處理邊界情況
- 代碼風格符合 PEP8」

場景三：數據分析

「我有一份數據，欄位包括：[列出欄位]。

請幫我：

1. 建議適合的分析角度
2. 提出 3-5 個可視化圖表的想法
3. 用 Python pandas 寫出數據清洗和分析的代碼框架
4. 指出分析時需要注意的潛在問題」

場景四：學習輔導

「我想學習 [主題]，目前的基礎是 [描述]。

請為我設計一個學習計劃：

- 總共 [時間] 的學習周期
- 每週大約 [時間] 的學習時間
- 包含理論和實踐的平衡
- 推薦具體的學習資源
- 設定階段性的里程碑和驗收標準」

場景五：決策分析

「我面臨以下決策情境：

[描述情境]

可選方案：

1. [方案 A]
2. [方案 B]
3. [方案 C]

決策考量因素（按重要性排序）：

1. [因素 1]
2. [因素 2]

3. [因素 3]

請幫我：

- 分析每個方案的優劣
- 指出我可能忽略的因素
- 如果資訊不足，請告訴我還需要什麼資訊
- 給出你的建議和理由」

7.5 常見錯誤與修正

錯誤類型	例子	修正後
過於模糊	「幫我寫點東西」	「幫我寫一封求職信，應徵市場經理職位，強調 5 年數碼營銷經驗」
一次問太多	「解釋量子力學、相對論和弦理論」	「我們先集中理解量子力學的基本概念，特別是波粒二象性」
缺乏背景	「這個怎麼樣？」	「考慮到我的預算 5000 元和初學者身份，這個吉他型號適合嗎？」
沒有指定格式	「總結這篇文章」	「用三個要點總結這篇文章，每個要點不超過 20 字」
假設 AI 知道上下文	「繼續剛才的話題」	「繼續關於 React 性能優化的討論，特別是虛擬列表的實

錯誤類型	例子	修正後
		現」

7.6 進階技巧

技巧一：自我一致性檢查

「請回答這個問題，然後檢查你的答案是否自治。

如果發現矛盾，請修正並說明原因。」

技巧二：多角度分析

「請從以下三個角度分析這個問題：

1. 經濟學家的觀點
2. 社會學家的觀點
3. 環保主義者的觀點

每個角度給出主要論點和潛在盲點。」

技巧三：對抗性思考

「請論證 [觀點 A] 的正確性。

然後，扮演反對者，論證 [觀點 B] 的正確性。

最後，綜合雙方觀點，給出平衡的結論。」

7.7 Prompt 的未來趨勢

1. **結構化 Prompt**：使用 XML、JSON 或專用語法標記不同部分，提高 AI 理解準確度。

<role>你是一位醫生</role>

<context>病人主訴頭痛，持續三天</context>

<task>列出可能的病因，但不要給出確診</task>

<constraint>使用通俗語言，避免醫學術語</constraint>

2. 自動化 Prompt 優化：AI 系統開始內置 Prompt 優化功能，自動改寫用戶的粗略輸入。

3. 個人化 Prompt 庫：用戶建立自己的 Prompt 模板庫，根據場景快速調用。

4. 多模態 Prompt：結合圖片、音頻、視頻作為 Prompt 的一部分，實現更豐富的表達。

7.8 實用建議

對於初學者：

1. 從模仿開始：收集好用的 Prompt 範例，理解其結構
2. 多嘗試：同一個問題用不同方式問，比較結果
3. 建立模板：為常見任務建立自己的 Prompt 模板庫

對於進階用戶：

1. 系統學習：了解 Transformer 架構，理解 AI 的運作原理
 2. 版本控制：記錄 Prompt 的迭代版本，追蹤效果變化
 3. A/B 測試：對重要應用，測試不同 Prompt 的效果差異
-

第八章：AI Hallucination — 幻覺與現實的邊界

7.1 什麼是 AI 幻覺？

AI Hallucination (幻覺) 指 AI 自信滿滿地生成虛假或無根據的內容。這是 LLM 最危險的缺陷之一。

常見幻覺類型：

- 1. **事實錯誤**：編造不存在的歷史事件、人物
- 2. **引用虛構**：創造假的研究論文、書籍、網址
- 3. **代碼錯誤**：生成看似合理但無法運行的程式
- 4. **過度推斷**：從有限信息得出錯誤結論

7.2 為何會產生幻覺？

技術原因：

- LLM 本質是「概率預測機器」，根據統計模式生成內容
- 訓練數據有截止日期，不了解最新事件
- 優化目標是「流暢自然」而非「絕對準確」

觸發因素：

- 提問過於模糊或超出知識範圍
- 要求回答特定格式 (如「給我三個例子」)
- 追問細節迫使 AI 編造

7.3 應對策略

策略	說明
RAG (檢索增強生成)	讓 AI 先查文檔再回答，限制在可靠來源

策略	說明
驗證鏈 (Chain of Verification)	要求 AI 自我驗證答案
人類回饋 (RLHF)	透過人工標註減少幻覺
工具使用	讓 AI 查實時數據而非憑記憶回答
置信度標註	區分確定答案和不確定猜測

7.4 用戶應有的心態

- 永遠 double check：特別是事實性陳述、數據、引用
- 分級使用：創意寫作可以寬容，醫療法律建議必須驗證
- 了解局限：AI 是助手，不是權威

第八章：未來展望 — 人機協作的新紀元

9.1 短期趨勢 (1-3 年)

多模態融合：

- 文本、圖像、音頻、視頻的統一理解
- 例如：上傳一張照片，AI 能描述內容、提取文字、分析情緒

Agent 生態成熟：

- 標準化的 Agent 協議 (如 MCP - Model Context Protocol)
- 可組合的 Agent 網絡，專業 Agent 協作完成複雜任務

個人化記憶：

- AI 長期記住用戶的生活細節、偏好、目標
- 成為真正的「數字伴侶」

9.2 中期展望 (3-10 年)

科學研究加速：

- AI 協助假設生成、實驗設計、數據分析
- 在藥物發現、材料科學等領域突破

創意產業變革：

- 個人化電影、遊戲、音樂生成
- 創作門檻大幅降低，人人可成為創作者

教育個人化：

- AI 導師根據學生特點調整教學
- 終身學習成為常態

9.3 長期願景 (10 年以上)

通用人工智能 (AGI)：

- 具備人類水平的推理、創造、學習能力
- 能夠在陌生領域快速適應

人機融合：

- 腦機接口技術成熟
- 直接的思想與 AI 交互

社會結構轉型：

- 工作本質重新定義
- 需要新的經濟模式和社會契約

9.4 核心價值：增能而非取代

最理想的未來不是 AI 取代人類，而是 **AI 增強人類能力**：

- **創意**：AI 處理瑣事，人類專注創新
 - **決策**：AI 提供數據洞察，人類做價值判斷
 - **學習**：AI 個人化教學，人類掌握知識
 - **連結**：AI 打破語言障礙，人類深化理解
-

第九章：結論與建議

10.1 核心洞察

1. **AI 正從工具進化為伙伴**：從執行指令到理解意圖、主動規劃
2. **基礎設施決定能力邊界**：Token、API、Context Window 等技術細節深刻影響 AI 表現
3. **幻覺是結構性挑戰**：需要技術改進與人類監督並行
4. **自主可控是刚需**：OpenClaw 類方案滿足隱私、定制、成本等需求
5. **人機協作是未來**：成功的關鍵在於找到人類與 AI 的最佳分工

10.2 給使用者的建議

即學即用：

- 從日常任務入手，培養使用 AI 的習慣
- 學習 Prompt Engineering，提高溝通效率
- 建立「AI 輔助而非替代」的心態

持續學習：

- 關注 AI 發展動態，但避免過度追逐新工具
- 理解技術原理，建立批判性思維
- 投資那些 AI 難以替代的能力：創造力、同理心、價值判斷

負責任使用：

- 驗證 AI 輸出，特別是重要決策
- 保護個人隱私，謹慎分享敏感信息
- 認識 AI 的環境成本（訓練和運行的能耗）

10.3 給開發者的建議

架構設計：

- 設計時考慮人機協作，保留人工介入點
- 建立監控和回饋機制，持續改進 AI 表現
- 重視可解釋性，讓 AI 的決策透明可追溯

技術選型：

- 根據場景選擇模型，不一定追求最大最強
- 考慮自託管方案，平衡能力與控制
- 投資 RAG、微調等技術，提升特定領域表現

附錄

A. 術語表

術語	解釋
----	----

術語	解釋
LLM	Large Language Model，大語言模型
Token	AI 處理文字的基本單位
Context Window	AI 能同時處理的上下文長度
RAG	Retrieval-Augmented Generation，檢索增強生成
API	Application Programming Interface，應用程式介面
Agent	能自主規劃和行動的 AI 系統
Hallucination	AI 生成虛假內容的現象
Prompt	給 AI 的指令或問題
Fine-tuning	在特定數據上調整模型參數
Embeddings	將文字轉為向量表示的技術

B. 參考資源

- OpenAI Documentation: <https://platform.openai.com/docs>
- Anthropic Claude Documentation: <https://docs.anthropic.com>
- Google AI Documentation: <https://ai.google.dev>
- OpenClaw: <https://github.com/openclaw/openclaw>
- LangChain: <https://langchain.com>
- Hugging Face: <https://huggingface.co>

報告完結

本報告由筑夢編撰，旨在提供 AI 技術的全面概述與實用見解。

